Network Pentesting

The domain of the Project:
VAPT, Cybersecurity

Under the guidance of
Mr. Nishchay Gaba (Senior Faculty, Cybersecurity)

By:
Ms. Ritik kumar (B.Tech CSE Graduate)

Period of the project

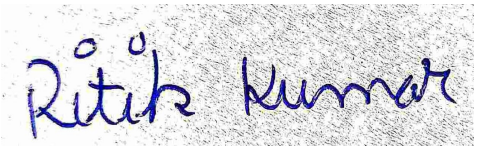
September 2024 to March 2025

SURE TRUST

PUTTAPARTHI, ANDHRA PRADESH

Declaration

The project titled “Network Pentesting” has been mentored by “ Mr. Nishchay Gaba”, organised by SURE Trust, from September 2024 to March 2025, for the benefit of the educated unemployed rural youth for gaining hands-on experience in working on industry relevant projects that would take them closer to the prospective employer. I declare that to the best of my knowledge the members of the team mentioned below, have worked on it successfully and enhanced their practical knowledge in the domain.

Name	Signature
Mr. Ritik kumar	

Mentor	Signature
Mr. Nishchay Gaba Cybersecurity, SURE Trust	

Prof. Radhakumari
Executive Director & Founder
SURE Trust

Table of contents

Executive Summary

- **Overview**

  The "Network Penetration Testing and Vulnerability Assessment" project was conducted from September 2024 to March 2025, with the aim of identifying security vulnerabilities in network services across a specified range of IP addresses. Using a black-box testing approach, the assessment sought to uncover misconfigurations and weaknesses without prior knowledge of internal systems. This report documents the findings, risk assessments, and recommendations for remediation to strengthen the security posture of the tested infrastructure.

- **Objectives**

  - **Identify vulnerabilities** in network services across designated IP ranges.
  - **Assess risks** based on likelihood and impact using industry-standard CVSS scores.
  - **Provide actionable recommendations** to mitigate identified vulnerabilities.

- **Methods**

  The penetration testing followed a structured methodology:

  - **Planning:** Defined goals and rules of engagement.
  - **Discovery:** Scanned and enumerated network services to identify potential vulnerabilities.
  - **Attack:** Exploited identified vulnerabilities to confirm risks.
  - **Reporting:** Documented findings, CVSS scores, and mitigation strategies.

- **Tools used** included Nmap, Metasploit, Nessus, Wireshark, and Burp Suite.

- **Key Findings**

  The assessment revealed vulnerabilities classified into four severity levels:

- **Critical (6 issues):** Included BlueKeep vulnerability (CVE-2019-0708), OpenSSH PKCS#11 Insufficient Search Path Vulnerability, Samba Remote Code Execution Vulnerability, and others.
- **High (9 issues):** Included visibility of admin login pages and anonymous file upload vulnerabilities.
- **Medium (6 issues):** Included denial-of-service vulnerabilities in Apache HTTP Server and privilege escalation in MySQL Server.
- **Low (2 issues):** Included anonymous FTP login vulnerability and authentication confusion in OpenSSH.

- **Recommendations**

  To address these vulnerabilities:

  - **Patch systems promptly:** Apply updates for affected software such as Windows RDP, OpenSSH, Samba, etc.
  - **Enforce encryption:** Use HTTPS for web traffic and SNMPv3 for device communication.
  - **Restrict access:** Implement firewall rules and network segmentation to limit exposure.
  - **Disable unnecessary services:** Turn off unused features such as RDP or SNMPv1.
  - **Monitor activity:** Deploy intrusion detection/prevention systems (IDS/IPS) for real-time monitoring.

- **Impact for Stakeholders**

  This project highlights critical areas where immediate action is required to prevent exploitation that could lead to data breaches, reputational damage, or financial loss. The recommendations provide a roadmap for reinforcing cybersecurity defenses effectively.

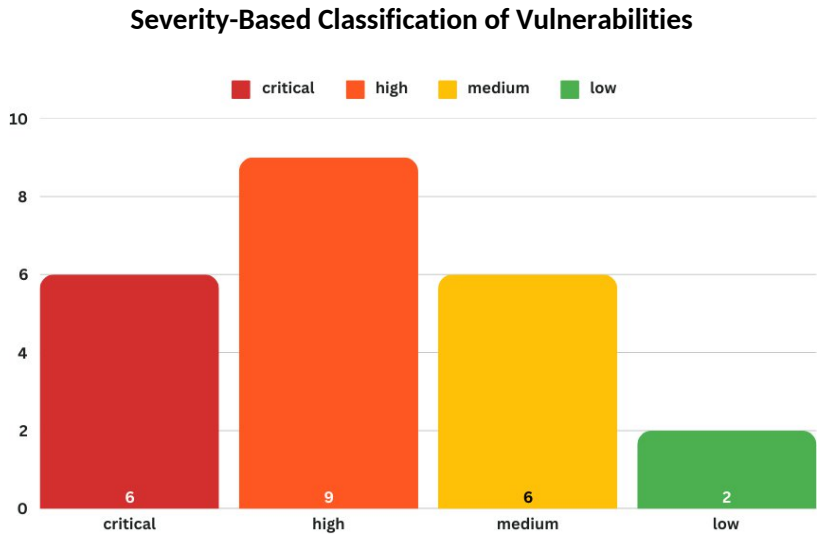

Severity-Based Classification of Vulnerabilities
critical
high
medium
low
10
8
6
4
2
0
6
9
6
2
critical
high
medium
low

Fig. 1

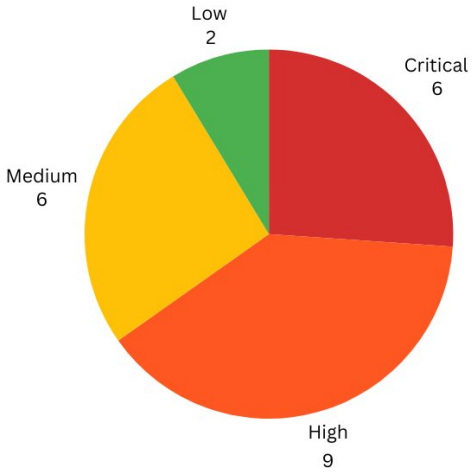

Low
2
Critical
6
Medium
6
High
9

Fig.2

Introduction

- **Background and Context:**
  The proliferation of network devices and services has created a complex landscape increasingly vulnerable to cyber threats. Organizations face continuous challenges in securing their network infrastructures against evolving attack vectors. Regular security assessments, such as penetration testing, are essential to identify vulnerabilities before malicious actors can exploit them. This project addresses this need by conducting a thorough network penetration test to identify weaknesses and provide actionable recommendations for remediation.
- **Problem Statement:**
  Many organizations lack adequate resources or expertise to proactively assess and mitigate network vulnerabilities. This gap can lead to significant security breaches and data compromises, impacting business operations, financial stability, and reputation. The primary goal of this project is to bridge this gap by providing a comprehensive evaluation of network security and offering prioritized remediation strategies.
- **Scope and Limitations:**
  The scope of this project includes black-box penetration testing of network services across a specified range of IP addresses (listed in the full report). This approach simulates real-world attacks where the tester has no prior knowledge of the system's internal workings. The project focuses on identifying vulnerabilities that can be exploited from an external perspective.
- Limitations of this project include:
  - **Time Constraints:** The testing period was limited to September 2nd, 2025, to January 30th, 2025, which may not cover all potential attack scenarios.
  - **Scope Limitations:** Only the specified IP ranges were assessed, excluding other potentially vulnerable areas of the network.
  - **Non-Disruptive Testing:** The testing was performed in a non-evasive manner to avoid causing any disruption to services, which may limit the depth of the assessment.

- **Innovation Component:**
  This project incorporates innovative approaches by:
  - **Utilizing a combination of industry-standard tools:** integrating Nmap, Metasploit, Nessus, Wireshark, and Burp Suite for comprehensive vulnerability detection.
  - **Applying real-world attack simulations:** mimicking attacker techniques to validate and prioritize vulnerabilities.
  - **Providing detailed, actionable recommendations:** offering specific steps for remediation based on the latest security best practices.
  - **Adopting a risk-based approach:** assessing vulnerabilities based on likelihood and impact to focus on the most critical issues.

Project Objectives

- **Project Objectives**
  This project, a network penetration test, was undertaken to identify security vulnerabilities within a specified network infrastructure and to provide actionable recommendations for enhancing its security posture. The primary goal was to simulate real-world attack scenarios to uncover potential weaknesses before they could be exploited by malicious actors. This was achieved through a black-box testing approach, where the assessment team had no prior knowledge of the internal systems, mimicking the perspective of an external attacker. The project focused on identifying vulnerabilities in network services and providing prioritized steps for remediation to improve the overall security of the targeted network. Successful completion of the project would result in a detailed report outlining identified vulnerabilities, a risk assessment matrix, and a comprehensive remediation plan.
- **Clearly Defined Objectives and Goals:**
  - **Vulnerability Identification:** Conduct a thorough penetration test to identify security vulnerabilities present in the specified network infrastructure.
  - **Risk Assessment:** Evaluate and classify identified vulnerabilities based on their potential impact and likelihood of exploitation, using industry-standard frameworks such as CVSS (Common Vulnerability Scoring System).
  - **Remediation Recommendations:** Provide actionable and prioritized recommendations for mitigating identified vulnerabilities to improve the overall security posture.
  - **Security Posture Improvement:** Enhance the network security by addressing identified weaknesses, thereby reducing the potential for successful cyberattacks.
- **Expected Outcomes and Deliverables:**
  - **Comprehensive Vulnerability Report:** A detailed report outlining all identified vulnerabilities, including their descriptions, potential impact, severity ratings, and affected systems.

- **Risk Assessment Matrix:** A risk assessment matrix that categorizes vulnerabilities based on their severity (Critical, High, Medium, Low) and provides a clear understanding of the associated risks.
- **Remediation Plan:** A prioritized remediation plan with specific steps and recommendations for addressing each vulnerability, including patching, configuration changes, and security enhancements.
- **Executive Summary:** A concise overview of the project findings, objectives, and recommendations, tailored for executive-level stakeholders.
- **Knowledge Transfer:** Documentation of the methodologies, tools, and techniques used during the penetration test to enable internal teams to conduct future assessments.
- **Improved Security Awareness:** Increased awareness of security risks and best practices among stakeholders and IT staff through the presentation of findings and recommendations.

