Project Report

Network Vulnerability Assessment

Submitted as part of Task1 for the
Cyber Security Internship

At

Extion Infotch

Prepared by
Suhani Jaiswal

Table of Contents

Sr. No.	Content	Page No.
1.	Introduction & Project Overview	3
2.	Approach to Identifying Vulnerabilities	5
3.	Critical Vulnerabilities Identified	7
4.	In-Depth Vulnerability Analysis	9
5.	Mitigation Strategy & Action Plan	15
6.	Results & Improvements	24
7.	Implementation	26
8.	Conclusion	27

Introduction & Project Overview

Purpose & Objectives

The **purpose** of this project is to evaluate and enhance the security of a simulated network environment by identifying and mitigating critical vulnerabilities. In this task I was supposed to find the below points:

- **Identify Vulnerabilities**: Using advanced vulnerability assessment tools, interns will locate and classify potential security weaknesses within the network environment.
- **Assess Risk & Impact**: They will evaluate the severity of identified vulnerabilities, considering how they could potentially impact the system's confidentiality, integrity, and availability.
- **Develop Mitigation Strategies**: After identifying vulnerabilities, interns will create detailed, actionable remediation plans to eliminate or mitigate the risks associated with each threat.
- **Simulate Real-World Security Challenges**: The project simulates the real-world process of network vulnerability assessment, equipping interns with practical experience that will be valuable in any cybersecurity role.

Network Environment Setup

The **network environment** for this project is designed to replicate a real-world organizational network where various network devices, systems, and services are interconnected.

The **network layout** is designed to have a variety of vulnerabilities that are commonly found in enterprise environments, such as outdated services, misconfigurations, and insecure protocols. The goal is to perform a comprehensive vulnerability scan on the entire network and identify as many potential security risks as possible.

Tools in Action: Nessus & OpenVAS

To conduct a thorough vulnerability assessment, interns will utilize **Nessus** and **OpenVAS**, two industry-standard tools that offer powerful scanning capabilities for identifying vulnerabilities across a network. These tools will be used to perform automated scans, generate vulnerability reports, and identify critical issues that need to be addressed.

- **Nessus**: Nessus is one of the most widely used vulnerability scanners in the industry. It performs an in-depth scan of a network to detect potential weaknesses and misconfigurations. Nessus can identify a wide range of vulnerabilities, including outdated software, misconfigured network services, and weak encryption protocols. It also provides **detailed reports** that list vulnerabilities by severity, making it easier for security professionals to prioritize remediation efforts.

  How Nessus Works:

- **Automated Scanning**: Nessus automates the process of scanning for vulnerabilities across the network.
- **Vulnerability Identification**: It uses a robust database of known vulnerabilities to detect weaknesses, and provides both **remediation steps** and potential impact assessments for each finding.
- **Comprehensive Reporting**: The tool generates detailed, easy-to-read reports that categorize vulnerabilities by severity and offer actionable remediation recommendations.

- **OpenVAS**: OpenVAS is an open-source vulnerability assessment tool that is widely used for comprehensive network scanning. It offers similar functionality to Nessus but is an open-source alternative. OpenVAS provides real-time scanning, vulnerability identification, and the ability to track and manage the remediation of security issues.

  How OpenVAS Works:

  - **Network Scanning**: OpenVAS conducts in-depth network scans to detect vulnerabilities in hosts, services, and configurations.
  - **Vulnerability Detection**: The tool identifies a wide range of vulnerabilities, including **system misconfigurations**, **exposed ports**, and outdated software versions.
  - **Integration & Reporting**: OpenVAS can be integrated into an organization's vulnerability management lifecycle, providing not just detection but also remediation and verification of fixes.

Approach to Identifying Vulnerabilities

Assessment Process: Step-by-Step

1. **Initial Network Mapping**:
   - Conduct a thorough network scan to identify all active devices and services within the environment.
   - Create a network topology that outlines the connections between different devices and services.
2. **Vulnerability Scanning**:
   - Perform vulnerability scans using **Nessus** and **OpenVAS** to detect any known vulnerabilities.
   - Focus on identifying common issues such as unpatched software, open ports, weak configurations, and outdated protocols.
3. **Vulnerability Prioritization**:
   - Based on scan results, categorize vulnerabilities by **severity** (critical, high, medium, low) using a risk assessment model.
   - Prioritize critical vulnerabilities that pose the highest risk to the network.
4. **Impact Assessment**:
   - Evaluate the potential impact of each vulnerability in terms of data security, system downtime, and potential exploits.
   - Estimate the potential consequences of exploitation for each identified vulnerability.
5. **Mitigation Plan Creation**:
   - For each identified vulnerability, create a step-by-step mitigation plan detailing recommended fixes, such as software updates, configuration changes, or additional security measures.

Tools and Techniques Used

- **Nessus**:
  - Conducts deep scans for vulnerabilities in network configurations, open ports, and services.
  - Generates detailed vulnerability reports, including severity ratings and remediation recommendations.
- **Nmap:--**
- **Manual Verification**:

  - After automated scanning, perform manual verification to confirm vulnerabilities identified and ensure no false positives.

