Vulnerability Assessment and Risk Analysis Report

1. Introduction

Vulnerability Assessment and Penetration Testing (VAPT) play a crucial role in identifying security weaknesses within systems before they can be exploited by malicious attackers. Regular security assessments help organizations understand their exposure to risks and improve their overall security posture by addressing vulnerabilities at an early stage.

In this task, I performed a vulnerability assessment on a controlled lab environment using open-source security tools. The assessment focused on identifying common system and network vulnerabilities, analyzing their severity, and prioritizing risks based on industry-standard scoring methods. A vulnerable virtual machine was used as the target system to safely simulate real-world security issues without impacting production environments.

The assessment was conducted using Kali Linux as the testing platform, with OpenVAS used for vulnerability scanning and analysis. Identified vulnerabilities were further evaluated using CVSS scoring and a qualitative risk matrix to determine their potential impact and likelihood of exploitation. The findings were documented in a structured manner to support remediation planning and risk mitigation.

The objective of this report is to present the methodology followed, tools used, vulnerabilities identified, and risk assessment performed during the VAPT process. This report also highlights key security issues commonly found in vulnerable systems and provides recommendations to reduce potential security risks.

2. Objective and Scope

The primary objective of this task was to perform a Vulnerability Assessment and Penetration Testing (VAPT) exercise using open-source security tools in a controlled laboratory environment. The assessment aimed to identify common security vulnerabilities, evaluate their severity, and prioritize risks using standardized risk assessment techniques.

During this task, I focused on identifying weaknesses related to outdated software, exposed services, and insecure configurations. The vulnerabilities identified through automated scanning were analyzed using CVSS scoring to understand their potential impact and exploitability. A qualitative risk matrix was then used to categorize risks as high, medium, or low, enabling effective prioritization for remediation.

The scope of this assessment was limited to a virtualized test environment to ensure safe and ethical testing. Kali Linux was used as the attacker machine, and a vulnerable system was used as the target for assessment. Due to local system security constraints, Metasploitable 2 was used instead of Metasploitable 3, while still fulfilling all vulnerability assessment and learning objectives of the task. No testing was performed on live or production systems.

The assessment covered system-level and network-level vulnerabilities identified through vulnerability scanning tools. Exploitation of vulnerabilities was not performed, as the focus of this task was on vulnerability identification, risk evaluation, and documentation rather than active exploitation.

3. Tools and Environment

The vulnerability assessment was performed in a controlled virtual environment to ensure safe and ethical testing. A virtualized setup was used to simulate real-world systems without affecting any production environment.

Kali Linux was used as the primary operating system for conducting the assessment. The virtual machines were configured using VMware, which enabled proper network communication between the attacker and target systems.

OpenVAS was used to perform vulnerability scanning and identify security weaknesses in the target system. The scan results were analyzed through the OpenVAS web interface, and relevant findings were documented for further analysis. The NVD CVSS Calculator was used to understand and validate the severity of critical vulnerabilities.

Metasploitable 2 was used as the target system for this assessment. It is an intentionally vulnerable virtual machine designed for security testing and learning purposes, allowing vulnerabilities to be identified in a safe environment.

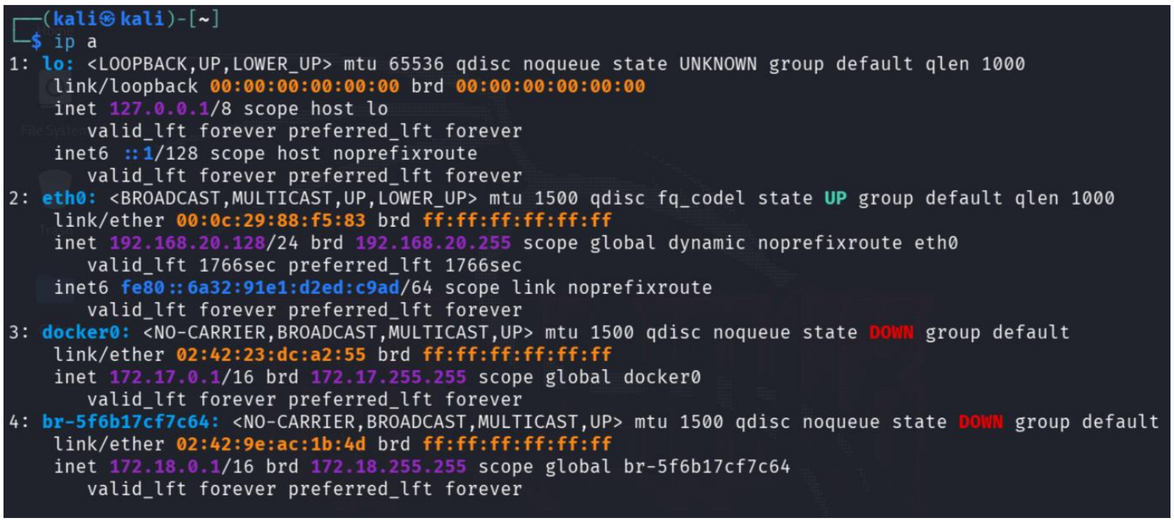
4. VAPT Methodology

I followed a structured Vulnerability Assessment and Penetration Testing (VAPT) methodology to ensure that the security assessment was systematic, controlled, and easy to analyze. The methodology was divided into four main phases: **Planning, Discovery, Assessment, and Reporting**.