Methodology

- **Methodology and Results**
  This section details the methods, technologies, and tools used to conduct the network penetration test, the approach to data collection, the project architecture, and the final project working screenshots. It also provides a link to the project's GitHub repository.
- **Methods/Technology Used:**
  The penetration test was performed using a black-box approach, simulating an external attacker with no prior knowledge of the internal network infrastructure. This approach aimed to identify unknown weaknesses and vulnerabilities from an outsider's perspective. The testing was conducted remotely and in a non-evasive manner to avoid disrupting normal operations. Key phases included planning, discovery, attack, and reporting.
  The following technologies were leveraged:
  - **Network Scanning:** TCP/IP protocol analysis, port scanning, and service enumeration.
  - **Vulnerability Assessment:** Automated vulnerability scanning and manual analysis.
  - **Exploitation:** Exploiting identified vulnerabilities to validate and assess their impact.
  - **Reporting:** Comprehensive documentation of findings, risks, and remediation strategies.
- **Tools/Software Used:**
  The following tools were used during the penetration test:
  - **Nmap:** Used for network discovery and port scanning to identify active hosts and services.
  - **Metasploit:** A framework for developing and executing exploit code against identified vulnerabilities.
  - **Nessus:** A vulnerability scanner used to identify security flaws and misconfigurations.
  - **Wireshark:** A network protocol analyzer used to capture and analyze network traffic.

- **Burp Suite:** A web application security testing tool used to identify web-based vulnerabilities.

- **Data Collection Approach:**
  The data collection approach involved several steps:
  - **Information Gathering:** Identifying active IPs and open ports using Nmap.
  - **Scanning & Enumeration:** Scanning for vulnerabilities in network services using Nessus and manual techniques.
  - **Exploitation:** Attempting to exploit known vulnerabilities using Metasploit and custom scripts.
  - **Post-Exploitation:** Verifying the impact of vulnerabilities and gathering additional information where applicable.
  - **Documentation:** Documenting all findings, including affected systems, vulnerability details, and potential impact.
- **Project Architecture:**
  The project architecture can be described as follows:
  - **Target Network:** The network infrastructure consisting of the specified range of IP addresses.
  - **Attacker System:** A remote host provisioned specifically for the penetration test.
  - **Scanning and Exploitation Tools:** Nmap, Nessus, Metasploit, Wireshark, and Burp Suite installed on the attacker system.

Network Pentesting Report

- **Vulnerability Summary**
  This table summarizes the findings from the provided document, categorizing them by severity and listing the specific vulnerability names.

Vulnerability Severity	Number of Occurrences	Vulnerability Name
Critical	6	Login Page Capturing - Sensitive Data Exposure via HTTP
		BlueKeep vulnerability (CVE-2019-0708)
		OpenSSH ssh-agent PKCS#11 Insufficient Search Path Vulnerability
		Samba Remote Code Execution Vulnerability
		SNMPv1 Vulnerabilities (MikroTik)
		Apache HTTP Server mod_auth_digest Uninitialized Memory Vulnerability
High	9	Visibility of Admin login page
		Anonymous File Upload with Public Exposure Vulnerability
		VSFTPD 3.0.3 Denial of Service Vulnerability
		Jetty HTTP/2 Connection Leak Leading to Denial of Service
		Exim Invalid Free Vulnerability in PAM Authentication

Vulnerability Severity	Number of Occurrences	Vulnerability Name
		Privilege Escalation in OpenSSH Due to Supplemental Group Initialization Flaw
		Privilege Escalation in OpenSSH Due to Unprotected Unix Domain Socket Forwarding
		Cross-Site Scripting (XSS) Vulnerability in Blog Comment System
Medium	6	Visibility of Versions in the webpage
		Denial of Service Vulnerability in Apache HTTP Server via Partial HTTP Requests
		Denial of Service Vulnerability in Apache HTTP Server Due to Improper Locking
		Privilege Escalation Vulnerability in MySQL Server's Merge Functionality
		SMTP Smuggling Vulnerability in Exim Mail Server
		Command Injection Vulnerability in OpenSSH Prior to Version 9.6
Low	2	Potential Authentication Confusion in OpenSSH Prior to Version 8.9
		Anonymous FTP Login Vulnerability

Results

a. BlueKeep Vulnerability

BlueKeep is a **critical** remote code execution (RCE) vulnerability in the **Remote Desktop Protocol (RDP) service** on older Microsoft Windows versions. It allows an **unauthenticated** attacker to execute arbitrary code on a vulnerable system **without user interaction** by sending specially crafted RDP requests.
Since this vulnerability is **wormable**, it can be used to **self-propagate** across networks, similar to **WannaCry** ransomware, making it a significant threat.

Impact- Critical (CVSS Score: 9.8)

Affected Versions

- Windows XP
- Windows 7
- Windows Server 2003
- Windows Server 2008 / 2008 R2

CVE-ID- CVE-2019-0708 – https://nvd.nist.gov/vuln/detail/CVE-2019-0708

Technical Impact

- **Remote Code Execution (RCE):** Attackers can run arbitrary code on vulnerable systems.
- **Wormable Propagation:** Can spread laterally to unpatched machines without user interaction.
- **Privilege Escalation:** Exploiting BlueKeep may grant SYSTEM-level access.
- **Data Exfiltration:** Attackers can steal sensitive data from compromised systems.
- **Service Disruption:** May cause system crashes, leading to denial-of-service (DoS) conditions.

Mitigation

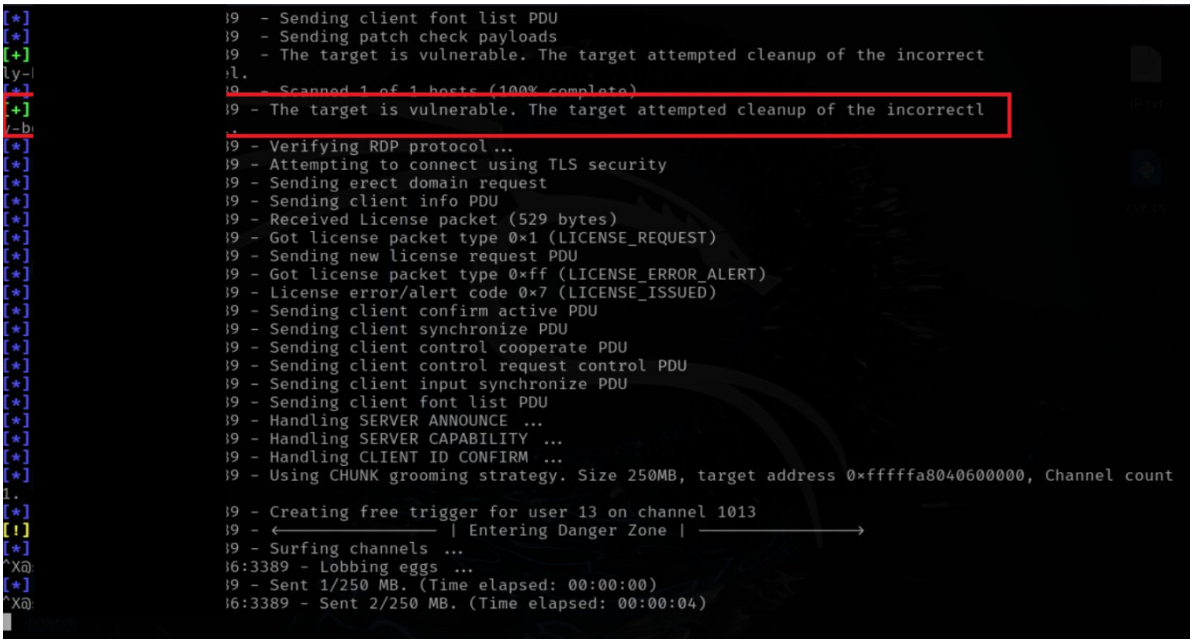
- Apply Security Patches: Install Microsoft's official patch (KB4499175) to fix the vulnerability.
- Disable RDP if Not Needed: If Remote Desktop Protocol is unnecessary, disable it.

- Enable Network-Level Authentication (NLA): Restricts RDP access to authenticated users.
- Use Firewalls: Block TCP port 3389 (RDP) from untrusted sources.
- Deploy RDP Gateways: Use secure RDP gateways instead of exposing RDP directly.
- Monitor for Exploitation Attempts: Check logs for unusual RDP connections.

Reference

- Microsoft Advisory & Patch: https://support.microsoft.com/en-us/help/4499175
- NVD Report: https://nvd.nist.gov/vuln/detail/CVE-2019-0708
- CVE Database: https://cve.mitre.org
- SANS ISC BlueKeep Analysis: https://isc.sans.edu
- OWASP RDP Security Best Practices: https://owasp.org

Proof of concept



b. Samba Remote Code Execution

CVE-2017-7494 is a **critical** remote code execution (RCE) vulnerability affecting **Samba versions 3.5.0 to 4.6.3, 4.5.9, and 4.4.13**.
A **malicious client** can exploit this flaw by **uploading a shared library** to a **writable Samba share** and then **forcing the server to load and execute it**.
This allows the attacker to **execute arbitrary code** on the system, potentially leading to full system compromise.

Impact:- **Critical** (CVSS Score: 9.8)

Affected Versions

- Samba 3.5.0 to 4.6.3
- Samba 4.5.0 to 4.5.9
- Samba 4.4.0 to 4.4.13

CVE-ID- **CVE-2017-7494** – https://nvd.nist.gov/vuln/detail/CVE-2017-7494

Technical Impact-

- **Remote Code Execution:** Attackers can execute arbitrary code on the Samba server.
- **Full System Compromise:** Successful exploitation may grant root-level access.
- **Lateral Movement:** Attackers can use the compromised system to attack other network resources.
- **Malware Deployment:** The vulnerability can be used to install backdoors or ransomware.

Mitigation

- **Apply Security Patches:** Upgrade to **Samba 4.6.4, 4.5.10, or 4.4.14** or later versions.
- **Disable Unnecessary Writable Shares:** Restrict write access to Samba shares when not needed.
- **Set 'nt pipe support = no' in smb.conf:** This mitigates the vulnerability but may impact some features.

- **Enforce Strong Access Controls:** Limit access to Samba shares to trusted users and groups.
- **Monitor Samba Logs:** Check for unauthorized file uploads or suspicious activity in Samba logs.

