Explore | Expand | Enrich

Cyber Security Test Projects

Ethnus
Mar 2025

Cyber Security Test Projects

College Level (2.5 hours - Basic Checks & Awareness)

- **Objective:** Perform basic security reconnaissance and configuration checks on provided Windows and Linux Virtual Machines (VMs). Identify examples related to common OWASP Top 10 vulnerabilities.
- **Scenario:** You are performing an initial security audit for "Innovate Startups". Your task is to check basic configurations on their standard employee workstation images (Windows/Linux) and identify potential risks based on descriptions of their web applications.
- **Prerequisites:**
  - Access to two provided Virtual Machines (VMs) running on the host Windows machine:
    - A Windows VM (e.g., Windows 10/11 or Server). Assume standard user login provided.
    - A Linux VM (e.g., Ubuntu Desktop/Server). Assume standard user login with `sudo` access provided.
  - Tools available within VMs: Web browser, Terminal/Command Prompt, Text Editor, `nmap` (on Linux VM or attacker VM if separate), Windows built-in tools (like `secpol.msc`, `services.msc`, `wf.msc`, `ipconfig`, `netstat`), Linux command-line tools (`ip`, `ss`, `grep`, `cat`, `systemctl`, `ufw` or `firewalld`).
  - Provided text file (`webapp_descriptions.txt`) containing simple descriptions or code snippets related to web vulnerabilities.
- **Detailed Steps/Modules:**
  - **Module 1: Windows VM Basic Checks (Approx. 45 mins)**
    - 1.1. Log into the provided Windows VM.
    - 1.2. **Firewall Check:** Open "Windows Defender Firewall with Advanced Security". Is the Domain Profile Firewall State 'On'? (Yes/No).

- 1.3. **Password Policy Check:** Open "Local Security Policy" (`secpol.msc`). Navigate to Account Policies -> Password Policy. What is the configured value for "Minimum password length"?
  - **Flag 1:** The numeric value found for Minimum password length.
- 1.4. **Running Services Check:** Open the Services console (`services.msc`). Find and list the exact "Service name" of **two** services that are currently 'Running' but might be considered non-essential for a basic workstation (e.g., Print Spooler, Fax, Remote Registry - choose based on VM setup).
- 1.5. **Network Check:** Open Command Prompt (`cmd`). Run `ipconfig /all`. What is the IPv4 Address of the primary network adapter? Run `netstat -ano | findstr "LISTENING"`. List one TCP port number (other than common ephemeral ports > 49151) that the system is listening on.
  - **Flag 2:** The IPv4 Address found.

○ **Module 2: Linux VM Basic Checks (Approx. 45 mins)**

- 2.1. Log into the provided Linux VM.
- 2.2. **SSH Configuration Check:** Open a terminal. View the SSH server configuration file (`sudo cat /etc/ssh/sshd_config`). What is the value set for `PermitRootLogin`? (e.g., `yes`, `no`, `prohibit-password`).
  - **Flag 3:** The value found for `PermitRootLogin`.
- 2.3. **Firewall Status Check:** Check the status of the firewall. Use `sudo ufw status` (if UFW is used) or `sudo firewall-cmd --state` (if firewalld is used). Is the firewall active/running? (Active/Inactive or Running/Not Running).
- 2.4. **Listening Ports Check:** Use the command `ss -tlpn` (or `netstat -tlpn`) to list listening TCP ports. Identify the port number used by the SSH service (`sshd`).
- 2.5. **File Permissions Check:** Navigate to the user's home directory (`cd ~`). Use `ls -l` to view file permissions. Find a file owned by the

user that has world-writeable permissions (e.g., `-rw-rw-rw-`). What is the name of this file? (A sample file like `insecure_file.txt` might be pre-created for this purpose).