4.1 Planning

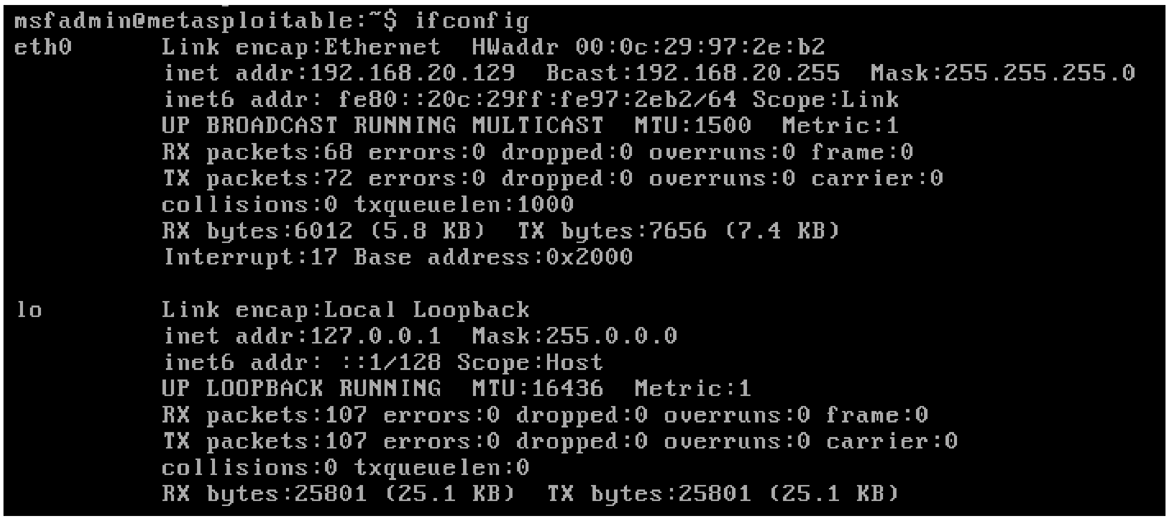
In the planning phase, I prepared the testing environment and identified the target system before starting any vulnerability scans.

- Identified Kali Linux as the attacker machine.
- Identified Metasploitable 2 as the target vulnerable system.
- Verified network configuration and IP address of the Kali Linux machine.
- Verified IP address of the Metasploitable machine to ensure both systems were on the same network.
- Tested connectivity between Kali Linux and Metasploitable to confirm successful communication.
- Ensured the assessment was conducted in an isolated virtual lab environment.



```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:88:f5:83 brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.128/24 brd 192.168.20.255 scope global dynamic noprefixroute eth0
       valid_lft 1766sec preferred_lft 1766sec
    inet6 fe80::6a32:91e1:d2ed:c9ad/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:23:dc:a2:55 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
4: br-5f6b17cf7c64: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:9e:ac:1b:4d brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global br-5f6b17cf7c64
       valid_lft forever preferred_lft forever
```

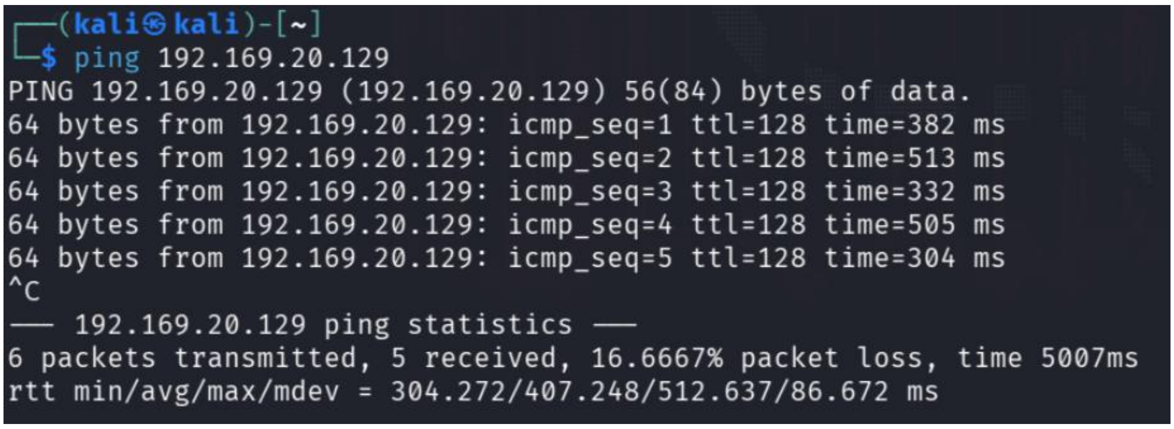
Figure 1: Kali Linux Network Configuration and IP Address Verification



```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:97:2e:b2
          inet addr:192.168.20.129  Bcast:192.168.20.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe97:2eb2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:68 errors:0 dropped:0 overruns:0 frame:0
          TX packets:72 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6012 (5.8 KB)  TX bytes:7656 (7.4 KB)
          Interrupt:17 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:107 errors:0 dropped:0 overruns:0 frame:0
          TX packets:107 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:25801 (25.1 KB)  TX bytes:25801 (25.1 KB)
```