Reference

- **Samba Security Advisory:** https://www.samba.org/samba/security/CVE-2017-7494.html
- **NVD Report:** https://nvd.nist.gov/vuln/detail/CVE-2017-7494
- **CVE Database:** https://cve.mitre.org
- **OWASP File Sharing Security Best Practices:** https://owasp.org
- **SANS ISC Samba Vulnerability Analysis:** https://isc.sans.edu

c. Login Page Capturing – Sensitive Data Exposure via HTTP

If login credentials are transmitted over **HTTP instead of HTTPS**, they can be intercepted using packet-sniffing tools like **Wireshark**, leading to **sensitive data exposure**. Without encryption, usernames and passwords are sent in **plaintext**, making them vulnerable to attackers on the network.

Impact- 9.0 (Critical)

CVE-ID- No specific CVE-ID (General security misconfiguration)

Technical Impact-

- **Sensitive Data Exposure:** Attackers can capture login credentials directly from network traffic.
- **Unauthorized Access:** Stolen credentials can be used to **compromise user accounts**.
- **Man-in-the-Middle (MITM) Attacks:** Attackers can intercept, modify, or replay login requests.
- **Compliance Violations:** Many security standards (e.g., PCI-DSS, GDPR) **prohibit** plaintext credential transmission.

Mitigation

- **Enforce HTTPS:** Ensure the login page and all authentication requests use **TLS encryption (HTTPS)**.
- **Use Secure Cookies:** Enable Secure and HttpOnly flags for cookies to prevent interception.

- **HSTS (HTTP Strict Transport Security):** Implement HSTS headers to force browsers to use HTTPS.
- **Disable HTTP for Login Pages:** Redirect all HTTP requests to HTTPS automatically.
- **Monitor Network Traffic:** Regularly check for **unencrypted credential transmissions** using network monitoring tools.

Reference

- **OWASP Secure Login Guidelines:** https://owasp.org
- **NIST Security Standards:** https://csrc.nist.gov
- **PCI DSS Compliance for Secure Transmission:** https://www.pcisecuritystandards.org
- **SANS Network Security Analysis:** https://isc.sans.edu

Proof of concept

d. SNMPv1 Vulnerabilities (MikroTik)

SNMPv1 **transmits data in plaintext**, exposing **sensitive information**, including **community strings**.
MikroTik devices using **SNMPv1** are highly vulnerable to **interception and unauthorized access**, especially if the **default "public" community string** is left unchanged.

Impact- Critical (9.8)

CVE-ID- **No specific CVE-ID** (General SNMPv1 security weakness)

Technical Impact

- **Sensitive Data Exposure:** Attackers can intercept **network configuration details** and credentials.
- **Unauthorized Access:** If an attacker gains read or write access, they can **extract or modify device settings**.
- **Network Reconnaissance:** Exposed SNMP data helps attackers map network structures.
- **Configuration Manipulation:** If write access is enabled, attackers may **alter routing, firewall, or NAT rules**.

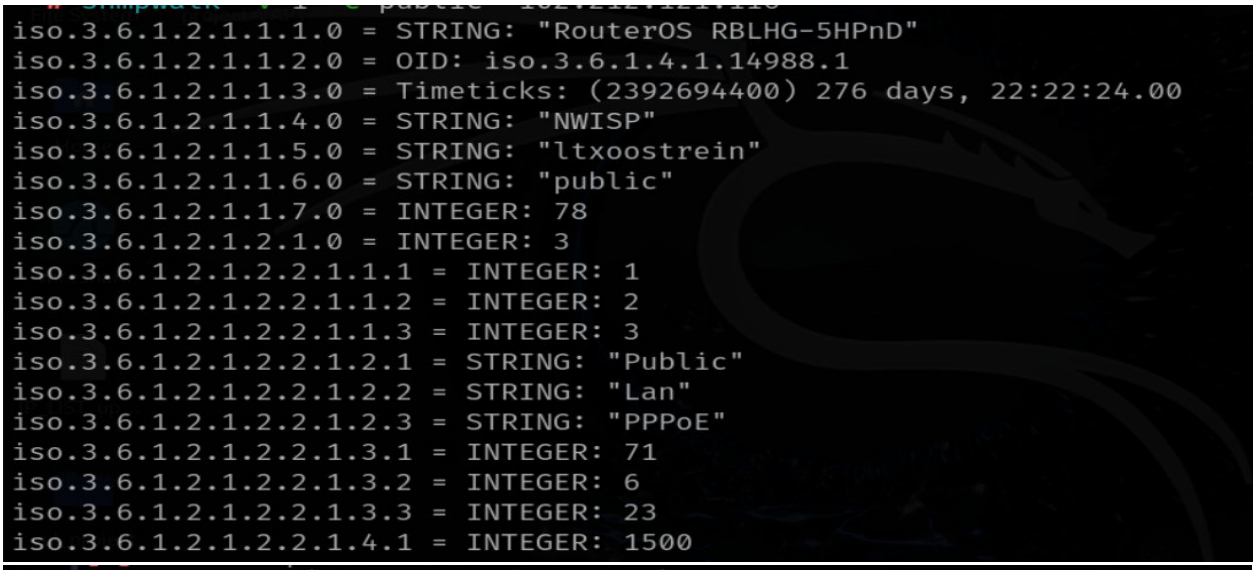
Mitigation

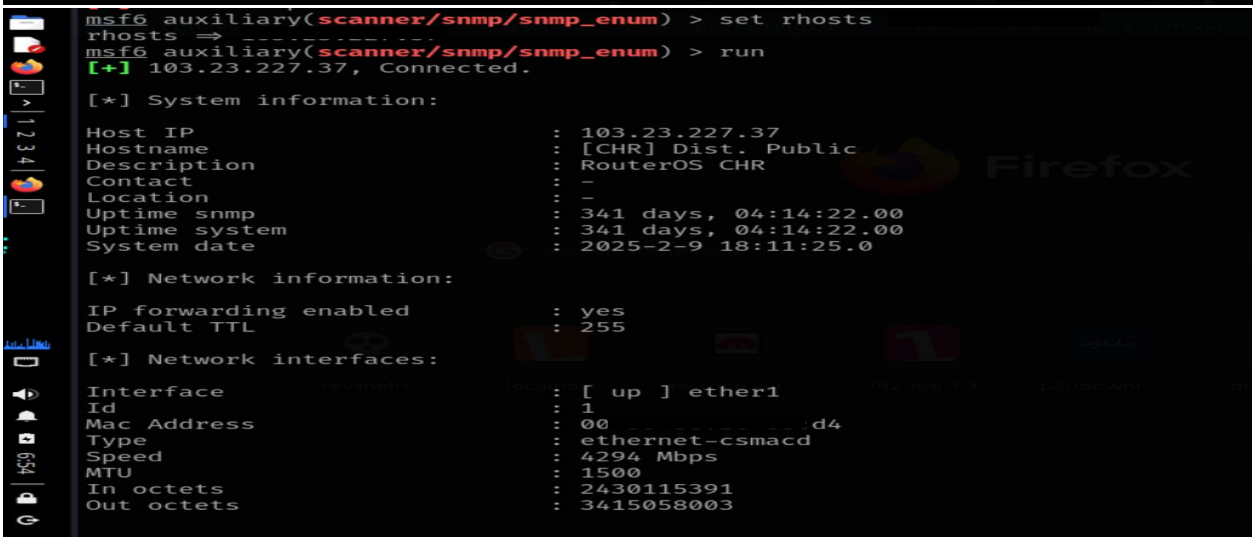
- **Disable SNMPv1:** If possible, **disable SNMPv1** and use **SNMPv3**, which supports encryption and authentication.
- **Change Default Community Strings:** Avoid using "public" and "private"; set **complex, unique strings**.
- **Restrict SNMP Access:** Configure firewalls to **limit SNMP access** to trusted IP addresses.
- **Enable Authentication & Encryption:** Use **SNMPv3 with SHA/AES** for secure communication.
- **Monitor SNMP Logs:** Regularly inspect logs for **suspicious SNMP queries**.
- **Upgrade Firmware:** Ensure MikroTik RouterOS is **up-to-date** with security patches.

Reference

- **MikroTik SNMP Security Guide:** https://wiki.mikrotik.com
- **OWASP SNMP Security Best Practices:** https://owasp.org
- **NIST SNMP Security Guidelines:** https://csrc.nist.gov
- **SANS Network Security Recommendations:** https://isc.sans.edu

Proof of concept





e. Apache HTTP Server mod_auth_digest Memory Leak

CVE-2017-9788 is a **critical** vulnerability in the **Apache HTTP Server's mod_auth_digest module**, affecting **versions prior to 2.2.34 and 2.4.x before 2.4.27**.

The vulnerability occurs due to **improper handling of [Proxy-]Authorization headers of type 'Digest'**, where value placeholders are **not correctly initialized or reset** between successive key-value assignments.

An **attacker can exploit this flaw** by sending an initial key **without an '=' assignment**, causing the server to **reuse stale memory values** from previous requests. This can result in:

- **Information Disclosure** – Leakage of sensitive memory data, which may contain **authentication credentials or other confidential information**.
- **Denial of Service (DoS)** – In some cases, exploitation can **trigger segmentation faults**, crashing the server.

Impact- Critical (9.1)

Affected Versions

- Apache HTTP Server versions before 2.2.34
- Apache HTTP Server 2.4.x before 2.4.27

CVE-ID- **CVE-2017-9788 –** https://nvd.nist.gov/vuln/detail/CVE-2017-9788

Technical Impact

- **Sensitive Data Exposure:** Attackers may retrieve residual memory content from previous requests.
- **Denial of Service:** Crafted requests can trigger memory errors, causing a server crash.
- **Authentication Bypass (Limited Scope):** If sensitive credentials leak, attackers may gain unauthorized access.

Mitigation

- Apply Security Patches: Upgrade to Apache HTTP Server 2.2.34 or 2.4.27+.
- Disable mod_auth_digest if Not Needed: If digest authentication is not required, disable the module.
- Restrict Authorization Headers in Proxies: Ensure proxies handling authentication headers sanitize input properly.
- Monitor Server Logs: Check for abnormal authentication requests that could indicate exploitation attempts.

Reference

- **Apache Security Advisory:** https://httpd.apache.org/security/vulnerabilities_24.html
- **NVD Report:** https://nvd.nist.gov/vuln/detail/CVE-2017-9788
- **CVE Database:** https://cve.mitre.org
- **OWASP Web Security Best Practices:** https://owasp.org
- **SANS HTTP Security Recommendations:** https://isc.sans.edu

f. OpenSSH ssh-agent PKCS#11 Insecure Library Search Path

CVE-2023-38408 is a **critical** vulnerability in **OpenSSH versions prior to 9.3p2**.