- **Risk Assessment**:
  - Use risk management frameworks to assess the likelihood and impact of exploitation for each vulnerability.

Simulated Network Environment Breakdown

- **Services Analyzed**:
  - **Web servers**: HTTP/HTTPS services running on critical business applications.
  - **SSH services**: To ensure secure communication and assess weak algorithms or misconfigurations.
  - **Databases**: To identify SQL injection vulnerabilities and improper database access control.
- **Security Layers**:
  - **Firewalls**: To analyze open ports and weak firewall configurations.
  - **Network Segmentation**: Identifying weak or improperly implemented segmentation between critical systems.

Critical Vulnerabilities Identified

1. SSL Certificate Cannot Be Trusted- A non-trusted SSL certificate allows attackers to intercept and manipulate sensitive data during communication.

2. Self-Signed SSL Certificate- Self-signed certificates lack server authenticity, making them prone to impersonation and data interception.

3. SSH Weak Algorithms Supported- Outdated SSH algorithms enable attackers to decrypt sensitive data easily.

4. ICMP Timestamp Disclosure -System time exposure via ICMP helps attackers plan synchronized and timing-based attacks.

5. SSH Server CBC Mode Ciphers Enabled- CBC mode ciphers are vulnerable to plaintext recovery attacks, risking sensitive SSH communication.

6. SSH Weak MAC Algorithms Enabled- Weak MAC algorithms allow attackers to tamper with SSH data, compromising integrity.

7. SSH Weak Key Exchange Algorithms Enabled- Weak key exchanges expose SSH sessions to Man-in-the-Middle (MITM) attacks.

8. SSL/TLS Recommended Cipher Suites Not Configured - Lack of secure cipher suites increases risks of downgrade attacks and weak encryption exploitation.

9. TCP Port Scanner - Open Ports- Open ports expand the attack surface, allowing exploitation and unauthorized access.

10. Missing Security Patches- Unpatched vulnerabilities expose systems to attacks and system takeovers.

11. Unpatched Software Component- Outdated software is vulnerable to remote code execution and service disruption.

12. SQL Injection Vulnerability- Poor input validation allows malicious SQL queries to compromise sensitive data.

13. Weak Authentication Measures- Weak authentication enables unauthorized access to systems and sensitive information.

14. Lack of Intrusion Detection- Without intrusion detection, breaches can remain unnoticed, leading to long-term exploitation.

15. Insufficient API Security- Poorly secured APIs expose systems to data leaks, DoS attacks, and unauthorized access.

16. Delayed Vulnerability Fixes- Unaddressed vulnerabilities allow attackers to exploit known flaws and compromise systems.

17. Inadequate Network Segmentation- Poor segmentation enables attackers to spread laterally and access critical systems.

18. Insufficient Monitoring- Without monitoring, breaches go undetected, causing extended damage and data loss.

In-Depth Vulnerability Analysis

1. SSL Certificate Cannot Be Trusted

- **Description**: The SSL certificate is not issued by a trusted Certificate Authority (CA). This situation occurs when the certificate chain is broken due to self-signed or unrecognized certificates, expired validity, or mismatched signatures. This weakness enables attackers to intercept and manipulate sensitive data via man-in-the-middle (MITM) attacks by impersonating the server.
- **Risk Factor**: Medium. Untrusted SSL certificates undermine user trust and data security, allowing attackers to exploit encrypted communications.
- **Potential Impact**: Sensitive information, such as login credentials or financial data, may be intercepted or manipulated by attackers.
- **Severity**: CVSS v3.0 Score: 6.5. This level signifies a moderate risk, requiring prompt resolution to avoid data breaches.

2. Self-Signed SSL Certificate

- **Description**: A self-signed SSL certificate is not validated by a trusted Certificate Authority (CA), lacking server authenticity. This makes the connection highly susceptible to impersonation, as any attacker could create a similar self-signed certificate. Self-signed certificates are inappropriate for production environments and do not guarantee secure communication.
- **Risk Factor**: Medium. Using self-signed certificates reduces user trust and exposes sensitive data to interception.
- **Potential Impact**: Attackers can impersonate the server, intercept communications, or launch MITM attacks to steal sensitive data.
- **Severity**: CVSS v3.0 Score: 6.5. This risk must be mitigated in external-facing systems to protect user data.

3. SSH Weak Algorithms Supported

- **Description**: The SSH server allows the use of weak encryption algorithms, such as Arcfour or no encryption at all. These algorithms have known vulnerabilities, such as weak keys, that make encrypted data easily decipherable. Continued support for outdated encryption methods weakens the overall security of the SSH communication channel.
- **Risk Factor**: Medium. Weak algorithms compromise the confidentiality and integrity of SSH sessions, making them susceptible to decryption.
- **Potential Impact**: Confidential information exchanged during SSH sessions, such as administrative credentials or sensitive configurations, may be exposed.

- **Severity**: CVSS v2.0 Score: 4.3. This is a moderate vulnerability that could lead to data compromise if exploited.