Figure 2: Metasploitable 2 Network Configuration and IP Address



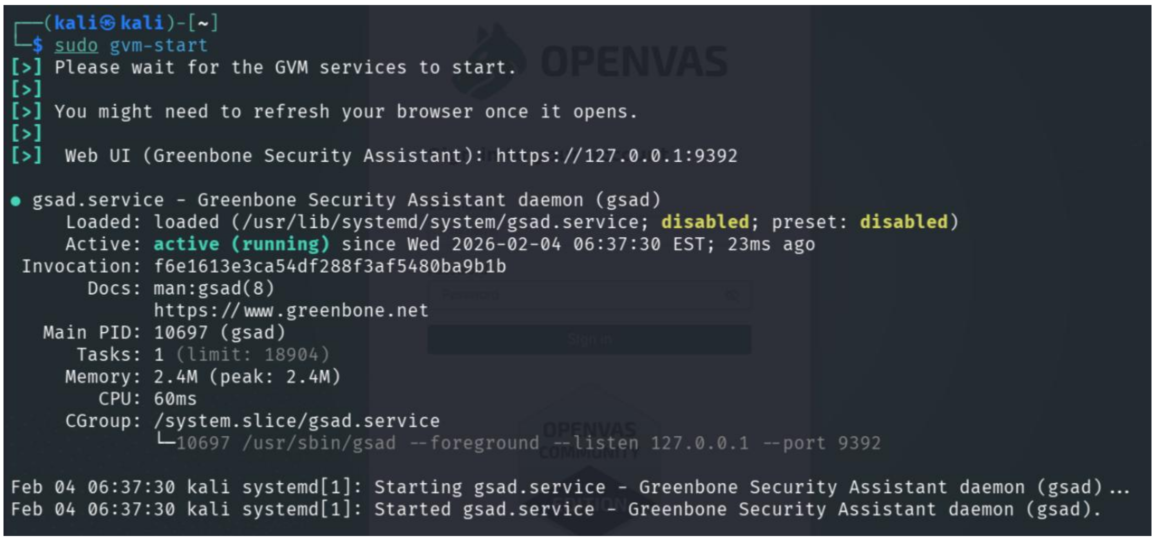
```
┌──(kali㉿kali)-[~]
└─$ ping 192.169.20.129
PING 192.169.20.129 (192.169.20.129) 56(84) bytes of data.
64 bytes from 192.169.20.129: icmp_seq=1 ttl=128 time=382 ms
64 bytes from 192.169.20.129: icmp_seq=2 ttl=128 time=513 ms
64 bytes from 192.169.20.129: icmp_seq=3 ttl=128 time=332 ms
64 bytes from 192.169.20.129: icmp_seq=4 ttl=128 time=505 ms
64 bytes from 192.169.20.129: icmp_seq=5 ttl=128 time=304 ms
^C
--- 192.169.20.129 ping statistics ---
6 packets transmitted, 5 received, 16.6667% packet loss, time 5007ms
rtt min/avg/max/mdev = 304.272/407.248/512.637/86.672 ms
```

Figure 3: Network Connectivity Verification Between Kali Linux and Metasploitable 2 Using Ping

4.2 Discovery

In the discovery phase, I initiated the vulnerability scanning process to identify exposed services and potential security weaknesses.

- Started Greenbone Vulnerability Manager (GVM) services on Kali Linux.
- Accessed the OpenVAS web interface through the local browser.
- Verified that OpenVAS services were running successfully.
- Created a new scan task targeting the Metasploitable 2 system.
- Configured the scan using the default OpenVAS scanner and scan configuration.



```
┌──(kali㉿kali)-[~]
└─$ sudo gvm-start
[>] Please wait for the GVM services to start.
[>]
[>] You might need to refresh your browser once it opens.
[>]
[>]  Web UI (Greenbone Security Assistant): https://127.0.0.1:9392

● gsad.service - Greenbone Security Assistant daemon (gsad)
     Loaded: loaded (/usr/lib/systemd/system/gsad.service; disabled; preset: disabled)
     Active: active (running) since Wed 2026-02-04 06:37:30 EST; 23ms ago
 Invocation: f6e1613e3ca54df288f3af5480ba9b1b
       Docs: man:gsad(8)
             https://www.greenbone.net
   Main PID: 10697 (gsad)
      Tasks: 1 (limit: 18904)
     Memory: 2.4M (peak: 2.4M)
        CPU: 60ms
     CGroup: /system.slice/gsad.service
             └─10697 /usr/sbin/gsad --foreground --listen 127.0.0.1 --port 9392

Feb 04 06:37:30 kali systemd[1]: Starting gsad.service - Greenbone Security Assistant daemon (gsad)...
Feb 04 06:37:30 kali systemd[1]: Started gsad.service - Greenbone Security Assistant daemon (gsad).
```