- **Module 3: OWASP Top 10 Awareness (Approx. 45 mins)**

    - 3.1. Open the provided `webapp_descriptions.txt` file.
    - 3.2. **SQL Injection Identification:** Find the description or code snippet that shows user input being directly inserted into a database query string without proper sanitization or parameterization. Quote the line of code or the part of the description that demonstrates this vulnerability.
    - 3.3. **Cross-Site Scripting (XSS) Identification:** Find the description or code snippet where user input is taken and directly reflected back onto an HTML page without proper encoding or filtering. Quote the line of code or the part of the description that demonstrates this vulnerability.
    - 3.4. **Insecure Default Identification:** Find the description mentioning default credentials being used for an administrative interface. What is the default username mentioned?
        - **Flag 4:** The default username identified.

- **Module 4: Basic Network Scan (Local VMs) (Approx. 15 mins)**

    - 4.1. From the Linux VM's terminal (or a dedicated attacker VM if provided), use `nmap` to perform a basic TCP port scan against the **Private IP address** of the Windows VM (found in Flag 2).
    - 4.2. Command Example: `nmap <Windows_VM_IP>`
    - 4.3. List all open TCP ports reported by Nmap for the Windows VM target.

- **Deliverables:**

    - Flag 1: Numeric value of Minimum password length (Windows).
    - Flag 2: IPv4 Address of Windows VM.
    - List of two non-essential Running service names (Windows).
    - One listening TCP port number from `netstat` (Windows).
    - Flag 3: Value of `PermitRootLogin` (Linux).

- Firewall status (Linux).
- SSH port number (Linux).
- Name of the world-writeable file (Linux).
- Quoted line/description for SQL Injection example.
- Quoted line/description for XSS example.
- Flag 4: Default username identified (OWASP).
- List of open TCP ports found by Nmap on the Windows VM.

State Level (3.5 hours - Basic Hardening & Exploitation)

- **Objective:** Apply basic security hardening configurations to Windows and Linux VMs. Identify and perform simple exploits for common OWASP Top 10 vulnerabilities using a provided vulnerable web application. Analyze basic logs.
- **Scenario:** Following the initial audit, "Innovate Startups" asks you to apply some basic hardening settings and test their prototype web application for common flaws.
- **Prerequisites:**
    - Access to the same Windows and Linux VMs as the College level. Assume user accounts with administrative/sudo privileges.
    - Access to a simple, deliberately vulnerable web application (e.g., hosted on the Linux VM, or a separate provided VM/container like DVWA/Juice Shop accessible via browser from the test VMs). URL/IP address of the web app provided.
    - Tools: All tools from College level, plus potentially browser developer tools for web inspection. A simple log file (`auth.log` snippet or `web_access.log` snippet) provided.
- **Detailed Steps/Modules:**
    - **Module 1: Windows Hardening Application (Approx. 45 mins)**
        - 1.1. Log into the Windows VM.
        - 1.2. **Password Policy:** Using Local Security Policy (`secpol.msc`), enforce a "Minimum password length" of **12** characters.
        - 1.3. **Disable Service:** Using the Services console (`services.msc`), find the "Remote Registry" service (or another specified service), stop it, and set its Startup type to **Disabled**.
        - 1.4. **Firewall Rule:** Using "Windows Defender Firewall with Advanced Security", create a new **Inbound Rule**.
            - Rule Type: Port
            - Protocol: TCP
            - Specific local ports: `3389` (RDP)
            - Action: Block the connection
            - Profile: Apply to all profiles (Domain, Private, Public)