The issue exists in **ssh-agent's handling of the PKCS#11 feature**, where it **uses an insufficiently trustworthy search path** for loading shared libraries. If **agent forwarding** is enabled and the ssh-agent is forwarded to an **attacker-controlled system**, an attacker can exploit this flaw to achieve **remote code execution (RCE)** by loading malicious PKCS#11 modules from unsafe locations like **/usr/lib**.

Impact- **Critical** (9.8)

Affected Versions

- OpenSSH versions before 9.3p2

CVE-ID- CVE-2023-38408 – https://nvd.nist.gov/vuln/detail/CVE-2023-38408

Technical Impact

- **Remote Code Execution (RCE):** Attackers can execute arbitrary code by injecting a malicious PKCS#11 module.
- **Privilege Escalation:** If ssh-agent runs with higher privileges, attackers may gain elevated access.
- **Supply Chain Risk:** Systems relying on ssh-agent forwarding for authentication may be compromised.

Mitigation

- **Upgrade OpenSSH:** Patch to **OpenSSH 9.3p2 or later** to fix the vulnerability.
- **Disable PKCS#11 in ssh-agent (if not needed):** Avoid using PKCS#11 modules unless absolutely necessary.
- **Restrict Agent Forwarding:** Only enable agent forwarding to trusted hosts.
- **Enforce Trusted Paths:** Use PKCS11Provider to explicitly define safe paths for module loading.
- **Monitor SSH Activity:** Regularly audit logs for unexpected ssh-agent behavior.

Reference

- **NVD Report:** https://nvd.nist.gov/vuln/detail/CVE-2023-38408
- **OpenSSH Security Advisory:** https://www.openssh.com/security.html
- **OWASP SSH Security Best Practices:** https://owasp.org
- **SANS SSH Security Guidelines:** https://isc.sans.edu


High

a. Jetty HTTP/2 Connection Leak Resulting in Denial of Service

CVE-2024-22201 is a **security vulnerability in the Jetty web server**, affecting versions **prior to 9.4.54, 10.0.20, 11.0.20, and 12.0.6**.
The issue occurs when an **HTTP/2 SSL connection experiences TCP congestion and times out**. When this happens:

- The **HTTP/2 session is marked as closed**, and a **GOAWAY frame** is queued for transmission.
- Due to **TCP congestion**, the **GOAWAY frame is never sent**, leaving the **connection in an idle state**.
- This results in a **connection leak**, where system resources remain consumed indefinitely.

Attack Scenario

An attacker can exploit this flaw by:

- Establishing **multiple HTTP/2 SSL connections**.
- Forcing these connections into a **TCP congestion state**.
- Causing **resource exhaustion** by leaving a large number of **stale connections**.
- Eventually, **legitimate clients are denied access** as the server runs out of **available file descriptors**.

Impact- **High** (7.5)

Affected Versions

- Jetty versions before 9.4.54, 10.0.20, 11.0.20, and 12.0.6

CVE-ID- **CVE-2024-22201** – https://nvd.nist.gov/vuln/detail/CVE-2024-22201

Technical Impact

- **Denial of Service (DoS):** Attackers can consume all available file descriptors, preventing new connections.

- **Resource Exhaustion:** The server may become unresponsive due to the accumulation of idle connections.
- **Operational Disruption:** Critical web services running on Jetty could be rendered unavailable.

Mitigation

- **Upgrade Jetty:** Apply patches to **Jetty 9.4.54, 10.0.20, 11.0.20, or 12.0.6**.
- **Monitor TCP Congestion:** Implement monitoring tools to detect abnormal TCP congestion patterns.
- **Set Connection Limits:** Configure Jetty to **limit idle connections** and enforce strict timeouts.
- **Use Reverse Proxies:** Deploy **reverse proxies (e.g., Nginx, Apache)** to mitigate direct attacks on Jetty.
- **Apply Rate Limiting:** Implement rate limiting for HTTP/2 connections to reduce the impact of an attack.

Reference

- **NVD Report:** https://nvd.nist.gov/vuln/detail/CVE-2024-22201
- **Jetty Security Advisory:** https://www.eclipse.org/jetty/security
- **OWASP Denial of Service Prevention Guide:** https://owasp.org
- **SANS Network Security Best Practices:** https://isc.sans.edu

b. Privilege Escalation in OpenSSH Caused by Improper Supplemental Group Initialization

CVE-2021-41617 is a privilege escalation vulnerability in OpenSSH, affecting versions 6.2 through 8.x (prior to 8.8).

The flaw occurs when using certain non-default configurations involving:

- AuthorizedKeysCommand
- AuthorizedPrincipalsCommand

When these directives are used, OpenSSH may improperly initialize supplemental groups, causing helper programs to execute with group privileges associated with the sshd process rather than those of the intended user.

This misconfiguration allows unauthorized users to escalate privileges by leveraging the elevated group permissions of sshd.

Impact- **High** (7.0)

Affected Versions

- OpenSSH 6.2 through 8.x (prior to 8.8)

CVE-ID- **CVE-2021-41617** – https://nvd.nist.gov/vuln/detail/CVE-2021-41617

Technical Impact

- **Privilege Escalation:** Attackers can execute commands with unintended group privileges.
- **Unauthorized Access:** Malicious users may gain higher access than intended.
- **System Compromise:** Attackers could leverage this flaw for further exploitation.

Mitigation

- **Upgrade OpenSSH:** Apply patches to **OpenSSH 8.8 or later** to fix the issue.
- **Audit SSH Configuration:** Ensure that AuthorizedKeysCommand and AuthorizedPrincipalsCommand are used securely.
- **Use Sandboxing:** Configure OpenSSH to run commands in a restricted environment.
- **Restrict User Privileges:** Limit access to sensitive files and directories to mitigate privilege escalation risks.
- **Monitor SSH Activity:** Log and analyze SSH command execution for anomalies.

Reference

- **NVD Report:** https://nvd.nist.gov/vuln/detail/CVE-2021-41617
- **OpenSSH Security Advisory:** https://www.openssh.com/security.html
- **OWASP SSH Security Best Practices:** https://owasp.org
- **SANS Secure Configuration Guide:** https://isc.sans.edu

c. Privilege Escalation in OpenSSH Caused by Unprotected Unix Domain Socket Forwarding

CVE-2016-10010 is a **privilege escalation vulnerability in OpenSSH**, affecting versions **prior to 7.4**.
The flaw is present in configurations where **privilege separation is disabled**. In such cases:

- The **sshd process creates forwarded Unix-domain sockets with root privileges**.
- Due to **insufficient protections**, a local attacker can leverage these sockets to **gain unauthorized privileges**.
- The exact exploitation vectors are unspecified, but the flaw **could allow privilege escalation or unauthorized system access**.

Impact- High (7.0)

Affected Versions

- OpenSSH versions before 7.4

CVE-ID- **CVE-2016-10010** – https://nvd.nist.gov/vuln/detail/CVE-2016-10010

Technical Impact

- **Privilege Escalation:** Local users can exploit the flaw to execute commands with elevated privileges.
- **Unauthorized Access:** Attackers may abuse forwarded sockets to gain control over the system.
- **System Compromise:** Exploitation of this flaw could result in complete system takeover.

Mitigation

- **Upgrade OpenSSH:** Patch to **OpenSSH 7.4 or later** to fix the vulnerability.
- **Enable Privilege Separation:** Ensure **Privilege Separation** is enabled in the OpenSSH configuration.

- **Restrict Socket Forwarding:** Limit or disable Unix-domain socket forwarding if not required.
- **Apply Access Controls:** Implement strict permissions on SSH configurations and socket access.
- **Monitor SSH Activity:** Regularly audit SSH logs for unusual activity.

Reference

- **NVD Report:** https://nvd.nist.gov/vuln/detail/CVE-2016-10010
- **OpenSSH Security Advisory:** https://www.openssh.com/security.html
- **OWASP SSH Hardening Guide:** https://owasp.org
- **SANS Secure Configuration Guide:** https://isc.sans.edu

d. Exposure of Admin Login Page

The **admin login page** is a critical attack surface, as it serves as the primary **entry point** for accessing administrative controls of a system.
If an attacker **discovers** the admin login page, they may attempt:

- **Brute Force Attacks:** Repeated login attempts using common passwords.
- **Credential Stuffing:** Using leaked username-password combinations from previous breaches.
- **Enumeration Attacks:** Identifying valid usernames and security configurations.

Defensive Mechanisms:

- If brute-force attempts fail, the system may have **protections like rate-limiting or MFA**, making direct attacks difficult.
- However, attackers or penetration testers may explore **alternative bypass techniques**, such as:
  - **Exploiting weak session management**
  - **Abusing forgotten password flows**
  - **Finding backdoor access points**

Impact- **High** (8.0)

Affected Versions

All web applications with exposed admin login pages, especially those lacking:

- Rate limiting
- Multi-factor authentication (MFA)
- Proper access restrictions (IP whitelisting, VPN enforcement, etc.)

CVE-ID- **No specific CVE assigned**, but related authentication vulnerabilities can be found in the https://cve.mitre.org/

Technical Impact

- **Unauthorized Access:** Attackers may gain admin privileges if authentication is bypassed.
- **Data Exposure:** Admin access can lead to data breaches or modifications.
- **System Takeover:** Successful exploitation can result in complete control over the system.

Mitigation

- **Restrict Access:** Limit admin page visibility using **IP whitelisting** or VPN access.
- **Enforce Strong Authentication:** Use **multi-factor authentication (MFA)** and strong password policies.
- **Enable Rate Limiting & Lockouts:** Block repeated failed login attempts.
- **Use CAPTCHA & WAF:** Prevent automated brute-force tools from attacking login pages.
- **Rename the Admin Page:** Change default admin URLs to reduce discoverability.
- **Monitor Login Activity:** Log failed login attempts and implement alerts for unusual access patterns.

Reference

- **OWASP Authentication Guidelines:** https://owasp.org
- **SANS Secure Authentication Best Practices:** https://isc.sans.edu
- **CVE Database for Authentication Vulnerabilities:** https://cve.mitre.org

Proof of concept

e. VSFTPD 3.0.3 Denial of Service (DoS) Vulnerability

VE-2021-30047 is a **Denial of Service (DoS) vulnerability** in **VSFTPD 3.0.3**. The flaw arises due to a **limitation in the number of concurrent connections** the server can handle.

- **Attackers can exploit this limitation** by initiating **multiple simultaneous connections** to the FTP server.
- This **exhausts the available connection slots**, **preventing legitimate users from accessing the service**.
- **No authentication is required**, making it an easy target for **unauthenticated DoS attacks**.