Figure 4: Greenbone Vulnerability Manager (GVM) Services Started Successfully

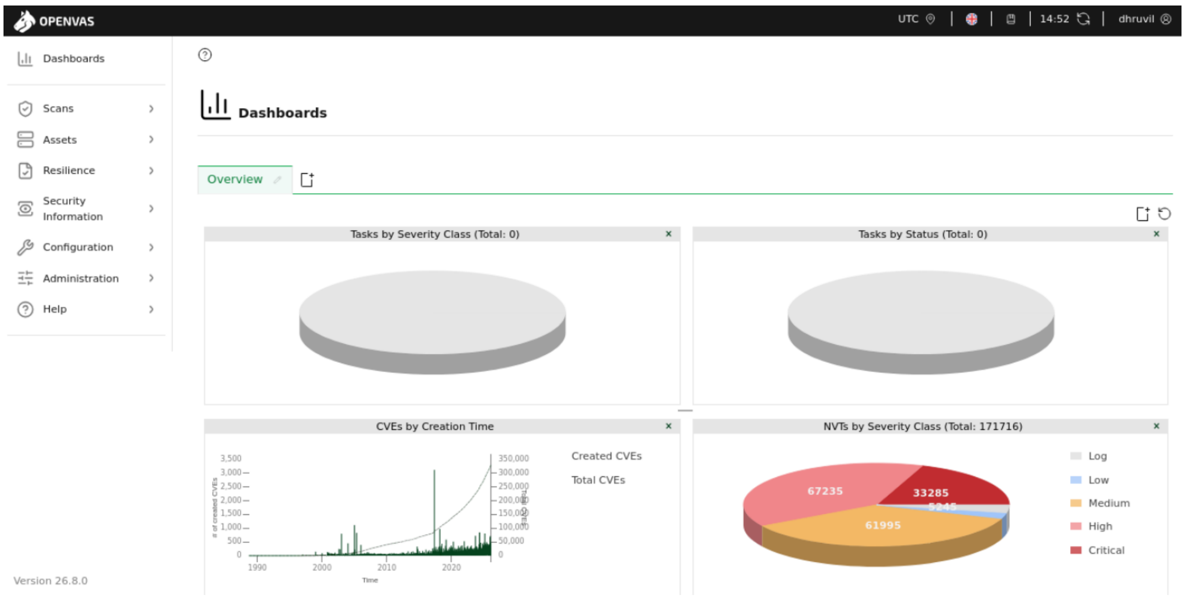
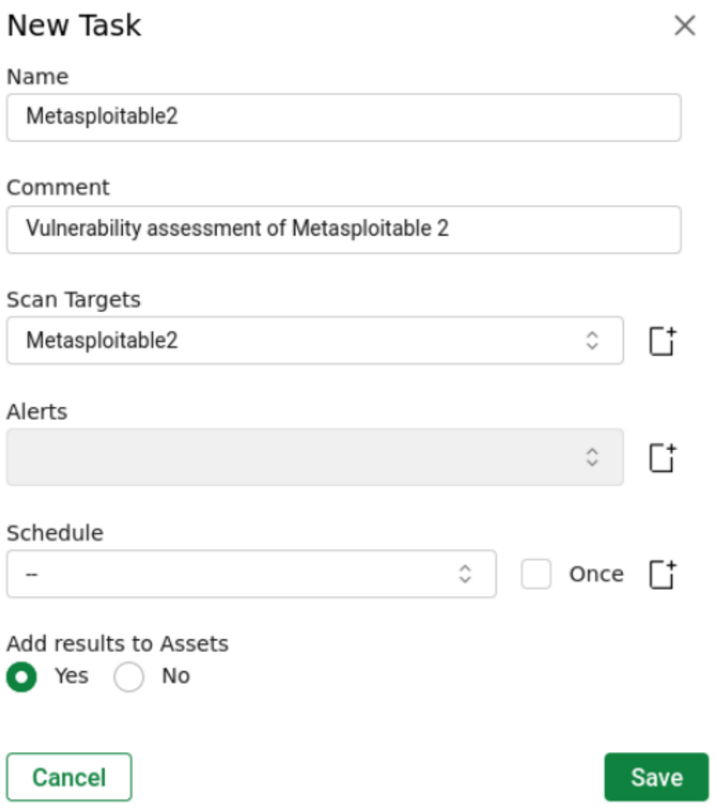


Figure 5: OpenVAS Web Interface Dashboard



New Task

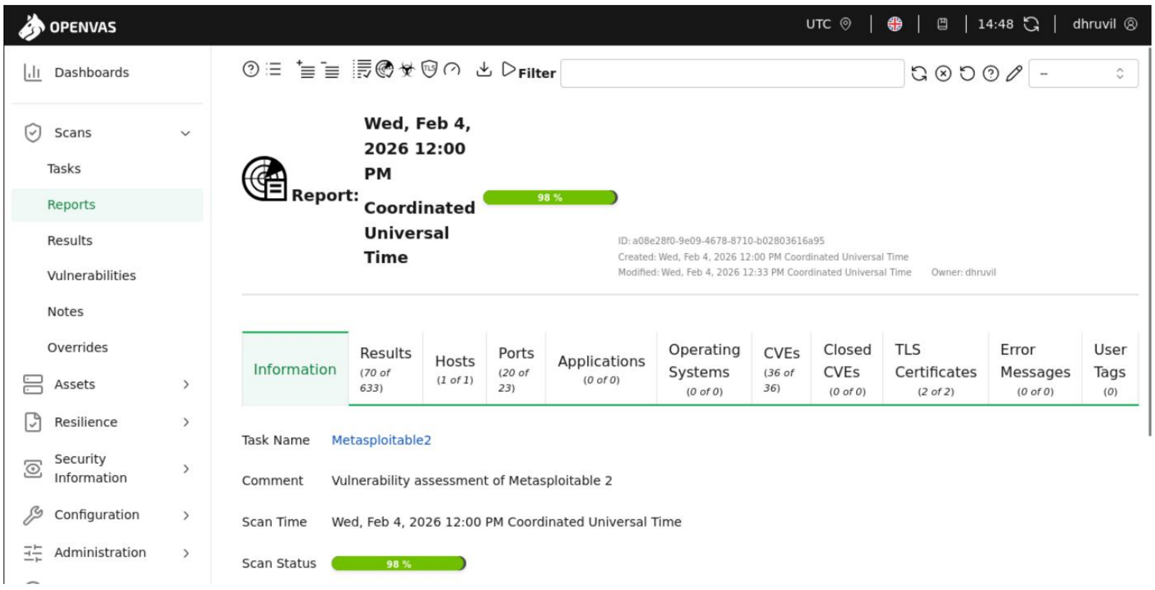
Name	Metasploitable2
Comment	Vulnerability assessment of Metasploitable 2
Scan Targets	Metasploitable2
Alerts	
Schedule	-- ☐ Once
Add results to Assets	[✓ Yes] No

Figure 6: Creation of Vulnerability Scan Task for Metasploitable 2

4.3 Assessment

In the assessment phase, I analyzed the results generated by the vulnerability scan and identified high-risk issues.