- Name: `Block External RDP`
- 1.5. **Verification:** Take a screenshot showing the new firewall rule ("Block External RDP") in the list of Inbound Rules OR provide the exact command using `netsh advfirewall` if applicable/trained.
  - **Flag 1:** Screenshot filename OR verification command.
- **Module 2: Linux Hardening Application (Approx. 45 mins)**
  - 2.1. Log into the Linux VM.
  - 2.2. **SSH Security:** Edit the SSH configuration file (`sudo nano /etc/ssh/sshd_config`). Change `PermitRootLogin` to `no`. Save the file and restart the SSH service (e.g., `sudo systemctl restart sshd`).
  - 2.3. **Firewall Configuration (UFW Example):**
    - Ensure UFW is installed (`sudo apt install ufw` if needed) and enabled (`sudo ufw enable`).
    - Set default policies: `sudo ufw default deny incoming`, `sudo ufw default allow outgoing`.
    - Allow SSH **only** from a specific private IP address (e.g., the Windows VM's private IP, assume `192.168.56.10` for this example): `sudo ufw allow from 192.168.56.10 to any port 22 proto tcp`.
    - Allow HTTP traffic: `sudo ufw allow 80/tcp`.
  - 2.4. **Firewall Configuration (firewalld Example):**
    - Ensure firewalld is running (`sudo systemctl status firewalld`).
    - Set the default zone if needed (e.g., `public`).
    - Remove existing SSH service from the public zone: `sudo firewall-cmd --zone=public --remove-service=ssh --permanent`.
    - Add a rich rule to allow SSH only from a specific IP (e.g., Windows VM IP `192.168.56.10`): `sudo firewall-cmd --zone=public --add-rich-rule='rule

`family="ipv4" source address="192.168.56.10" service name="ssh" accept' --permanent`.

        - Allow HTTP: `sudo firewall-cmd --zone=public --add-service=http --permanent`.
        - Reload firewall: `sudo firewall-cmd --reload`.
    - 2.5. **Verification:** Provide the output of `sudo ufw status verbose` OR `sudo firewall-cmd --list-all`.
        - **Flag 2:** Output of the firewall status command.
- **Module 3: Basic Log Analysis (Approx. 30 mins)**

    - 3.1. A log file snippet is provided (e.g., `auth_snippet.log` or `web_access_snippet.log`).
    - 3.2. **Failed Login Attempt:** If `auth_snippet.log` is provided, use `grep` or manual inspection to find an IP address associated with multiple "Failed password" attempts for a specific user. What is that IP address?
    - 3.3. **Suspicious Web Access:** If `web_access_snippet.log` is provided, use `grep` or manual inspection to find requests to unusual or potentially sensitive paths (e.g., `/admin`, `/config.bak`, `/../../`, requests with `UNION SELECT`). Identify one such suspicious requested path/URL.
    - **Flag 3:** The IP address with failed logins OR the suspicious web path identified.
- **Module 4: OWASP Top 10 - Simple Exploitation (Approx. 75 mins)**

    - 4.1. Access the provided vulnerable web application URL using the browser within one of the VMs.
    - 4.2. **Reflected Cross-Site Scripting (XSS):** Find an input field (like a search bar or a parameter in the URL) that reflects user input back onto the page without proper sanitization. Craft a simple payload that triggers a JavaScript alert box (e.g., `<script>alert('XSS-STATE-FLAG')</script>`). Execute the payload successfully.

        - **Flag 4:** The content displayed inside the alert box (`XSS-STATE-FLAG`).
        - Provide the vulnerable URL parameter name OR the name/label of the vulnerable input field.
    - 4.3. **Insecure Direct Object References (IDOR) / Broken Access Control:** Find a part of the application where you access user-specific data using an identifier in the URL (e.g., `view_profile.php?user_id=101`). Try changing the identifier to access another user's data (e.g., `user_id=102`). If successful, what piece of information (e.g., email address, username, full name) belonging to the *other* user is revealed?
        - **Flag 5:** The specific piece of information found for the other user.
        - Provide the URL parameter that was manipulated (e.g., `user_id`).

- **Deliverables:**

    - Flag 1: Screenshot filename or command verifying the Windows Firewall rule.
    - Flag 2: Output of the Linux firewall status command (`ufw status verbose` or `firewall-cmd --list-all`).
    - Flag 3: IP address with failed logins OR suspicious web path from log analysis.
    - Flag 4: Content of the XSS alert box.
    - Vulnerable parameter/field name for XSS.
    - Flag 5: Specific information revealed via IDOR.
    - URL parameter manipulated for IDOR.