Impact- **High** (7.5)

Affected Versions

- VSFTPD (Very Secure FTP Daemon) version 3.0.3

CVE-ID- **CVE-2021-30047** – https://nvd.nist.gov/vuln/detail/CVE-2021-30047

Technical Impact

- **Denial of Service:** The FTP server becomes unresponsive to legitimate users.
- **Resource Exhaustion:** The attack consumes server resources, potentially leading to crashes.
- **Business Disruption:** Organizations relying on VSFTPD for file transfers may face service outages.

Mitigation

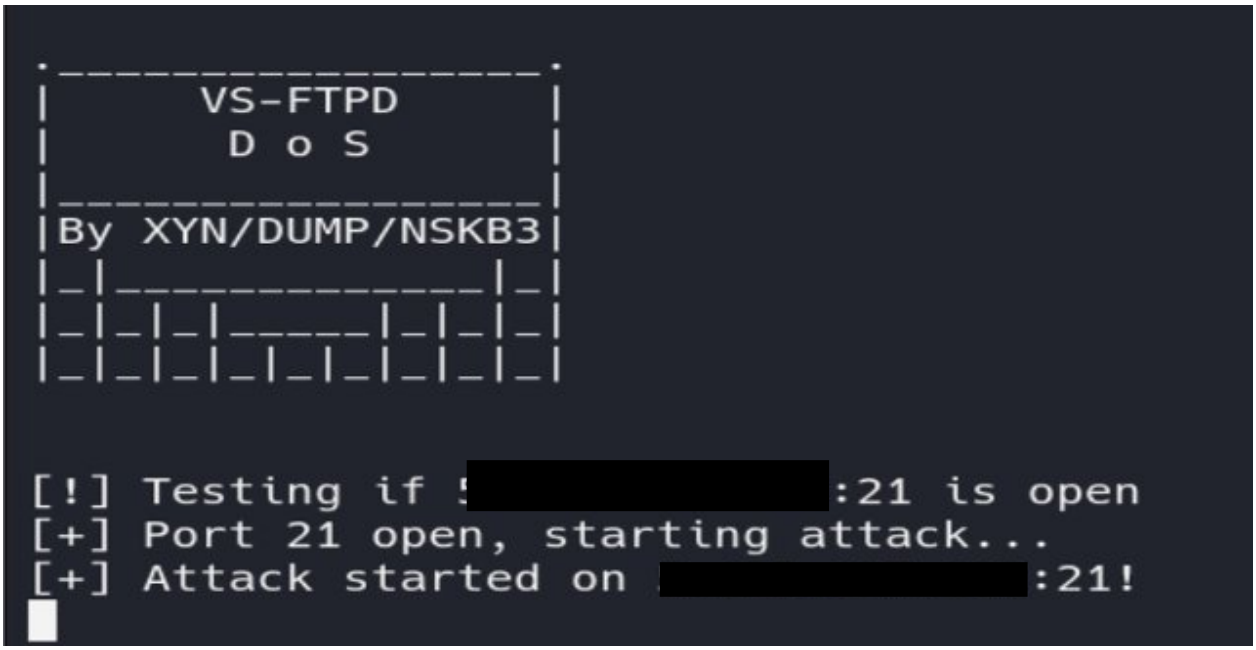
- **Upgrade VSFTPD:** Check for patches or upgrade to a secure version if available.
- **Limit Connection Rate:** Configure **connection rate limits** to restrict excessive connections from a single IP.
- **Enable IP-Based Restrictions:** Use **firewall rules** to block repeated connection attempts from the same source.
- **Implement Connection Timeouts:** Reduce the **timeout** for inactive connections to free up server resources.

- **Monitor Server Logs:** Set up **logging and alerts** for unusual connection spikes.

Reference

- **NVD Report:** https://nvd.nist.gov/vuln/detail/CVE-2021-30047
- **VSFTPD Official Repository:** https://security.appspot.com/vsftpd.html
- **OWASP FTP Security Guidelines:** https://owasp.org

Proof of concept



f. Unauthorized Anonymous File Upload Leading to Public Exposure

This vulnerability arises when a system **permits anonymous users to upload files** without proper **authentication or validation**.

- **Uploaded files are publicly accessible**, allowing **attackers to upload malicious scripts, web shells, or malware payloads**.
- This can lead to **remote code execution, privilege escalation, data breaches, and full system compromise**.
- **Commonly exploited in web applications**, misconfigured cloud storage, and public FTP servers.

Impact- : High (8.0)

Affected Versions

- Any web application or system that allows anonymous file uploads without proper authentication, validation, or access controls.

CVE-ID- **No specific CVE assigned**, but related file upload vulnerabilities can be found in the https://cve.mitre.org/

Technical Impact

- **Remote Code Execution (RCE):** Attackers can upload **web shells** (e.g., PHP, ASP, JSP scripts) to execute arbitrary commands.
- **Malware Hosting:** The system can be used to **store and distribute malicious files**, leading to reputational damage.
- **Privilege Escalation:** Exploiting misconfigurations may allow attackers to gain **higher access privileges**.
- **Defacement or Data Theft:** Sensitive files can be replaced, deleted, or stolen.

Mitigation

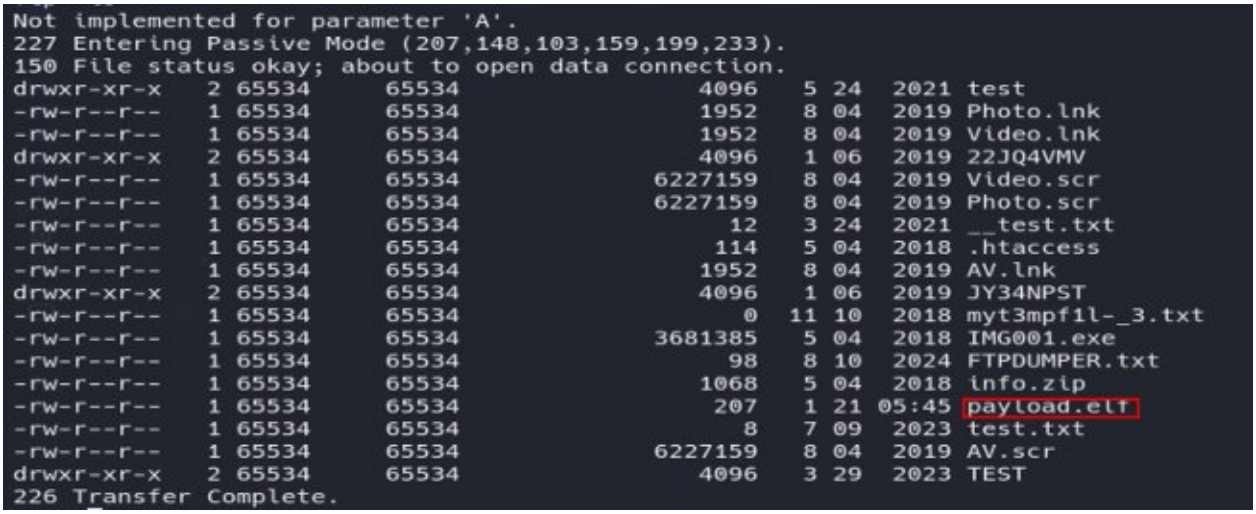
- **Restrict Anonymous Uploads:** Require authentication for file uploads.
- **Implement File Type Validation:** Allow only specific file extensions and **verify file contents (MIME type checking)**.

- **Disable Executable File Uploads:** Block scripts like .php, .asp, .exe, and .sh from being uploaded.
- **Use a Secure Upload Directory:** Store uploaded files in a **non-web-accessible** directory to prevent direct execution.
- **Apply Access Controls:** Ensure only authorized users can access uploaded files.
- **Enable Malware Scanning:** Use **antivirus scanning** on uploaded files.
- **Monitor and Log Upload Activity:** Set up **logging and alerting** for suspicious file uploads.

Reference

- **OWASP Unrestricted File Upload Guidelines:** https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- **NIST Secure File Upload Recommendations:** https://csrc.nist.gov
- **CVE Database for File Upload Vulnerabilities:** https://cve.mitre.org

Proof of concept



g. Exim PAM Authentication Invalid Free Vulnerability

CVE-2022-37451 is a **Denial of Service (DoS) vulnerability** in **Exim**, a widely used **Mail Transfer Agent (MTA)**.

- The flaw exists in the **pam_converse function** within auths/call_pam.c.
- An **invalid free operation** occurs due to improper memory management, where **store_free is not used after store_malloc**.
- **Remote attackers** can exploit this vulnerability to **crash the Exim process**, leading to **service disruption**.

Impact- **High** (7.5)

Affected Versions

- Exim versions prior to 4.96

CVE-ID- **CVE-2022-37451** – https://nvd.nist.gov/vuln/detail/CVE-2022-37451

Technical Impact

- **Denial of Service (DoS):** The Exim mail server crashes, preventing legitimate email delivery.
- **Memory Corruption:** The invalid free operation may lead to **unexpected behavior or instability**.
- **Potential for Further Exploitation:** While this flaw primarily causes DoS, improper memory handling could **increase the attack surface** for further exploits.

Mitigation

- **Upgrade Exim:** Patch to **version 4.96 or later**, where this vulnerability has been fixed.
- **Restrict Untrusted Access:** Limit **remote access to Exim** using **firewall rules and access controls**.
- **Monitor Logs for Crashes:** Set up **logging and alerts** to detect unusual server behavior.

- **Apply Memory Protections:** Enable **Address Space Layout Randomization (ASLR) and other memory protection mechanisms** to mitigate exploitation risks.

Reference

- **NVD Report:** https://nvd.nist.gov/vuln/detail/CVE-2022-37451
- **Exim Official Security Advisories:** https://www.exim.org/security
- **OWASP DoS Prevention Guidelines:** https://owasp.org

h. OpenSSH scp Command Injection Vulnerability

CVE-2020-15778 is a **command injection vulnerability** in OpenSSH's **scp (Secure Copy Protocol) utility**.

- The flaw exists in the **tor remote function** within the scp.c file.
- Due to **improper input validation**, an attacker can craft a **malicious file name containing backtick (`) characters**.
- When the file is copied using **scp**, the shell interprets these **backticks as commands**, leading to **arbitrary command execution**.
- The **commands run with the same privileges** as the user executing the scp command.