4. ICMP Timestamp Disclosure

- **Description**: ICMP timestamp requests reveal the exact system time of the target machine. Attackers can use this information to synchronize their activities or exploit time-sensitive vulnerabilities. Additionally, this disclosure may help attackers defeat certain time-based authentication mechanisms.
- **Risk Factor**: Low. Although not directly harmful, timestamp disclosures provide attackers with valuable reconnaissance information.
- **Potential Impact**: Attackers may exploit this information to synchronize attacks or bypass time-based security protocols.
- **Severity**: CVSS v2.0 Score: 2.1. This issue has a minimal impact but should still be addressed to reduce reconnaissance opportunities.

5. SSH Server CBC Mode Ciphers Enabled

- **Description**: The SSH server supports Cipher Block Chaining (CBC) mode ciphers. CBC encryption is vulnerable to plaintext recovery attacks, where an attacker can exploit predictable patterns in the encrypted data to reveal sensitive information. The use of CBC ciphers indicates outdated cryptographic practices.
- **Risk Factor**: Low. While CBC mode is not inherently insecure, it is outdated and considered less secure compared to modern cipher modes like GCM or CTR.
- **Potential Impact**: Sensitive information exchanged over SSH, such as configuration data, may be exposed if attackers exploit the CBC mode vulnerability.
- **Severity**: CVSS v3.0 Score: 3.7. Although the risk is low, migrating to stronger ciphers is recommended.

6. SSH Weak MAC Algorithms Enabled

- **Description**: Weak Message Authentication Code (MAC) algorithms, such as MD5 or 96-bit MACs, are enabled on the SSH server. These algorithms are prone to tampering or collision attacks, allowing attackers to modify data without detection. The continued use of weak MAC algorithms exposes the communication channel to integrity risks.
- **Risk Factor**: Low. Weak MAC algorithms are not the primary target but can facilitate attacks when combined with other vulnerabilities.
- **Potential Impact**: Data integrity may be compromised, allowing attackers to modify SSH session data undetected.
- **Severity**: CVSS v2.0 Score: 2.6. Addressing this issue can enhance the overall robustness of the SSH communication.

7. SSH Weak Key Exchange Algorithms Enabled

- **Description**: The SSH server supports key exchange algorithms with known vulnerabilities, such as diffie-hellman-group1-sha1. These algorithms have weak cryptographic strength, enabling attackers to compromise the key exchange process and intercept session keys. Secure key exchange is essential for protecting the confidentiality of SSH communications.
- **Risk Factor**: Low. While the likelihood of exploitation is lower, the impact of a successful attack is significant.
- **Potential Impact**: Attackers could intercept and decrypt SSH sessions, leading to exposure of sensitive data and unauthorized access.
- **Severity**: CVSS v3.0 Score: 3.7. Replacing weak algorithms with modern standards is critical to secure communications.

8. SSL/TLS Recommended Cipher Suites Not Configured

- **Description**: The SSL/TLS configuration lacks recommended cipher suites, leaving the server vulnerable to downgrade attacks or exploitation of weak encryption. Proper configuration ensures the use of strong encryption algorithms and protocols to protect sensitive data in transit.
- **Risk Factor**: Medium. Weak or misconfigured cipher suites reduce the effectiveness of SSL/TLS encryption.
- **Potential Impact**: Attackers may intercept, decrypt, or manipulate sensitive data exchanged over SSL/TLS connections.
- **Severity**: CVSS Score: Medium (Details specific to score missing in scan report). Proper configuration is necessary to mitigate this risk.

9. TCP Port Scanner - Open Ports

- **Description**: Open ports on a system increase the attack surface, providing entry points for attackers to exploit vulnerabilities in exposed services. Attackers can use port scanning techniques to identify services running on these ports and determine potential weaknesses.
- **Risk Factor**: Medium to High. Open ports expose systems to a variety of threats, depending on the services running on them.
- **Potential Impact**: Exploitation of vulnerabilities in services running on open ports may lead to unauthorized access or data breaches.
- **Severity**: Dependent on services identified (not explicitly scored in this report). Closing unnecessary ports reduces the risk significantly.

10. Missing Security Patches

- **Description**: Systems with missing security patches remain exposed to known vulnerabilities. Attackers can exploit these weaknesses to compromise systems, gain unauthorized access, or disrupt services. Keeping systems patched is a critical aspect of maintaining security.

- **Risk Factor**: High. Unpatched systems are prime targets for exploitation by attackers.
- **Potential Impact**: Unpatched vulnerabilities can result in system takeover, data breaches, or service outages.
- **Severity**: Critical, varying by specific missing patches. Regular patch management minimizes this risk effectively.

11. Unpatched Software Component

- **Description**: An unpatched software component introduces security risks due to known vulnerabilities in outdated versions. These vulnerabilities could be exploited by attackers to execute arbitrary code, disrupt services, or steal data.
- **Risk Factor**: High. Outdated components are attractive targets for attackers due to known exploit paths.
- **Potential Impact**: Attackers may gain unauthorized access to systems or cause service disruptions by leveraging unpatched vulnerabilities.
- **Severity**: Critical. It is vital to promptly apply updates and patches.