National Level (5 hours - Scenario: Secure & Analyze)

- **Objective:** Perform more in-depth system hardening based on a scenario, analyze logs for Indicators of Compromise (IoCs), identify and exploit more complex OWASP vulnerabilities, and correlate findings.
- **Scenario:** "Innovate Startups" suspects their prototype web server (Linux VM) was recently compromised. They also want you to harden their new Windows database server VM according to best practices before it goes live. Your task is to investigate the potential compromise on the Linux VM using provided logs and system state, demonstrate a critical web vulnerability, and apply hardening configurations to the Windows VM.
- **Prerequisites:**
    - Access to the same Windows and Linux VMs, potentially with more logs or slightly different configurations. Assume administrative/sudo privileges.
    - Access to the vulnerable web application (potentially with more complex vulnerabilities like SQLi or Stored XSS).
    - Larger log files provided (e.g., `/var/log/auth.log`, `/var/log/apache2/access.log` or similar, Windows Event Logs snippets exported as `.evtx` if applicable).
    - Tools: All tools from State level, potentially including `lynis` or `chkrootkit` on Linux (if provided/allowed), Security Configuration Wizard or PowerShell scripting for Windows checks, potentially `sqlmap` for SQLi testing (if allowed and trained).
- **Detailed Steps/Modules:**

    - **Module 1: Windows Server Hardening (Scenario-Based) (Approx. 75 mins)**

        - 1.1. Log into the Windows VM (acting as the 'database server').
        - 1.2. **Auditing Configuration:** Using Local Security Policy (`secpol.msc`) or Group Policy Editor (`gpedit.msc`), navigate to Advanced Audit Policy Configuration. Enable auditing for:
            - Logon/Logoff -> Audit Logon (Success and Failure).

            - Object Access -> Audit File System (Success and Failure). (Note: Specific file/folder auditing needs SACLs configured separately, just enable the policy type).
        - 1.3. **Remove Unnecessary Features:** Using Server Manager or PowerShell (`Uninstall-WindowsFeature`), ensure features like "Windows Search Service" and "XPS Viewer" (or other specified non-essential features) are **not** installed.
        - 1.4. **Secure Protocol Configuration (Conceptual/Registry):** Describe the steps or registry keys you would modify to disable older insecure protocols like SMBv1 or SSLv3 (actual modification might be complex/risky in test).
        - 1.5. **Verification:** Provide a PowerShell command used to check if a specific Windows Feature (e.g., `Windows-Search-Service`) is installed (`Get-WindowsFeature -Name FeatureName | Select-Object Name, InstallState`).
            - **Flag 1:** Output of the `Get-WindowsFeature` command for a specified feature.
            - Briefly describe where (which policy section) Logon auditing is configured.
    - **Module 2: Linux Web Server - Log Analysis for IoCs (Approx. 75 mins)**

        - 2.1. Log into the Linux VM (the potentially compromised web server).
        - 2.2. **Identify Attacker IP:** Analyze the provided web server access logs (e.g., `/var/log/apache2/access.log`) and authentication logs (`/var/log/auth.log`). Look for patterns like: multiple 404 errors from one IP (scanning), failed login attempts followed by a successful login from the same IP, requests characteristic of vulnerability scanning tools (e.g., Nikto, sqlmap user agents), or access to webshell-like filenames. Identify a single IP address exhibiting multiple suspicious activities.
            - **Flag 2:** The suspected Attacker IP address.
        - 2.3. **Identify Compromise Timeframe:** Based on the logs (auth.log, web logs, file timestamps using `ls -l` in web directories if applicable), estimate the approximate date and time range when the

suspected compromise likely occurred (e.g., successful root login after multiple failures, first access to a webshell).

    - 2.4. **Identify Suspicious Files/Processes:** Check common web directories (`/var/www/html`, `/usr/share/nginx/html`) for recently modified or unusually named files (e.g., `.php` files with random names). Use `ps aux` or `top` to look for unusual running processes. List one suspicious filename or process name found.