Impact- **High** (7.8)

Affected Versions

- OpenSSH versions up to 8.3p1

CVE-ID- **CVE-2020-15778** – https://nvd.nist.gov/vuln/detail/CVE-2020-15778

Technical Impact

- **Remote Code Execution (RCE):** Attackers can execute arbitrary commands on the **remote system**.
- **Privilege Escalation:** If the **scp command is run as a privileged user**, attackers may **gain elevated access**.
- **System Compromise:** Malicious scripts can be **uploaded and executed remotely**, leading to **persistent access**.

Mitigation

- **Upgrade OpenSSH:** Patch to **version 8.4p1 or later**, where this vulnerability has been fixed.
- **Use SFTP Instead:** Replace scp with **SFTP (Secure File Transfer Protocol)**, which is more secure.
- **Manually Verify File Names:** Avoid copying files with **untrusted or unusual names**.
- **Disable SCP Support:** If not required, disable scp and enforce alternative **secure file transfer methods**.
- **Use Restricted Shells:** Apply rbash (restricted Bash) or similar **to limit command execution** on the server.

Reference

- **NVD Report:** https://nvd.nist.gov/vuln/detail/CVE-2020-15778
- **OpenSSH Security Advisory:** https://www.openssh.com/security.html
- **SCP Security Considerations – OWASP:** https://owasp.org

i. Blog Comment System Cross-Site Scripting (XSS) Vulnerability

This **Cross-Site Scripting (XSS) vulnerability** occurs in the **blog comment system** of a web application that **fails to sanitize user input properly**.

- Attackers can inject **malicious JavaScript code** into the comment section.
- When a victim views the affected page, the script **executes within their browser**.
- The **payload <image/src/onerror=prompt(8)>** demonstrates a **script execution attempt via the onerror event handler**.
- A successful attack can **trigger an alert box, steal session cookies, or redirect users to phishing sites**.

Impact- **High** (7.5)

Affected Versions

- Web applications with unsanitized user input in blog comments

CVE-ID- (No assigned CVE, custom vulnerability assessment)

Technical Impact

- **Session Hijacking:** Attackers can steal **authentication cookies** and impersonate users.
- **Credential Theft:** Keyloggers or phishing scripts can capture **usernames and passwords**.
- **Malicious Redirects:** Users may be redirected to **fake login pages** for credential harvesting.
- **Browser Exploitation:** JavaScript-based exploits may be used to **execute additional attacks**.

Mitigation

- **Input Sanitization:** Filter and encode user input to **remove or neutralize malicious scripts**.
- **Use Content Security Policy (CSP):** Implement CSP headers to **restrict script execution sources**.
- **HTTPOnly & Secure Cookies:** Protect **session cookies** from being accessed via JavaScript.
- **Implement Web Application Firewall (WAF):** Block **known attack patterns** in incoming requests.
- **Use Modern Frameworks:** Adopt frameworks like **React, Angular, or Vue.js**, which handle XSS protection by default.

Reference

- **OWASP XSS Prevention Cheat Sheet:** https://owasp.org/www-community/attacks/xss
- **CVE Database:** https://cve.mitre.org
- **NVD Security Guidelines:** https://nvd.nist.gov

Proof of concept


Blog Comments
Please post your comments for the blog
Submit
<image/src/onerror=prompt(8)>


Medium

a. Exposure of Software Version Information on Webpage

Version disclosure occurs when **web applications, servers, or software components** reveal their version numbers through:

- **HTTP response headers** (e.g., Server: Apache/2.4.49)
- **Page source code** (e.g., metadata comments like <!-- Powered by WordPress 5.8 -->)
- **Error messages** displaying software details (e.g., PHP 7.4.3 on Ubuntu)

Attackers can **leverage this information** to:

- Identify outdated versions with **known vulnerabilities**
- Conduct **targeted exploitation** based on publicly available exploits
- Plan **zero-day attacks** on systems with exposed version details

Impact- **Medium** (5.3)

Affected Versions

- Web applications, servers, or software components that expose version details in HTTP headers, page source, or error messages

OWASP- A06:2021 – Vulnerable and Outdated Components

Technical Impact

- **Fingerprinting:** Attackers can determine software versions to **find matching exploits**.
- **Exploit Readiness:** Public vulnerability databases (e.g., **ExploitDB, CVE databases**) provide attackers with **pre-built exploits**.
- **Brute Force & Automated Attacks:** Tools like **Nikto, Wappalyzer, and WhatWeb** automate **version detection** for exploitation.

Mitigation

- **Disable Version Disclosure:** Remove or obfuscate version information in **HTTP headers and error messages**.
    - For Apache: ServerTokens Prod and ServerSignature Off in httpd.conf
    - For Nginx: server_tokens off; in **nginx.conf**

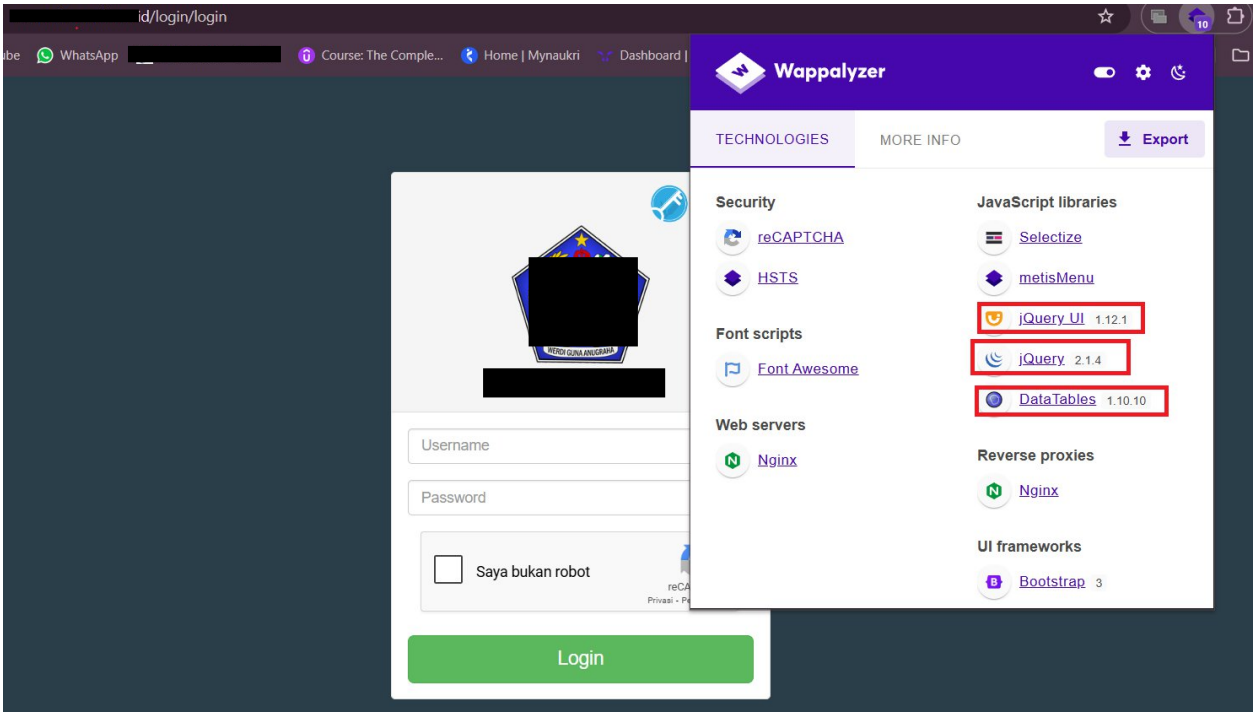
- **Use a Web Application Firewall (WAF):** WAFs can **mask or block** sensitive version details.

- **Customize Error Pages:** Prevent error messages from **leaking version details** (e.g., generic "Internal Server Error" pages).
- **Regular Patching:** Keep **all software components updated** to mitigate risks, even if version details are leaked.

Reference

- **OWASP Testing for Information Leakage:** https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Information_Gathering/01-Testing_for_Information_Leakage
- **CVE Database:** https://cve.mitre.org
- **NIST NVD:** https://nvd.nist.gov

Proof of concept



b. Privilege Escalation Vulnerability in MySQL Server Due to Improper Merge Functionality Handling

Impact- **Medium** (5.5)

Affected Versions

- MySQL 5.6.41 and prior
- MySQL 5.7.23 and prior
- MySQL 8.0.12 and prior

CVE-ID- **CVE-2018-3247** – https://nvd.nist.gov/vuln/detail/CVE-2018-3247

Technical Impact

- **Data Exposure** – Attackers can **access or modify** sensitive information stored in MySQL.
- **System Takeover** – An attacker with a foothold in the system could escalate to **full control** of the database.
- **Lateral Movement** – Compromised credentials can be used to **pivot** within the network.

Mitigation

- **Update MySQL to a patched version** (MySQL **5.6.42+, 5.7.24+, 8.0.13+**)
- **Restrict Privileged Access** – Limit **administrative roles** to trusted users only.
- **Enable MySQL Security Best Practices:**
  - **Disable unnecessary features** that could be exploited
  - Restrict network access to MySQL using firewall rules.
  - Use strong authentication mechanisms like MySQL native password hashing.
- **Monitor Database Logs** – Detect suspicious privilege escalation attempts.

Reference

- **Oracle Security Advisory:** https://www.oracle.com/security-alerts/
- **CVE Database:** https://cve.mitre.org
- **NIST NVD:** https://nvd.nist.gov

c. SMTP Smuggling Vulnerability in Exim Mail Server

This vulnerability is due to **improper handling of SMTP commands** in certain **PIPELINING** and **CHUNKING** configurations in Exim mail servers. Specifically, Exim's support for the sequence <LF>.<CR><LF> differs from other popular mail servers, causing **inconsistent email message processing**.

Attackers can exploit this flaw to perform **SMTP smuggling**, which allows them to:

- **Inject maliciously crafted email messages** with **spoofed MAIL FROM addresses**.
- **Bypass Sender Policy Framework (SPF)** checks to send unauthorized emails.
- **Facilitate phishing, spam, or email-based social engineering attacks** by making emails appear legitimate.

Impact- **Medium** (5.3)

Affected Versions

- Exim mail server versions prior to 4.97.1

CVE-ID- **CVE-2023-51766** – https://nvd.nist.gov/vuln/detail/CVE-2023-51766

Technical Impact

- **Email Spoofing** – Attackers can impersonate legitimate domains and send deceptive emails.
- **Security Policy Bypass** – SPF, DKIM, and DMARC verification can be **circumvented**.
- **Email-Based Attacks** – Malicious actors may use this vulnerability for **phishing** or **spreading malware**.