- Executed the vulnerability scan against the Metasploitable 2 system.
- Monitored scan progress and confirmed scan completion.
- Reviewed scan results categorized by severity levels.
- Identified critical vulnerabilities based on CVSS scores.
- Analyzed detailed vulnerability information for major findings.
- Validated vulnerability severity using the NVD CVSS Calculator.



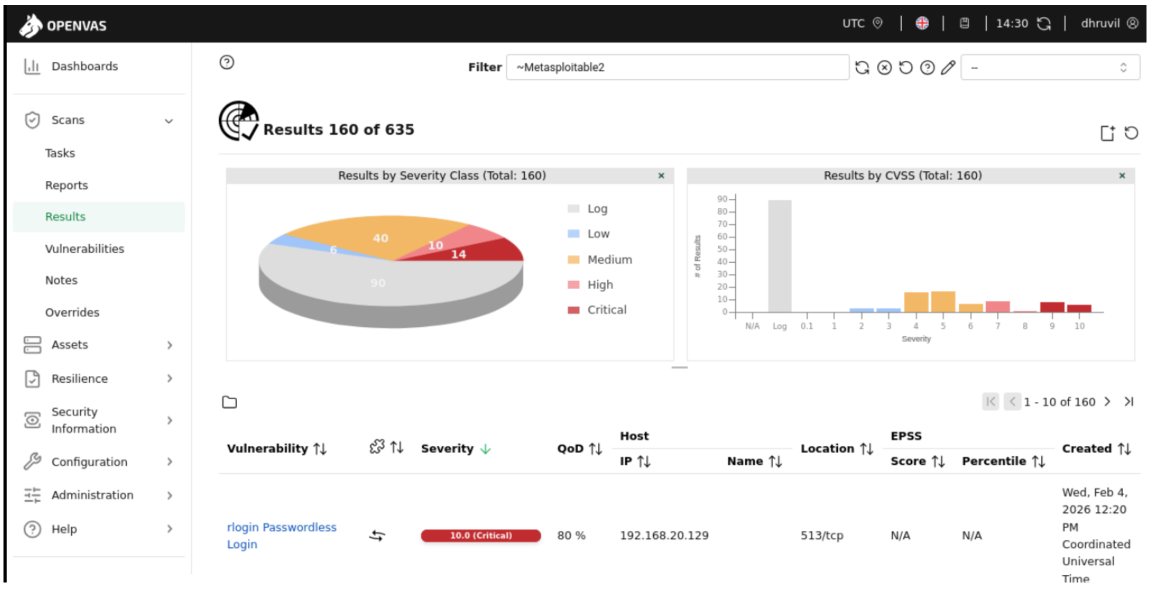
Report: Wed, Feb 4, 2026 12:00 PM Coordinated Universal Time

ID: a08e28f0-9e09-4678-8710-b02803616a95
Created: Wed, Feb 4, 2026 12:00 PM Coordinated Universal Time
Modified: Wed, Feb 4, 2026 12:33 PM Coordinated Universal Time   Owner: dhruvil

Information	Results (70 of 633)	Hosts (1 of 1)	Ports (20 of 23)	Applications (0 of 0)	Operating Systems (0 of 0)	CVEs (36 of 36)	Closed CVEs (0 of 0)	TLS Certificates (2 of 2)	Error Messages (0 of 0)	User Tags (0)

Task Name	Metasploitable2
Comment	Vulnerability assessment of Metasploitable 2
Scan Time	Wed, Feb 4, 2026 12:00 PM Coordinated Universal Time
Scan Status	98 %

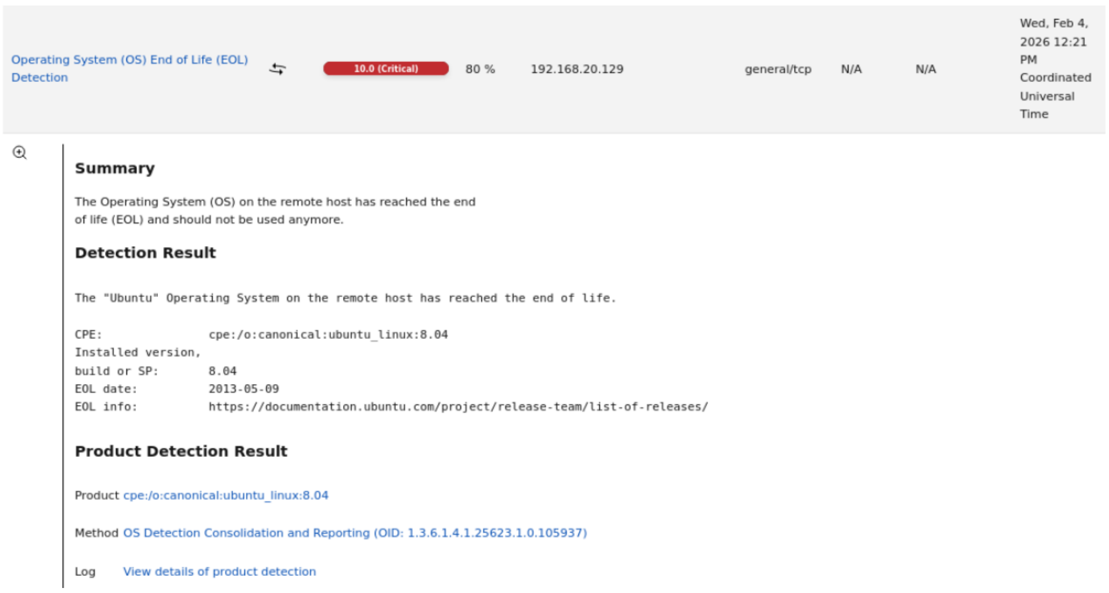
Figure 7: OpenVAS Vulnerability Scan Report Summary