- **Module 3: Linux Web Server - Deeper Hardening & Checks (Approx. 60 mins)**

    - 3.1. **Run Hardening Check Tool (Optional - if provided):** If `lynis` is available, run a basic scan (`sudo lynis audit system --quick`) and identify one high-priority hardening suggestion reported by the tool.
    - 3.2. **Check for Rootkits (Optional - if provided):** If `chkrootkit` is available, run a basic scan (`sudo chkrootkit`) and note if any infections or warnings are detected.
    - 3.3. **Review Cron Jobs:** Check system-wide and user cron jobs for any suspicious entries (`cat /etc/crontab`, `ls /etc/cron.*`, `crontab -l`). List any unusual commands found.
    - 3.4. **Verify Key Configs:** Manually verify critical settings:
        - SSH: Ensure `PasswordAuthentication` is set to `no` in `/etc/ssh/sshd_config`.
        - Firewall: Ensure the firewall rules (from State level or more strict) are still active and correctly configured.
    - 3.5. **Verification:** State whether `PasswordAuthentication` in sshd_config is set to `yes` or `no`. List one hardening suggestion from Lynis OR the result of the chkrootkit scan OR one suspicious cron entry found.
        - **Flag 3:** The hardening suggestion / chkrootkit result / suspicious cron command.

- **Module 4: OWASP Top 10 - SQL Injection Exploitation (Approx. 75 mins)**

    - 4.1. Access the provided vulnerable web application.

- 4.2. **Identify SQL Injection Point:** Find an input field or URL parameter that interacts with the database. Test for SQL injection vulnerabilities using common characters like single quotes (`'`), double quotes (`"`), or basic logical tests (e.g., `1' OR '1'='1`).
- 4.3. **Exploit SQL Injection:** Once a vulnerable parameter is found, perform **one** of the following (depending on application setup and training):
    - **Bypass Login:** If it's a login form, craft an input (e.g., for username field: `admin' --` or `admin' OR '1'='1`) that allows you to log in without knowing the password.
    - **Extract Data:** Use techniques like `UNION SELECT` to extract data from the database. For example, try to find the database version (`UNION SELECT @@version--`) or list table names from `information_schema`.
- 4.4. **Verification:**
    - Provide the name of the vulnerable parameter or input field.
    - Provide the specific SQL Injection payload you used successfully.
    - **Flag 4:** If login bypass: The welcome message or username displayed after successful login. OR If data extraction: The database version string OR one table name discovered from `information_schema`.

○ **Module 5: Correlate Findings & Scenario Conclusion (Briefly) (Approx. 15 mins)**

- 5.1. Based on your findings (Attacker IP, compromise time, suspicious files, SQLi vulnerability), briefly explain the likely attack chain in 1-2 sentences. (e.g., "The attacker from IP [Flag 2] likely exploited the [Module 4 Vulnerable Parameter] SQL injection vulnerability around [Compromise Timeframe] to gain access, possibly uploading the [Suspicious Filename] webshell.")

- **Deliverables:**

○ Flag 1: Output of `Get-WindowsFeature` command (Windows).

- Description of where Logon Auditing policy is configured (Windows).
- Flag 2: Suspected Attacker IP address (Linux Log Analysis).
- Estimated Compromise Timeframe (Linux Log Analysis).
- One suspicious filename or process name (Linux).
- Flag 3: Lynis suggestion / chkrootkit result / suspicious cron command (Linux Hardening).
- Verification of `PasswordAuthentication` setting (Linux Hardening).
- Vulnerable parameter/field name for SQL Injection (OWASP).
- Successful SQL Injection payload used (OWASP).
- Flag 4: Result of SQL Injection (Login message / DB Version / Table Name) (OWASP).
- Brief (1-2 sentence) explanation of the likely attack chain.

– End of Document –