Mitigation

- **Update Exim to version 4.97.1 or later** to patch the vulnerability.
- **Disable PIPELINING and CHUNKING if not required**, or ensure they are configured securely.
- **Implement SPF, DKIM, and DMARC properly** to **reduce** the impact of spoofed emails.
- **Monitor email logs** for unusual SMTP sequences and unauthorized sender activity.
- **Use email security gateways** to filter out **potentially malicious emails** before reaching end users.

Reference

- **Exim Official Security Advisories:** https://www.exim.org/security/
- **CVE Database:** https://cve.mitre.org
- **NIST NVD:** https://nvd.nist.gov
- **Email Security Best Practices (OWASP):** https://owasp.org/www-project-email-security/

d. Command Injection Vulnerability in OpenSSH Versions Prior to 9.6

This vulnerability occurs when **shell metacharacters** are present in a **username or hostname** referenced by an **expansion token** in OpenSSH configurations.
Attackers can exploit this flaw by **injecting shell commands** in scenarios such as:

- **Untrusted Git repositories** containing submodules with **maliciously crafted usernames or hostnames**.
- **Automated scripts or SSH configurations** that process **untrusted inputs** without sanitization.

When OpenSSH interprets these **malicious inputs**, it can lead to **OS command injection**, potentially allowing an attacker to execute **arbitrary commands** with the privileges of the SSH user.

Impact- **Medium** (6.5)

Affected Versions

- OpenSSH versions prior to 9.6

CVE-ID- **CVE-2023-51385** – https://nvd.nist.gov/vuln/detail/CVE-2023-51385

Technical Impact

- **Remote Code Execution (RCE)** – An attacker may execute unauthorized system commands.
- **Privilege Escalation** – If exploited in a privileged session, it could lead to system compromise.
- **Supply Chain Attacks** – Malicious Git submodules can be used to inject commands upon repository cloning.

Mitigation

- **Upgrade OpenSSH to version 9.6 or later** to apply the official patch.
- **Validate and sanitize user-provided inputs** in SSH configurations and scripts.
- **Disable expansion tokens for untrusted inputs** where possible.
- **Use SSH key-based authentication** to reduce risks associated with **user input manipulation**.
- **Restrict SSH access** to trusted users and implement **strict firewall rules** to limit exposure.

Reference

- **OpenSSH Security Advisories:** https://www.openssh.com/security.html
- **CVE Database:** https://cve.mitre.org
- **NIST NVD:** https://nvd.nist.gov

e. Denial of Service (DoS) Vulnerability in Apache HTTP Server Due to Partial HTTP Requests

This vulnerability allows **remote attackers** to cause a **Denial of Service (DoS)** by sending **partial HTTP requests**, a method leveraged by the **Slowloris attack**.

- **Cause**: Incomplete HTTP headers or deliberately slow transmissions prevent the server from closing idle connections, leading to resource exhaustion.
- **Affected Configurations**: Servers running **Apache HTTP Server versions prior to 2.2.15** that **lack the mod_reqtimeout module** are particularly vulnerable.

Impact- **Medium** (5.0)

Affected Versions

- Apache HTTP Server 1.x and 2.x (prior to 2.2.15)

CVE-ID- **Medium** (5.0)

Technical Impact

- **Server Resource Exhaustion** – Attackers can consume available connections, preventing new legitimate requests from being processed.
- **Prolonged Downtime** – The attack can be sustained with minimal bandwidth, keeping the server unresponsive for extended periods.
- **No Authentication Required** – Any remote attacker can exploit this flaw without needing authentication.

Mitigation

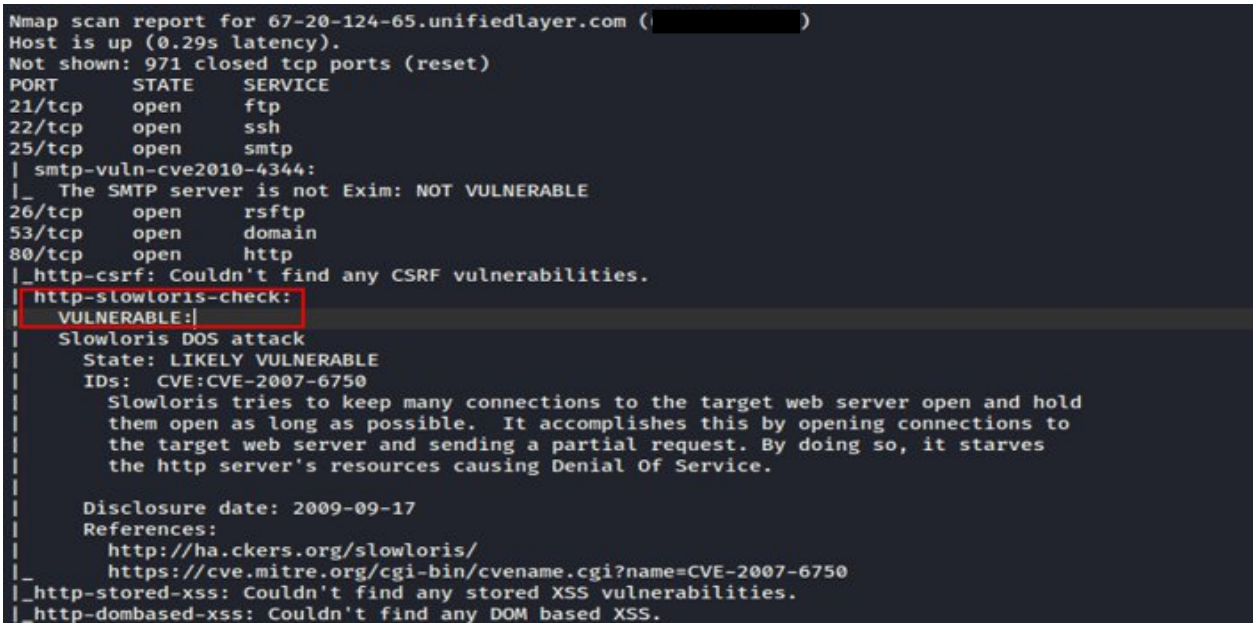
- **Upgrade Apache HTTP Server** to **2.2.15 or later**, where the **mod_reqtimeout** module is included by default.
- **Enable mod_reqtimeout** to enforce timeouts for incomplete requests(code):
    - RequestReadTimeout header=20-40,MinRate=500 body=20,MinRate=500

- **Use a Web Application Firewall (WAF)** to detect and block Slowloris-style attacks.
- **Limit the number of concurrent connections** from a single IP address using firewall rules or **mod_limitipconn**.
- **Implement Reverse Proxies (e.g., Nginx, Cloudflare, or AWS ALB)** to mitigate DoS attempts.
- **Monitor traffic patterns** for unusually slow or persistent connections.

Reference

- **Apache HTTP Server Security Advisories:** https://httpd.apache.org/security/
- **NIST NVD:** https://nvd.nist.gov
- **OWASP Slowloris Attack Reference:** https://owasp.org/www-community/attacks/Slowloris

Proof of concept



f. Denial of Service Vulnerability in Apache HTTP Server Due to Improper Locking

Impact- **Medium** (**5.0**)

Affected Versions

- **Apache HTTP Server 1.3.x** (prior to **1.3.30**)
- **Apache HTTP Server 2.0.x** (prior to **2.0.49**

CVE-ID- **CVE-2004-0174** – https://nvd.nist.gov/vuln/detail/CVE-2004-0174

Technical Impact

- **Server Unresponsiveness** – Critical resources may be locked, making the server unavailable.
- **Disrupted Web Services** – New client requests cannot be processed until the issue is resolved.
- **Exploitation Without Authentication** – The flaw can be triggered remotely without credentials.

Mitigation

- **Upgrade Apache HTTP Server** to **1.3.30 or later** (for 1.3.x users) or **2.0.49 or later** (for 2.0.x users).
- **Limit the Number of Listening Sockets** to avoid configurations prone to this issue.
- **Use a Reverse Proxy** (e.g., Nginx, HAProxy) to mitigate direct exposure to malicious connections.
- **Monitor Server Logs** for unusual connection patterns or high-frequency short-lived connections.

Reference

- **Apache HTTP Server Security Advisories:** https://httpd.apache.org/security/
- **NIST NVD:** https://nvd.nist.gov


Low

a. Anonymous FTP Login Vulnerability

The **Anonymous FTP Login Vulnerability** allows unauthorized users to access an FTP server without authentication. If misconfigured, it can expose sensitive files, leading to data leaks, unauthorized modifications, or even full system compromise. Attackers may exploit this flaw to access confidential data, upload malicious files, or use the FTP server as a foothold for further attacks within the network.

Impact:- Low

CVE-ID - CVE-1999-0497 – https://nvd.nist.gov/vuln/detail/CVE-1999-0497

Technical Impact-

- **Unauthorized Data Access** – Attackers can access sensitive files, leading to data breaches.
- **Information Disclosure** – Exposure of system details aids in further attacks.
- **Malware Upload** – If write access is enabled, attackers can upload malicious files.
- **Privilege Escalation** – Misconfigurations may allow attackers to gain higher access.
- **Lateral Movement** – The FTP server can be used to attack other network resources.
- **Service Abuse** – Attackers may exploit the server for illegal content hosting or DDoS attacks.

Mitigation

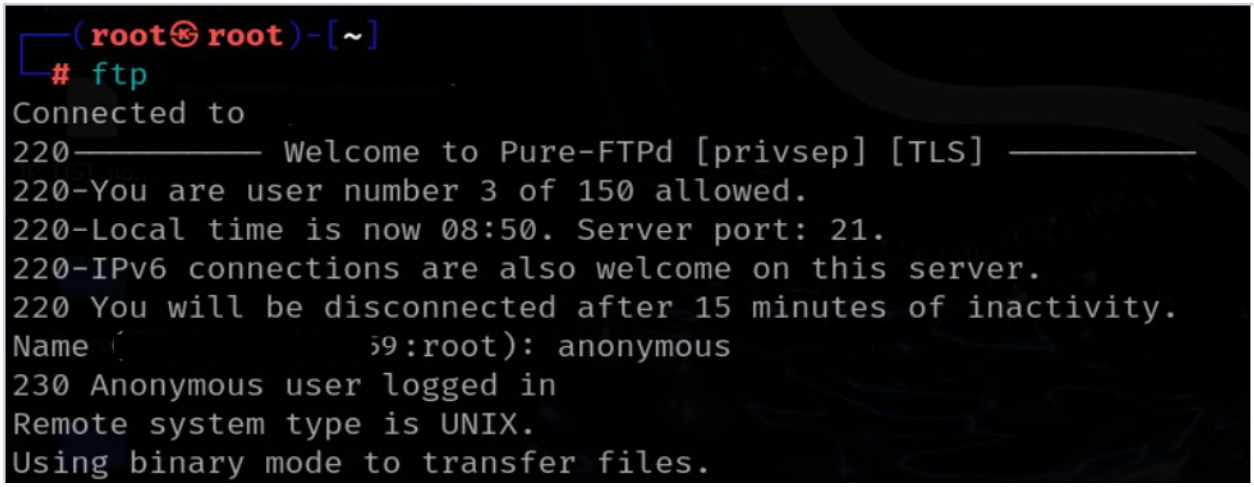
- **Disable Anonymous FTP Access** – Restrict access to authenticated users only.
- **Enforce Strong Authentication** – Use secure credentials and implement multi-factor authentication (MFA) if possible.
- **Restrict File Permissions** – Ensure anonymous users cannot upload, modify, or delete files.