Results 160 of 635

Vulnerability		Severity	QoD	Host IP	Host Name	Location	EPSS Score	EPSS Percentile	Created
rlogin Passwordless Login		10.0 (Critical)	80 %	192.168.20.129		513/tcp	N/A	N/A	Wed, Feb 4, 2026 12:20 PM Coordinated Universal Time

Figure 8: Vulnerability Severity Distribution Identified by OpenVAS



Operating System (OS) End of Life (EOL) Detection	10.0 (Critical)	80 %	192.168.20.129	general/tcp	N/A	N/A	Wed, Feb 4, 2026 12:21 PM Coordinated Universal Time

Summary

The Operating System (OS) on the remote host has reached the end of life (EOL) and should not be used anymore.

Detection Result

```
The "Ubuntu" Operating System on the remote host has reached the end of life.

CPE:              cpe:/o:canonical:ubuntu_linux:8.04
Installed version,
build or SP:      8.04
EOL date:         2013-05-09
EOL info:         https://documentation.ubuntu.com/project/release-team/list-of-releases/
```

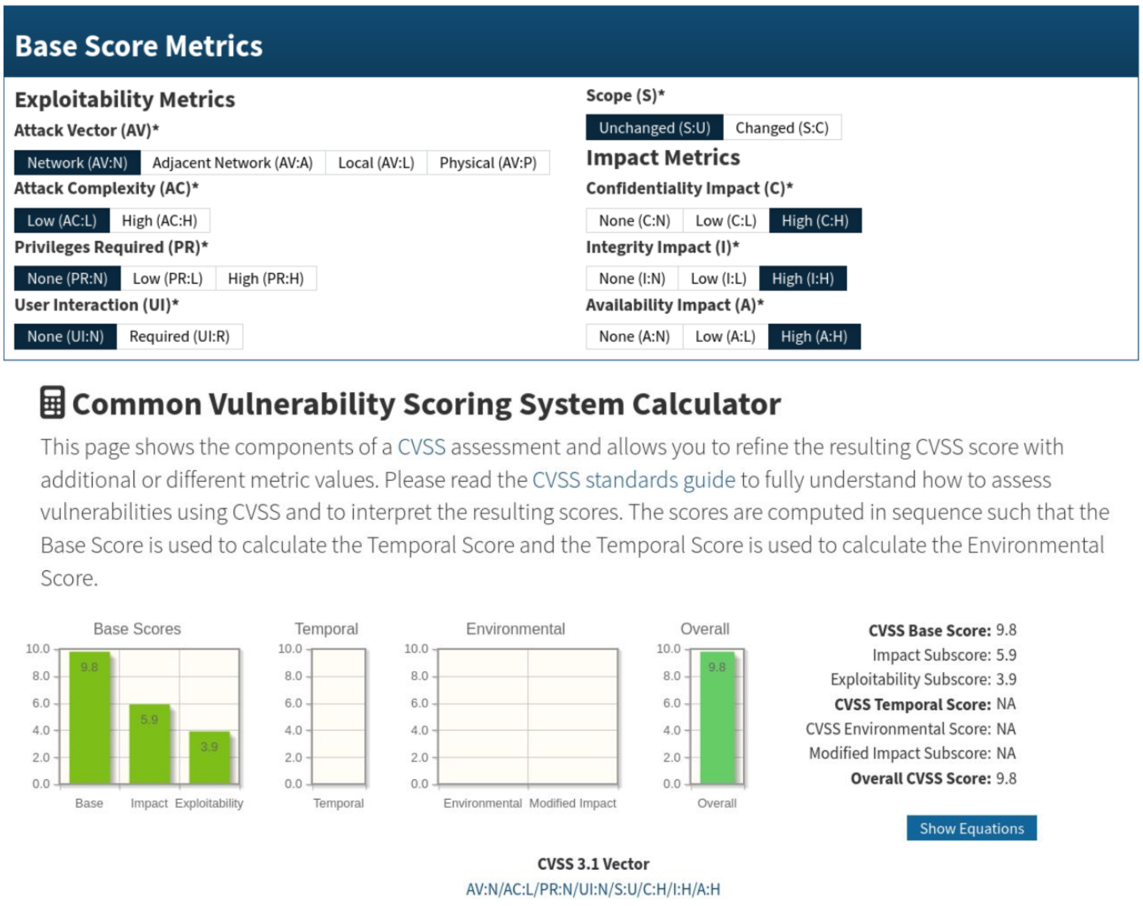
Product Detection Result

Product cpe:/o:canonical:ubuntu_linux:8.04

Method OS Detection Consolidation and Reporting (OID: 1.3.6.1.4.1.25623.1.0.105937)

Log View details of product detection

Figure 9: Detailed View of Critical Vulnerability – Operating System End of Life (EOL)



Base Score Metrics

Exploitability Metrics

Attack Vector (AV)*: [✓ Network (AV:N)] Adjacent Network (AV:A) Local (AV:L) Physical (AV:P)

Attack Complexity (AC)*: [✓ Low (AC:L)] High (AC:H)

Privileges Required (PR)*: [✓ None (PR:N)] Low (PR:L) High (PR:H)

User Interaction (UI)*: [✓ None (UI:N)] Required (UI:R)

Scope (S)*: [✓ Unchanged (S:U)] Changed (S:C)

Impact Metrics

Confidentiality Impact (C)*: None (C:N) Low (C:L) [✓ High (C:H)]

Integrity Impact (I)*: None (I:N) Low (I:L) [✓ High (I:H)]

Availability Impact (A)*: None (A:N) Low (A:L) [✓ High (A:H)]

Common Vulnerability Scoring System Calculator

This page shows the components of a CVSS assessment and allows you to refine the resulting CVSS score with additional or different metric values. Please read the CVSS standards guide to fully understand how to assess vulnerabilities using CVSS and to interpret the resulting scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.

CVSS Base Score: 9.8
Impact Subscore: 5.9
Exploitability Subscore: 3.9
CVSS Temporal Score: NA
CVSS Environmental Score: NA
Modified Impact Subscore: NA
Overall CVSS Score: 9.8

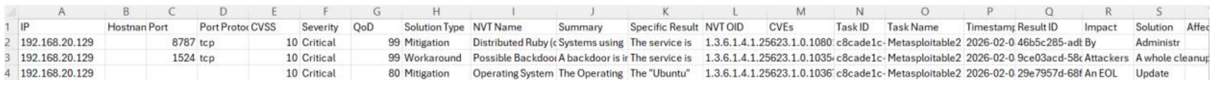
CVSS 3.1 Vector
AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Figure 10: CVSS v3.1 Severity Calculation Using NVD CVSS Calculator

4.4 Reporting

In the reporting phase, I documented the identified vulnerabilities and prioritized them based on risk.

- Exported vulnerability scan results from OpenVAS for documentation.
- Filtered critical vulnerabilities using a CSV/spreadsheet format.
- Recorded key details such as IP address, port number, service name, CVSS score, and severity level.
- Performed risk prioritization by analyzing the **likelihood of exploitation** and **potential impact** of critical vulnerabilities.
- Classified vulnerabilities as high, medium, or low risk based on severity and impact analysis.
- Prepared the final findings for remediation and inclusion in this report



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	
1	IP	Hostnan	Port	Port Protoc	CVSS	Severity	QoD	Solution Type	NVT Name	Summary	Specific Result	NVT OID	CVEs	Task ID	Task Name	Timestamp	Result ID	Impact	Solution	Affec
2	192.168.20.129		8787	tcp	10	Critical	99	Mitigation	Distributed Ruby (c	Systems using	The service is	1.3.6.1.4.1.25623.1.0.1080		c8cade1c-	Metasploitable2	2026-02-0	46b5c285-adl	By	Administr	
3	192.168.20.129		1524	tcp	10	Critical	99	Workaround	Possible Backdoo	A backdoor is ir	The service is	1.3.6.1.4.1.25623.1.0.1035		c8cade1c-	Metasploitable2	2026-02-0	9ce03acd-58c	Attackers	A whole cleanup	
4	192.168.20.129				10	Critical	80	Mitigation	Operating System	The Operating	The "Ubuntu"	1.3.6.1.4.1.25623.1.0.1036		c8cade1c-	Metasploitable2	2026-02-0	29e7957d-68f	An EOL	Update	

Figure 11: Filtered CSV Showing Documented Critical Vulnerabilities

5. Vulnerability Assessment Results

In this section, I present the key findings identified during the vulnerability assessment of the Metasploitable 2 system. The results were obtained using OpenVAS and were analyzed to identify critical and high-risk security issues.

5.1 Vulnerability Scan Overview

After completing the vulnerability scan, I reviewed the overall results to understand the security posture of the target system.

- The OpenVAS scan revealed multiple vulnerabilities across different severity levels.
- Vulnerabilities were categorized as **Critical, High, Medium, Low, and Log**.
- The presence of several high-risk vulnerabilities confirmed that the target system was intentionally insecure and suitable for assessment and learning purposes.
- Critical vulnerabilities required immediate attention due to their potential impact on system security

5.2 Critical Vulnerability Analysis

I focused on analyzing critical vulnerabilities to understand their impact and associated risks.

- Vulnerabilities with **CVSS scores greater than or equal to 9.0** were classified as Critical.
- One of the most significant findings was the **Operating System End of Life (EOL)** vulnerability.

- The target system was running **Ubuntu 8.04**, which no longer receives security updates.
- An unsupported operating system increases exposure to known exploits and security threats.
- This vulnerability posed a high risk due to both **high exploitability** and **high impact**.

5.3 Other High-Risk Findings

In addition to the critical operating system vulnerability, several other high-risk issues were identified.

- Multiple services were found running on open ports, increasing the attack surface.
- Some services were outdated and susceptible to known vulnerabilities.
- Potential backdoor-related services were detected, which could allow unauthorized access.
- These findings highlighted weak system hardening and insecure default configurations.

5.4 Documentation of Findings

All identified vulnerabilities were documented in a structured manner for further analysis and remediation planning.

- Scan results were exported from OpenVAS for documentation.
- Critical vulnerabilities were filtered and organized using a spreadsheet format.
- Key details such as IP address, port number, service name, CVSS score, and severity level were recorded.
- This documentation supported effective risk assessment and remediation planning.

---

6. Risk Assessment Basics

In this section, I assessed and prioritized the identified vulnerabilities based on their severity, likelihood of exploitation, and potential impact on the system. The risk assessment was performed using CVSS scoring and a qualitative risk evaluation approach.

6.1 CVSS-Based Risk Evaluation

To determine the severity of the identified vulnerabilities, I used the **Common Vulnerability Scoring System (CVSS)**.

- OpenVAS provided CVSS scores for each identified vulnerability.
- Critical vulnerabilities were identified based on **CVSS scores of 9.0 and above**.
- The severity of key findings was further validated using the **NVD CVSS Calculator**.
- CVSS scoring helped in understanding both the **exploitability** and **impact** of vulnerabilities in a standardized manner.

The **Operating System End of Life (EOL)** vulnerability received a **CVSS score of 10.0**, confirming its classification as a **Critical** security risk.

6.2 Likelihood and Impact Analysis

In addition to CVSS scoring, I analyzed vulnerabilities based on their likelihood of exploitation and potential impact.

- **Likelihood** was considered high for vulnerabilities that were publicly known and lacked security patches.
- **Impact** was considered high for vulnerabilities that could lead to system compromise or unauthorized access.
- Vulnerabilities with both high likelihood and high impact were prioritized for immediate attention.

This approach ensured that vulnerabilities were not only evaluated based on numerical scores but also on real-world risk factors.

6.3 Risk Prioritization

Based on CVSS scores and likelihood-impact analysis, vulnerabilities were categorized into risk levels.

- **Critical Risk:** Vulnerabilities with high CVSS scores and severe impact.
- **High Risk:** Vulnerabilities with moderate CVSS scores but significant potential impact.
- **Medium Risk:** Vulnerabilities with limited exploitability or impact.
- **Low Risk:** Vulnerabilities with minimal security impact.

This prioritization helped in identifying which vulnerabilities required urgent remediation and which could be addressed later.

---

7. Common Vulnerabilities

This section discusses common security vulnerabilities identified during the assessment and explains how such vulnerabilities are typically found and understood using vulnerable lab environments.

7.1 Network Vulnerabilities

Network vulnerabilities are commonly caused by misconfigurations, exposed services, and outdated software. During the assessment, I identified several network-related security issues on the target system.

- Multiple services were found running on open ports, increasing the attack surface.
- Some services were outdated and vulnerable to known exploits.
- Insecure default configurations were observed, which could allow unauthorized access.
- The use of an end-of-life operating system significantly increased network-level risk.

These vulnerabilities demonstrate how poor system hardening and lack of patch management can expose systems to serious security threats.

7.2 Web Vulnerabilities

Web vulnerabilities affect applications and services that are accessible through web interfaces. Common examples include **SQL Injection (SQLi)** and **Cross-Site Scripting (XSS)**.

- SQL Injection vulnerabilities allow attackers to manipulate backend databases.
- Cross-Site Scripting vulnerabilities enable attackers to inject malicious scripts into web pages.
- Such vulnerabilities are commonly found in poorly validated input fields and insecure web applications.

Although active exploitation was not performed during this task, the assessment highlighted the importance of securing web-facing services to prevent these types of attacks.

7.3 Learning Through Vulnerable Labs

Intentionally vulnerable platforms are widely used to understand real-world security issues in a safe and legal manner.

- **Metasploitable** was used to identify and analyze real vulnerabilities related to network services and system configurations.
- **VulnHub** provides a collection of vulnerable virtual machines that help in practicing vulnerability identification and security assessment techniques.

These platforms allow security professionals to gain hands-on experience without affecting live systems.

---

8. Remediation and Recommendations

Based on the vulnerabilities identified during the assessment, I proposed the following remediation measures to reduce security risks and improve the overall security posture of the system.

8.1 Operating System Hardening

- I recommend upgrading the operating system to a currently supported version.
- Using a supported operating system ensures regular security updates and patches.
- End-of-life systems should not be used in production environments due to known and unpatched vulnerabilities.

8.2 Service and Port Management

- I recommend disabling unnecessary services running on the system.
- Only essential services should be exposed to the network.
- Unused and insecure ports should be closed to reduce the attack surface.
- Service configurations should be reviewed and hardened to prevent unauthorized access.

8.3 Patch and Update Management

- I recommend implementing regular patch management practices.
- Software and services should be updated frequently to address known vulnerabilities.
- Automated update mechanisms can help ensure timely patch deployment.

8.4 Security Monitoring and Assessment

- Regular vulnerability assessments should be conducted to identify new security issues.
- Vulnerability scanning tools should be used periodically to monitor system security.
- Security assessments should be documented to track improvements and unresolved risks.

---

9. Conclusion

In this task, I performed a Vulnerability Assessment and Penetration Testing (VAPT) exercise on a controlled lab environment using open-source security tools. The assessment helped in identifying common security weaknesses related to outdated software, exposed services, and insecure configurations.

Through vulnerability scanning using OpenVAS, I identified multiple vulnerabilities of varying severity levels. Critical vulnerabilities, such as the Operating System End of Life issue, highlighted the importance of maintaining updated systems and following proper security practices. The use of CVSS scoring and risk analysis helped in prioritizing vulnerabilities based on their potential impact and likelihood of exploitation.

This task provided practical exposure to vulnerability assessment methodologies and strengthened my understanding of risk evaluation and documentation. The assessment emphasized the importance of proactive security testing and proper remediation planning to reduce security risks. Overall, this exercise helped in developing a structured approach towards identifying, analyzing, and documenting vulnerabilities in a secure and ethical manner.

---

10. References

1. Greenbone Security Assistant (OpenVAS) Documentation
   https://greenbone.net
2. National Vulnerability Database (NVD) – CVSS Calculator
   https://nvd.nist.gov/vuln-metrics/cvss
3. Kali Linux Official Documentation
   https://www.kali.org/docs
4. Metasploitable Project Documentation
   https://github.com/rapid7/metasploitable
5. OWASP – Web Application Security Resources
   https://owasp.org
6. NIST Cybersecurity Framework
   https://www.nist.gov/cyberframework