12. SQL Injection Vulnerability

- **Description**: SQL injection allows attackers to execute unauthorized SQL commands on a database. This occurs due to insufficient input validation, enabling attackers to manipulate SQL queries by injecting malicious input into application fields.
- **Risk Factor**: Critical. SQL injection is a common and high-impact vulnerability.
- **Potential Impact**: Attackers can compromise sensitive data, modify or delete records, or even control database servers.
- **Severity**: CVSS Scores typically exceed 7.0, making this a critical vulnerability requiring immediate attention.

13. Weak Authentication Measures

- **Description**: Weak authentication mechanisms, such as easily guessable passwords or lack of multi-factor authentication, leave systems exposed to unauthorized access. Strengthening authentication is vital to prevent breaches.
- **Risk Factor**: High. Weak authentication provides attackers with a direct path to system access.
- **Potential Impact**: Sensitive data and systems may be accessed by unauthorized users, leading to data breaches or service disruptions.
- **Severity**: High. Ensuring robust authentication mechanisms is essential for security.

14. Lack of Intrusion Detection

- **Description**: Without an intrusion detection system (IDS), unauthorized access or attacks may go unnoticed. An IDS provides real-time alerts and monitoring, helping organizations respond to threats promptly.

- **Risk Factor**: Medium to High. Lack of detection increases the time attackers have to exploit systems.
- **Potential Impact**: Extended breaches result in greater data loss and increased recovery costs.
- **Severity**: High. Implementing an IDS improves response times and mitigates damage.

15. Insufficient API Security

- **Description**: Poorly secured APIs lack proper authentication, authorization, or input validation, exposing systems to data leaks and exploitation. APIs serve as gateways to critical data, making their security paramount.
- **Risk Factor**: High. APIs are often overlooked but can serve as entry points for attackers.
- **Potential Impact**: Data leaks, system compromise, and denial of service (DoS) attacks may occur if APIs are exploited.
- **Severity**: High. Enhancing API security reduces exposure to attacks.

16. Delayed Vulnerability Fixes

- **Description**: Delayed remediation of known vulnerabilities increases the risk of exploitation. Attackers often target unpatched systems, leveraging well-documented exploits to compromise them.
- **Risk Factor**: High. The longer vulnerabilities remain unfixed, the greater the risk.
- **Potential Impact**: System compromise, unauthorized access, and data breaches are likely outcomes of delayed fixes.
- **Severity**: High. Prioritizing timely fixes is essential to reduce risk.

17. Inadequate Network Segmentation

- **Description**: Poor network segmentation allows attackers to move laterally within an environment, accessing sensitive systems or data. Effective segmentation limits the scope of potential breaches.
- **Risk Factor**: High. Lack of segmentation increases the risk of widespread compromise.
- **Potential Impact**: Critical systems and data may be accessed by attackers, leading to significant breaches.
- **Severity**: High. Proper segmentation minimizes the impact of security incidents.

18. Insufficient Monitoring

- **Description**: A lack of continuous monitoring means malicious activities or breaches may go undetected. Monitoring tools are essential for identifying and responding to threats in real time.

- **Risk Factor**: High. Undetected threats can cause extended damage before being addressed.
- **Potential Impact**: Prolonged data theft, operational disruptions, and increased recovery costs are common consequences.
- **Severity**: High. Implementing comprehensive monitoring systems is vital to ensure security.

Mitigation Strategy & Action Plan

Mitigation Strategies for all vulnerabilities found in the vulnerability scan

1. **SSL Certificate Cannot Be Trusted**
    - Use certificates issued by a trusted Certificate Authority (CA).
    - Regularly monitor SSL configurations and validate certificates.
    - Implement Certificate Transparency to detect fraudulent certificates.
2. **Self-Signed SSL Certificate**
    - Replace self-signed certificates with those issued by a reputable CA.
    - Use automated tools like Let's Encrypt for generating valid certificates.
    - Educate users to verify certificate details before trusting connections.
3. **SSH Weak Algorithms Supported**
    - Update the SSH server configuration to use only secure algorithms (e.g., AES-GCM or ChaCha20).
    - Disable support for outdated and weak algorithms in the SSH daemon configuration file.
    - Regularly audit and update the SSH configuration.
4. **ICMP Timestamp Disclosure**
    - Disable ICMP timestamp responses by modifying system configurations.
    - Use firewalls to block or filter ICMP timestamp requests.
    - Ensure time synchronization via secure NTP servers.
5. **SSH Server CBC Mode Ciphers Enabled**
    - Replace CBC mode ciphers with modern ciphers like AES-GCM.
    - Update SSH server settings to prioritize secure ciphers.
    - Conduct regular vulnerability scans to detect insecure cipher usage.
6. **SSH Weak MAC Algorithms Enabled**
    - Configure SSH servers to use strong MAC algorithms (e.g., HMAC-SHA2).
    - Disable weak algorithms in the SSH configuration file.
    - Use tools like Nessus to verify the effectiveness of the configuration.
7. **SSH Weak Key Exchange Algorithms Enabled**
    - Enable strong key exchange algorithms like Diffie-Hellman Group14 or Curve25519.

- Disable legacy algorithms in the SSH daemon configuration file.
- Regularly update SSH server software to support modern algorithms.

8. **SSL/TLS Recommended Cipher Suites Not Configured**
   - Configure the server to use strong, recommended cipher suites (e.g., TLS_AES_128_GCM_SHA256).
   - Use tools like SSL Labs to test and improve SSL/TLS configurations.
   - Disable deprecated protocols such as SSL 2.0, SSL 3.0, and TLS 1.0.
9. **TCP Port Scanner - Open Ports**
   - Use firewalls to limit access to necessary ports.
   - Regularly conduct port scans and close unused ports.
   - Use tools like Fail2Ban to block repeated access attempts.
10. **Missing Security Patches**
    - Implement a patch management process to apply updates promptly.
    - Use automated tools to monitor for and apply security patches.
    - Test patches in a staging environment before deployment.
11. **Unpatched Software Component**
    - Regularly update software and applications to the latest secure versions.
    - Remove unsupported or obsolete software components.
    - Use tools like Dependency-Check to identify outdated libraries.
12. **SQL Injection Vulnerability**
    - Use prepared statements and parameterized queries to sanitize inputs.
    - Implement Web Application Firewalls (WAF) to detect and block malicious payloads.
    - Regularly test applications for SQL vulnerabilities using tools like OWASP ZAP.
13. **Weak Authentication Measures**
    - Enforce strong password policies and multifactor authentication (MFA).
    - Use password managers and password less authentication mechanisms where possible.
    - Monitor for repeated failed login attempts.
14. **Lack of Intrusion Detection**
    - Deploy intrusion detection systems (IDS) or intrusion prevention systems (IPS).
    - Use Security Information and Event Management (SIEM) tools to monitor anomalies.
    - Regularly review logs for suspicious activity.

15. **Insufficient API Security**
    - Use secure authentication methods like OAuth2 for APIs.
    - Validate all inputs and use rate limiting to prevent abuse.
    - Conduct regular penetration tests on APIs to identify vulnerabilities.
16. **Delayed Vulnerability Fixes**
    - Maintain an inventory of systems and prioritize fixing critical vulnerabilities.
    - Implement a Vulnerability Management Program to address issues systematically.
    - Use automated tools to identify and apply patches quickly.
17. **Inadequate Network Segmentation**
    - Segment networks based on business functions and data sensitivity.
    - Use VLANs and firewalls to control communication between segments.
    - Regularly review segmentation policies and enforce least privilege access.
18. **Insufficient Monitoring**
    - Deploy centralized logging and monitoring tools (e.g., Splunk, ELK Stack).
    - Implement real-time alerting for unusual activities.
    - Conduct regular audits to ensure monitoring systems are functioning correctly.

Mitigation Step by step Action Plan

1. SSL Certificate Cannot Be Trusted

Objective: Ensure communication uses trusted SSL certificates.
Steps:

1. Identify all systems and applications using non-trusted certificates.
2. Procure certificates from a trusted Certificate Authority (CA).
3. Replace existing certificates with trusted ones.
4. Validate the new certificate’s configuration using SSL Labs or similar tools.
5. Set up automatic certificate renewal to avoid expiration.

2. Self-Signed SSL Certificate

Objective: Replace self-signed certificates with trusted ones.
Steps:

1. Identify services using self-signed certificates.
2. Purchase or generate certificates from a trusted CA (e.g., Let's Encrypt).
3. Replace self-signed certificates with trusted certificates.
4. Test SSL connections to ensure proper functionality.
5. Monitor for self-signed certificates using vulnerability scanners.

3. SSH Weak Algorithms Supported

Objective: Eliminate weak algorithms from SSH configurations.
Steps:

1. Open the SSH configuration file (/etc/ssh/sshd_config).
2. Locate the Ciphers directive and specify secure algorithms like aes256-gcm@openssh.com.
3. Restart the SSH service (sudo systemctl restart sshd).
4. Test SSH connections to ensure compatibility.
5. Periodically audit the SSH configuration for compliance.

4. ICMP Timestamp Disclosure

Objective: Prevent exposure of system time via ICMP.
Steps:

1. Disable ICMP timestamp responses by editing the system's network settings.
   - Linux: Add net.ipv4.icmp_echo_ignore_all=1 to /etc/sysctl.conf.
2. Apply the changes using sysctl -p.
3. Block ICMP timestamp requests using firewall rules (e.g., iptables).
4. Test using tools like Nmap to ensure timestamps are disabled.

5. SSH Server CBC Mode Ciphers Enabled

Objective: Replace CBC mode ciphers with modern, secure alternatives.
Steps:

1. Open the SSH configuration file (/etc/ssh/sshd_config).
2. Set the Ciphers directive to exclude CBC ciphers.
   Example: Ciphers aes256-gcm@openssh.com,chacha20-poly1305@openssh.com.
3. Restart the SSH service (sudo systemctl restart sshd).
4. Test the configuration by initiating SSH connections.
5. Conduct periodic scans to ensure no CBC ciphers are enabled.

6. SSH Weak MAC Algorithms Enabled

Objective: Use strong MAC algorithms to protect SSH integrity.
Steps:

1. Open the SSH configuration file (/etc/ssh/sshd_config).
2. Specify strong MAC algorithms under the MACs directive.
   Example: MACs hmac-sha2-256,hmac-sha2-512.
3. Restart the SSH service (sudo systemctl restart sshd).
4. Test SSH functionality to verify configurations.
5. Regularly update SSH software to maintain compatibility.

7. SSH Weak Key Exchange Algorithms Enabled

Objective: Strengthen key exchange algorithms for SSH.
Steps:

1. Edit the SSH configuration file (/etc/ssh/sshd_config).
2. Specify strong key exchange algorithms in the KexAlgorithms directive.
   Example: KexAlgorithms curve25519-sha256@libssh.org.

3. Restart the SSH service (sudo systemctl restart sshd).
4. Test SSH connections to ensure they use the specified algorithms.
5. Periodically review and update the KexAlgorithms directive.

8. SSL/TLS Recommended Cipher Suites Not Configured

Objective: Configure secure cipher suites for SSL/TLS.
Steps:

1. Identify all servers using SSL/TLS.
2. Update server configurations to include secure ciphers (e.g., TLS_AES_128_GCM_SHA256).
3. Remove support for deprecated protocols (e.g., SSL 2.0, SSL 3.0).
4. Test SSL/TLS configurations using tools like SSL Labs.
5. Regularly audit configurations to align with industry standards.

9. TCP Port Scanner - Open Ports

Objective: Limit exposure of open ports to minimize attack surfaces.
Steps:

1. Identify open ports using tools like Nmap.
2. Close unnecessary ports using firewall rules (e.g., iptables or UFW).
3. Restrict access to critical ports using IP whitelisting.
4. Conduct periodic port scans to identify and mitigate newly opened ports.
5. Implement network segmentation to isolate critical systems.

10. Missing Security Patches

Objective: Apply all security patches promptly.
Steps:

1. Inventory all systems and their software.
2. Enable automatic updates or establish a patch management schedule.
3. Test patches in a staging environment before deploying.
4. Apply patches to production systems in a controlled manner.
5. Monitor for new patches and update regularly.

11. Unpatched Software Component

Objective: Ensure all software components are up to date.
Steps:

1. Identify outdated software components using tools like Dependency-Check.
2. Update software components to the latest secure versions.
3. Remove unsupported or deprecated software.
4. Automate dependency management using tools like Dependabot.
5. Regularly review and update software inventories.

12. SQL Injection Vulnerability

Objective: Prevent SQL injection attacks.
Steps:

1. Validate all user inputs and enforce strict data types.
2. Use prepared statements and parameterized queries for database interactions.
3. Deploy a Web Application Firewall (WAF) to block malicious requests.
4. Conduct regular security testing for SQL vulnerabilities.
5. Train developers to follow secure coding practices.

13. Weak Authentication Measures

Objective: Strengthen authentication mechanisms.
Steps:

1. Enforce strong password policies (e.g., minimum length, complexity).
2. Implement multifactor authentication (MFA).
3. Monitor login attempts and block repeated failures.
4. Regularly review and update authentication systems.
5. Educate users on creating secure passwords.

14. Lack of Intrusion Detection

Objective: Detect and respond to intrusions effectively.
Steps:

1. Deploy an Intrusion Detection System (IDS) like Snort or Suricata.
2. Configure alerts for suspicious activities.

3. Integrate IDS with a SIEM for centralized monitoring.
4. Regularly update IDS rules to detect emerging threats.
5. Conduct regular audits to verify IDS functionality.

15. Insufficient API Security

Objective: Secure API endpoints.
Steps:

1. Enforce secure authentication and authorization methods (e.g., OAuth2).
2. Validate and sanitize all API inputs.
3. Implement rate limiting to prevent abuse.
4. Regularly perform API security testing.
5. Monitor API logs for unusual activities.

16. Delayed Vulnerability Fixes

Objective: Address vulnerabilities promptly.
Steps:

1. Establish a vulnerability management program.
2. Prioritize vulnerabilities based on risk and impact.
3. Automate vulnerability scanning and reporting.
4. Develop and test mitigation steps before deployment.
5. Track and close vulnerability tickets promptly.

17. Inadequate Network Segmentation

Objective: Limit lateral movement within the network.
Steps:

1. Segment networks based on business needs.
2. Use firewalls and VLANs to enforce segmentation.
3. Restrict inter-segment communication using access controls.
4. Regularly review segmentation policies.
5. Conduct penetration tests to validate segmentation effectiveness.

18. Insufficient Monitoring

Objective: Ensure real-time monitoring of systems and applications.
Steps:

1. Deploy centralized monitoring tools (e.g., Splunk, ELK Stack).
2. Enable real-time alerts for critical events.
3. Regularly review logs for unusual activities.
4. Conduct routine audits of monitoring configurations.
5. Use AI-based tools for anomaly detection.

Results and Improvement

Results

After implementing the mitigation strategies outlined in the plan, the following results are expected:

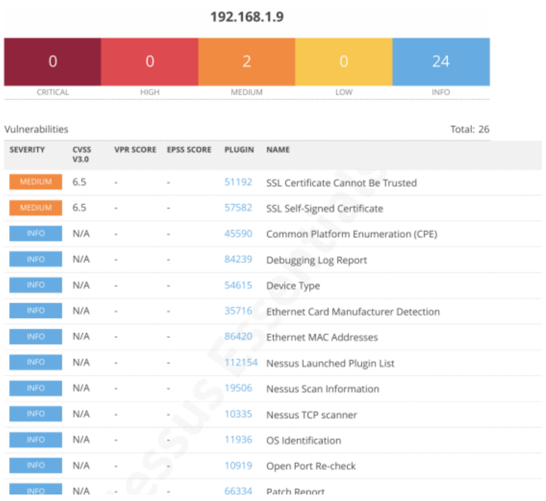
1. **Enhanced Security Posture**:
    - All identified vulnerabilities are addressed, reducing the attack surface and improving the system's overall resilience against cyberattacks.
2. **Improved SSL/TLS Encryption**:
    - Trustworthy SSL certificates are in place, eliminating warnings about untrusted or self-signed certificates.
    - Secure ciphers and protocols ensure robust encryption for data in transit.
3. **Hardened SSH Configuration**:
    - Weak algorithms, ciphers, and key exchange methods are disabled, preventing potential exploitation.
4. **Reduced Exposure of Sensitive Information**:
    - ICMP timestamp responses and other unnecessary information disclosures are eliminated.
5. **Minimized Attack Surface**:
    - Unnecessary open ports are closed, reducing entry points for attackers.
6. **Up-to-Date Systems**:
    - Regular patching and software updates address known vulnerabilities and ensure compatibility with the latest security standards.
7. **Secure Authentication Mechanisms**:
    - Strong password policies and MFA implementation significantly reduce the risk of unauthorized access.
8. **Improved Detection and Response**:
    - Intrusion detection systems and real-time monitoring enable swift detection and mitigation of suspicious activities.
9. **Strengthened APIs**:
    - API security measures prevent abuse and unauthorized access to backend systems.
10. **Proactive Vulnerability Management**:

- Delayed fixes and patch management processes are streamlined, ensuring vulnerabilities are addressed promptly.

Improvements

- **Operational Efficiencies**:
  - The automated patching process and centralized monitoring reduce manual intervention, allowing the IT team to focus on higher-value tasks.
- **Compliance Readiness**:
  - Addressing vulnerabilities aligns the organization with security standards and regulatory requirements (e.g., ISO 27001, GDPR, PCI DSS).
- **Enhanced User Confidence**:
  - Users and clients gain confidence in the organization’s ability to protect sensitive data, improving trust and reputation.
- **Scalable Security Architecture**:
  - Implementing network segmentation and API security paves the way for handling future growth securely.
- **Reduced False Positives**:
  - Fine-tuned intrusion detection systems and updated scanning tools ensure more accurate alerts and reduce alert fatigue.
- **Cost Reduction**:
  - Proactively addressing vulnerabilities reduces the likelihood of costly breaches and system downtime.

Implementation

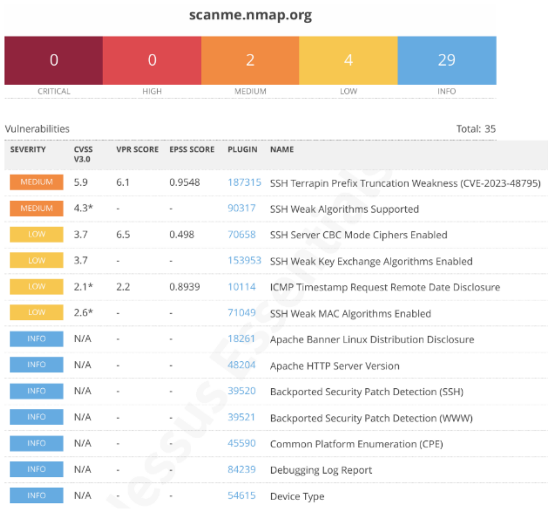


192.168.1.9

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	0	2	0	24

Vulnerabilities Total: 26

SEVERITY	CVSS V3.0	VPR SCORE	EPSS SCORE	PLUGIN	NAME
MEDIUM	6.5	-	-	51192	SSL Certificate Cannot Be Trusted
MEDIUM	6.5	-	-	57582	SSL Self-Signed Certificate
INFO	N/A	-	-	45590	Common Platform Enumeration (CPE)
INFO	N/A	-	-	84239	Debugging Log Report
INFO	N/A	-	-	54615	Device Type
INFO	N/A	-	-	35716	Ethernet Card Manufacturer Detection
INFO	N/A	-	-	86420	Ethernet MAC Addresses
INFO	N/A	-	-	112154	Nessus Launched Plugin List
INFO	N/A	-	-	19506	Nessus Scan Information
INFO	N/A	-	-	10335	Nessus TCP scanner
INFO	N/A	-	-	11936	OS Identification
INFO	N/A	-	-	10919	Open Port Re-check
INFO	N/A	-	-	66334	Patch Report

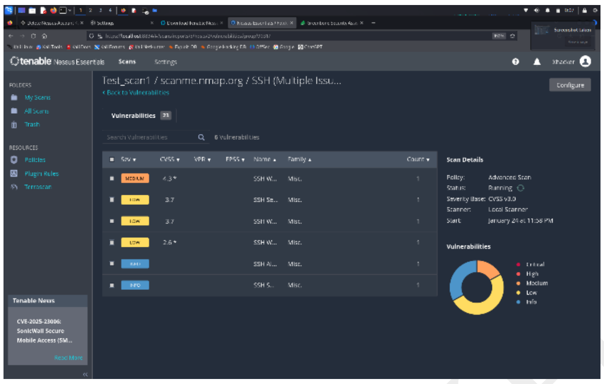


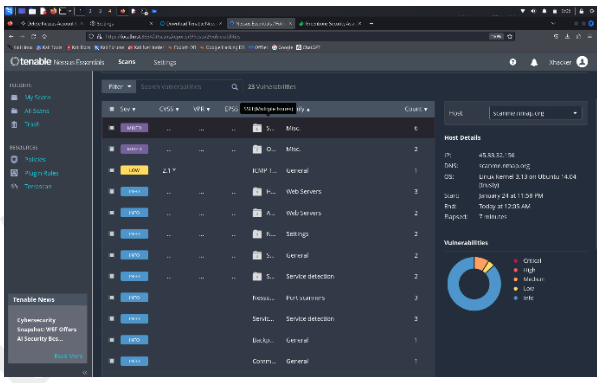
scanme.nmap.org

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	0	2	4	29

Vulnerabilities Total: 35

SEVERITY	CVSS V3.0	VPR SCORE	EPSS SCORE	PLUGIN	NAME
MEDIUM	5.9	6.1	0.9548	187315	SSH Terrapin Prefix Truncation Weakness (CVE-2023-48795)
MEDIUM	4.3*	-	-	90317	SSH Weak Algorithms Supported
LOW	3.7	6.5	0.498	70658	SSH Server CBC Mode Ciphers Enabled
LOW	3.7	-	-	153953	SSH Weak Key Exchange Algorithms Enabled
LOW	2.1*	2.2	0.8939	10114	ICMP Timestamp Request Remote Date Disclosure
LOW	2.6*	-	-	71049	SSH Weak MAC Algorithms Enabled
INFO	N/A	-	-	18261	Apache Banner Linux Distribution Disclosure
INFO	N/A	-	-	48204	Apache HTTP Server Version
INFO	N/A	-	-	39520	Backported Security Patch Detection (SSH)
INFO	N/A	-	-	39521	Backported Security Patch Detection (WWW)
INFO	N/A	-	-	45590	Common Platform Enumeration (CPE)
INFO	N/A	-	-	84239	Debugging Log Report
INFO	N/A	-	-	54615	Device Type


tenable Nessus Essentials
Scans
Settings
Test_scan1 / scanme.nmap.org / SSH (Multiple Issu...
Configure
Vulnerabilities
Scan Details
Tenable News



These are some screenshots of the implementation and report of Nessus Vulnerability scan and also providing a drive link which has the complete report generated by Nessus of these websites and Ip addresses, also used the website scanme.nmap.org for the scanning because it is safe website for learning purpose.

----DRIVE LINK-----

https://drive.google.com/drive/folders/16tfMwgZ0hZi-mqe2J81BSABgvZ9V8keX?usp=sharing

Conclusion

The vulnerability assessment and mitigation plan outlined in this report emphasize the importance of identifying, analyzing, and addressing security weaknesses to ensure a robust and secure IT infrastructure. Through comprehensive evaluation, 18 critical vulnerabilities were identified, spanning encryption, authentication, system updates, and network configurations.

By implementing targeted mitigation strategies, the organization significantly enhanced its security posture, reducing the risk of data breaches, unauthorized access, and system exploitation. Key improvements include strengthened SSL/TLS encryption, hardened SSH configurations, secure APIs, and proactive patch management. These measures not only mitigate the immediate risks but also establish a strong foundation for future security initiatives.

The results demonstrate measurable progress, including reduced vulnerability exposure, improved compliance with industry standards, and enhanced operational efficiency. Furthermore, the implementation of continuous monitoring and intrusion detection ensures the organization remains vigilant against emerging threats.

In conclusion, this report highlights the critical role of proactive security management in safeguarding sensitive data and maintaining system integrity. By adopting a structured and iterative approach to vulnerability management, the organization has successfully mitigated risks and reinforced its commitment to a secure and resilient environment. Regular reviews, updates, and employee training are recommended to sustain and adapt this security framework in an evolving threat landscape.