- **Use Secure Protocols** – Replace FTP with **SFTP (SSH File Transfer Protocol) or FTPS (FTP Secure)** to encrypt data.
- **Firewall & Network Restrictions** – Limit FTP access to trusted IP addresses and block unauthorized connections.
- **Regular Auditing & Monitoring** – Log and review FTP activity for signs of suspicious access or data breaches.
- **Disable Unused FTP Services** – If FTP is not necessary, disable the service entirely.

Reference

- **CVE Database (Common Vulnerabilities and Exposures)** – Search for FTP-related vulnerabilities. https://cve.mitre.org
- **NIST National Vulnerability Database (NVD)** – Provides security guidelines and vulnerability details. https://nvd.nist.gov
- **OWASP (Open Web Application Security Project)** – Security best practices for FTP and file transfer. https://owasp.org
- **SANS Internet Storm Center** – Reports on FTP threats and security advisories. https://isc.sans.edu
- **Vendor Security Advisories** – Check FTP server vendors like Microsoft, ProFTPD, vsftpd, or FileZilla for security patches.

Proof of concept



b. Potential Authentication Confusion in OpenSSH Prior to Version 8.9

A security concern was identified in **OpenSSH versions before 8.9** when using **public-key authentication with agent forwarding enabled**. If a **malicious SSH server** is modified to support the "None" authentication method, a client may be **unable to distinguish** whether FIDO authentication is confirming their connection **or allowing the server to authenticate on their behalf** to another system.
Although this behavior may cause authentication confusion, the **OpenSSH development team disputes** this as a vulnerability, stating that it does not constitute an authentication bypass.

Impact:- Low

Affected Versions- OpenSSH versions prior to 8.9

CVE-ID- CVE-2021-36368 – https://nvd.nist.gov/vuln/detail/CVE-2021-36368

Technical Impact

- **Authentication Confusion:** Users may mistakenly authorize unintended authentication actions.
- **Man-in-the-Middle Risk:** An attacker controlling a compromised SSH server could trick a client into approving actions unknowingly.
- **Possible Credential Forwarding Risk:** If agent forwarding is enabled, attackers could misuse the client's credentials

Mitigation

- Upgrade OpenSSH – Use version 8.9 or later to mitigate potential confusion.
- Disable Agent Forwarding – Avoid using -A unless absolutely necessary.
- Enable Verbose Logging – Always use -oLogLevel=verbose to ensure authentication details are clearly logged.
- Use Trusted SSH Servers – Only connect to known and verified SSH servers to prevent MITM risks.

- Monitor SSH Activity – Review logs to detect unexpected authentication behavior (/var/log/auth.log on Linux).

Reference

- OpenSSH Release Notes: https://www.openssh.com/releasenotes.html
- CVE Database: https://cve.mitre.org
- NIST NVD Report: https://nvd.nist.gov
- OWASP Security Guidelines for SSH: https://owasp.org

Learning and Reflection

- **Learning and Reflection**
  This section documents the new learnings and overall experience gained by the individual during the network penetration testing project. Since I worked alone on this project, this reflection is solely based on my individual experience.
- **New Learnings:**
  Throughout this project, I gained significant insights and new skills in various areas of cybersecurity, including technology and project management:
  - **Advanced Penetration Testing Techniques:** I deepened my understanding of black-box penetration testing, including the methodologies and techniques used to identify vulnerabilities from an external perspective. I learned how to effectively simulate real-world attack scenarios and think like an attacker.
  - **Proficiency with Security Tools:** I enhanced my proficiency with essential security tools such as Nmap, Metasploit, Nessus, Wireshark, and Burp Suite. I gained hands-on experience in using these tools to discover vulnerabilities, exploit weaknesses, and analyze network traffic.
  - **Vulnerability Analysis and Risk Assessment:** I improved my ability to analyze vulnerabilities, assess their potential impact, and assign severity ratings using industry-standard frameworks such as CVSS. I learned how to prioritize vulnerabilities based on risk factors such as likelihood and impact.
  - **Remediation Strategies:** I expanded my knowledge of remediation strategies and best practices for mitigating identified vulnerabilities. I gained experience in developing actionable recommendations for patching, configuration changes, and security enhancements.
  - **Project Management Skills:** Working independently on this project allowed me to develop and refine my project management skills. I learned how to effectively plan, organize, and execute a

penetration test from start to finish, managing my time and resources efficiently.

- **Documentation and Reporting:** I improved my documentation and reporting skills by creating a comprehensive vulnerability report, a risk assessment matrix, and a prioritized remediation plan. I learned how to effectively communicate complex technical information to both technical and non-technical audiences.

- **Overall Experience:**

  This project provided an invaluable opportunity to apply my cybersecurity knowledge and skills in a practical setting. Working alone on the project allowed me to take ownership of the entire process, from planning to reporting. I gained hands-on experience in identifying, analyzing, and mitigating security vulnerabilities in network infrastructure.

  1. **Independence and Responsibility:** I developed independence and a sense of responsibility by managing the entire project lifecycle. This included defining the scope, conducting the testing, analyzing the results, and documenting the findings.
  2. **Problem-Solving Skills:** I enhanced my problem-solving skills by troubleshooting technical challenges and finding creative solutions to overcome obstacles encountered during the penetration test.
  3. **Continuous Learning:** The project reinforced the importance of continuous learning and staying up-to-date with the latest security threats and best practices. I had to research and learn about new vulnerabilities and exploitation techniques to effectively conduct the penetration test.
  4. **Confidence Building:** Successfully completing the project boosted my confidence in my cybersecurity skills and abilities. I gained a sense of accomplishment from identifying and mitigating critical security vulnerabilities.
  5. **Professional Growth:** Overall, this project contributed to my professional growth by enhancing my technical skills, project management abilities, and communication skills. It provided me with valuable experience that I can leverage in future cybersecurity endeavors.

Conclusion and Future Scope

- **Recap Objectives and Achievements:**
  This network penetration testing project successfully achieved its objectives by identifying critical security vulnerabilities within the specified network infrastructure. Through a black-box testing approach, potential weaknesses were uncovered, analyzed, and documented. The project resulted in a detailed vulnerability report, a risk assessment matrix, and a prioritized remediation plan. Key achievements include:
    - Identification of critical vulnerabilities such as BlueKeep (CVE-2019-0708), OpenSSH PKCS#11 Insufficient Search Path Vulnerability, Samba Remote Code Execution Vulnerability, and others.
    - Assessment of risks based on industry-standard frameworks like CVSS, enabling prioritized remediation efforts.
    - Provision of actionable and practical recommendations for mitigating identified vulnerabilities to improve the overall security posture.

- **Future Scope:**
  To further enhance the organization's security posture and expand the scope of this project, the following areas are recommended for future exploration:
    - **Web Application Security Testing:** Extend the testing scope to include web applications hosted within the network to identify vulnerabilities such as cross-site scripting (XSS), SQL injection, and authentication flaws.
    - **Wireless Network Security Assessment:** Conduct a security assessment of the wireless networks to identify vulnerabilities such as weak encryption, unauthorized access points, and rogue devices.

- **Cloud Security Assessment:** Evaluate the security configuration and compliance of cloud-based resources and services to identify potential misconfigurations, vulnerabilities, and compliance issues.
- **Mobile Application Penetration Testing:** Conduct penetration testing on mobile applications used within the organization, focusing on identifying vulnerabilities such as insecure data storage, authentication flaws, and code injection vulnerabilities.
- **Social Engineering Assessments:** Perform social engineering tests to evaluate the organization's susceptibility to phishing attacks, pretexting, and other social engineering techniques.
- **Security Awareness Training:** Develop and implement security awareness training programs to educate employees about common security threats and best practices for protecting sensitive information.
- **Automated Vulnerability Scanning:** Implement automated vulnerability scanning tools to continuously monitor the network for new vulnerabilities and misconfigurations.
- **Regular Security Audits:** Conduct regular security audits and reviews to ensure that security controls are effective and up-to-date.
- **Mobile Pentesting:** Integrate mobile application penetration testing to evaluate the security of mobile apps used by the organization. This will include assessing vulnerabilities such as insecure data storage, authentication flaws, and code injection risks specific to mobile platforms.

References

1. **NIST National Vulnerability Database (NVD)** – https://nvd.nist.gov
2. **Apache HTTP Server Security Advisories** – https://httpd.apache.org/security/
3. **OpenSSH Security Advisories** – https://www.openssh.com/security.html
4. **Microsoft Security Response Center (MSRC) – BlueKeep Advisory** – https://msrc.microsoft.com/update-guide/vulnerability/CVE-2019-0708
5. **Samba Security Advisories** – https://www.samba.org/samba/security/
6. **ChatGPT – AI-Assisted Research and Explanation** – https://openai.com/
7. **Jetty Security Advisories** – https://github.com/eclipse/jetty.project/security
8. **Exim Security Advisories** – https://www.exim.org/security.html
9. **MySQL Security Updates (Oracle)** – https://www.oracle.com/security-alerts/
10. **OWASP Top 10 Web Security Risks** – https://owasp.org/www-project-top-ten/
11. **MikroTik Security Advisories** – https://mikrotik.com/support
12. **VSFTPD Security Updates** – https://security-tracker.debian.org/tracker/source-package/vsftpd
13. **Nmap (Network Mapper) Security & Vulnerability Scanning** – https://nmap.org/
14. **Canva – Used for Graph and Visualization Creation** – https://www.canva.com/
15. **CVE Details Database** – https://www.cvedetails.com/
16. **CVSS v3 Calculator – NIST** – https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator