Chapter 2

2.1 **Initiate Vulnerability Scans:-**

1) Prepare the environment

Set up a dedicated scanning environment (a virtual machine or isolated lab network). Make sure you have permission to scan every device or network you will test. Use intentionally vulnerable lab targets if you need exploitable examples.

2) Install and initialize the scanner

Install the OpenVAS/GVM software and complete its initial feed and database setup so the scanner has the latest vulnerability checks. The initialization step downloads vulnerability feeds and prepares the management services.

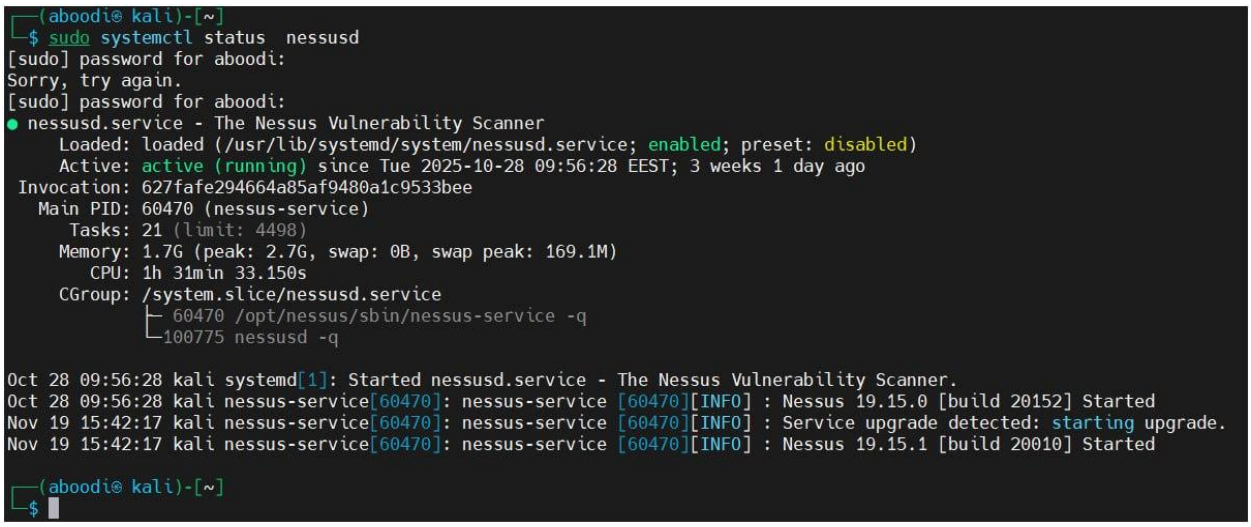
sudo apt update

sudo apt install openvas -y

sudo gvm-setup

3) Start the GVM services

Start the Greenbone services so the web interface and background engines are running. Confirm the management web UI is reachable from your workstation.



4) Create and define targets

In the management interface, create a target definition for each system or network range you want scanned. For each target, record:

- A clear name for the target.
- The host IPs or network ranges included.
- Which ports or protocols to scan (use default port sets or customized lists if needed).

5) Choose a scan configuration

Select an appropriate scan profile/configuration for your objectives (examples: quick checks, full vulnerability discovery, authenticated scans). Pick a profile that balances depth vs. time depending on the scope of your project.

6) Create and schedule a scan task

Create a scan task that ties the selected target to the chosen scan configuration. Optionally schedule it to run immediately or at a specific time. Label tasks clearly so reports can be traced to the corresponding target and configuration.

7) Run the scan and monitor progress

Start the task and monitor its progress through the UI. Note that larger networks or deeper scans take longer. Keep an eye on progress and any errors or connectivity issues reported by the scanner.

8) Review and interpret results

When the scan finishes, open the generated report. Focus on:

- Vulnerability severity (Critical / High / Medium / Low).
- False positives (verify before acting).
- Where vulnerabilities are located (host, service, port).
- CVE identifiers and affected software versions.

9) Export and document findings

Export the report in formats required for your project (PDF/HTML/XML). In your project documentation include:

- Environment and scope
- Scan configuration and rationale

- Key findings and prioritized list of issues
- Reproducible steps to validate the top issues
- Recommended mitigations and remediation steps
- Limitations and any false-positive notes

10) Mitigation & re-scan

Recommend fixes (patching, configuration changes, access control) and, after remediation, re-run scans to confirm issues were resolved.

11) Ethics, permissions & safety

Always run scans only on systems you own or have explicit written authorization to test. Scanning unauthorized systems can be illegal and disruptive. Prefer isolated lab environments to demonstrate findings.

2.2 Analyze Scan Results :-

1. Understand the Report Structure

After a scan finishes, OpenVAS generates a report showing:

- **Host summary:** which systems were scanned and if they were reachable.
- **Vulnerability list:** detected issues per host and per service.
- **Severity levels:** each vulnerability has a numerical score (0.0–10.0) based on CVSS (Common Vulnerability Scoring System).
- **Details section:** description, impact, affected software, and recommended solutions.

---

2. Severity Classification

OpenVAS uses severity ratings to help you prioritize:

- **Critical (9.0–10.0):** High-impact flaws allowing remote compromise or full control.
- **High (7.0–8.9)** Serious weaknesses that could expose sensitive data or system access.
- **Medium (4.0–6.9):** Moderate risks that could assist in further attacks.
- **Low (0.1–3.9):** Minor information leaks or best-practice violations.
- **Log/Info (0.0):** Informational results, not actual vulnerabilities.

When writing your report, group findings by severity so readers can easily see which issues matter most.

---

3. Interpreting Individual Vulnerabilities

For each finding, review:

- **Title/Plugin Name:** what vulnerability or check it represents.
- **Description:** what the scanner detected and why it's considered a risk.
- **Affected Component:** the specific service, software version, or port.
- **Impact:** what an attacker could do if the vulnerability is exploited.
- **Solution/Recommendation:** patch, configuration change, or mitigation steps.
- **CVE Reference:** official vulnerability ID you can cite in your project.

This information shows both technical details and practical importance for your analysis.

4. Validate Findings

Scanners sometimes report **false positives**. Before including every issue:

- Check if the target actually runs the affected service/version.
- Verify using other tools or manual inspection.
- Mark verified and unverified findings separately in your project report.

This shows critical thinking and improves the credibility of your results.

5. Prioritize for Remediation

Rank issues based on:

1. **Severity (Critical first).**
2. **Exploitability (is a public exploit available?).**
3. **Impact (data loss, system compromise, etc.).**
4. **Exposure (public-facing vs. internal systems).**

You can then recommend an action plan focusing on the highest-priority items.

6. Summarize in Report

Your written analysis should include:

- **Total hosts scanned** and how many were reachable.
- **Number of vulnerabilities per severity level.**
- **Most critical issues** and their potential impact.
- **Recommended remediation strategies.**
- **Re-scan plan** to verify fixes later.

Include charts or tables if allowed — they make the data more readable.

7. Draw Conclusions

Finish by reflecting on:

- The **security posture** of the tested systems.
- **Common patterns** (e.g., outdated software, weak configurations).
- **Effectiveness of OpenVAS** as a tool for identifying risks.
- **Lessons learned** for improving vulnerability management in real networks.

2.3 Data Collection and Storage:-

1. Overview

OpenVAS, part of the Greenbone Vulnerability Management (GVM) framework, performs automated security assessments. Its data collection and storage process is the backbone that allows it to detect vulnerabilities, keep scan records, and generate detailed reports.
This process involves several components working together to gather, manage, and preserve information systematically.

2. Data Collection Process

a. Network Discovery

Before scanning for vulnerabilities, OpenVAS first identifies active hosts and open ports within the target network.
It collects basic system information such as:

- IP addresses and hostnames
- Operating systems and versions
- Active services and ports
- Banner information (e.g., web server version, SSH details)

This stage helps the scanner understand what targets exist and what technologies they run.

b. Service Enumeration

Once services are identified, OpenVAS probes deeper to collect application-level details.

For example:

- Web servers → framework, CMS type, version, headers
- Databases → version and authentication settings
- Network services → supported protocols and cipher suites

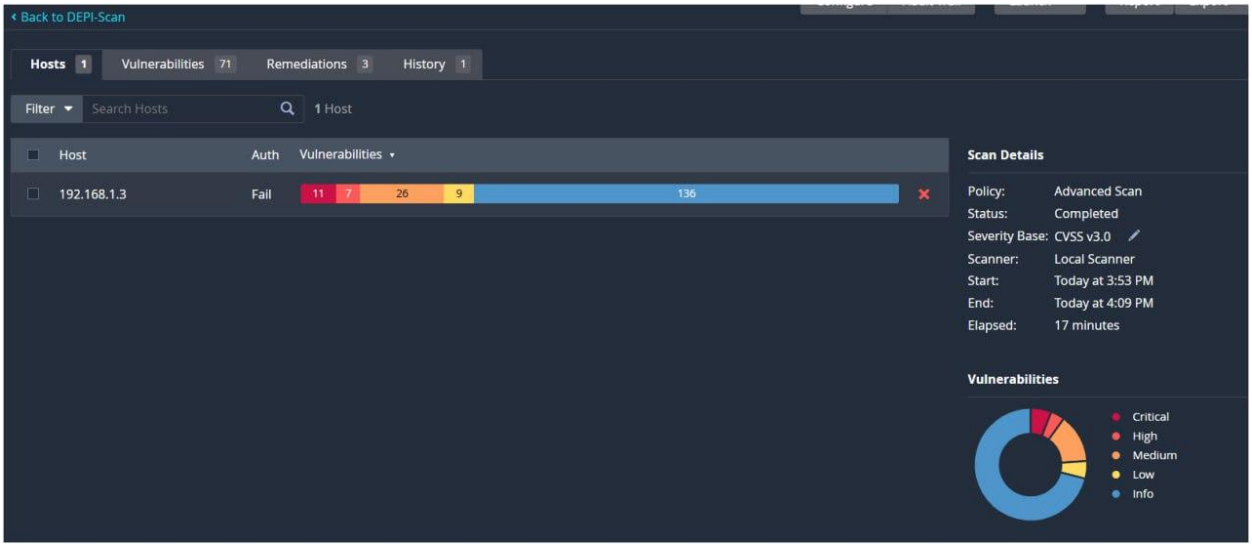
This layer of information becomes the basis for identifying potential vulnerabilities.

c. Vulnerability Detection

OpenVAS uses a library of Network Vulnerability Tests (NVTs) to analyze the collected data.
Each NVT corresponds to a known vulnerability or configuration issue. The scanner collects:

- Vulnerability IDs and descriptions
- CVE references (if available)
- Detected service versions
- Risk and impact assessments

These NVTs are updated regularly from the Greenbone Community Feed or Enterprise Feed, ensuring the scanner has the latest vulnerability definitions.



d. Scan Metadata

During the scan, OpenVAS also collects metadata such as:

- Scan start and end time
- Scanner version and profile used
- Target configuration details
- Network latency and response metrics

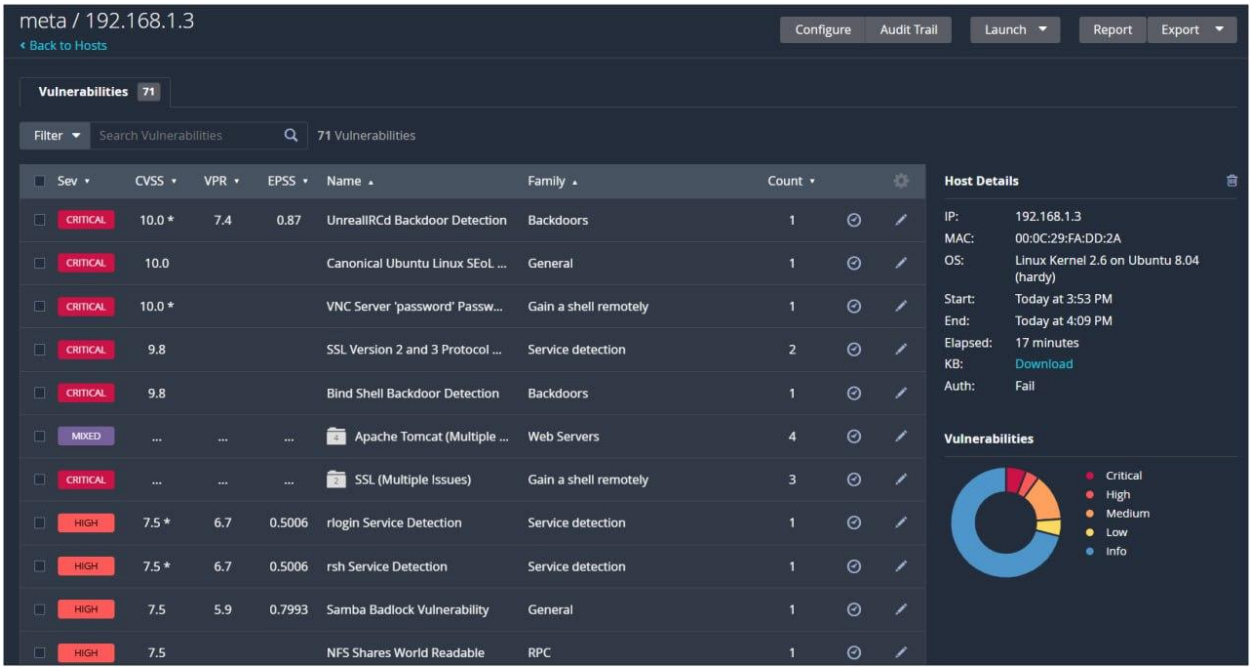
This metadata is important for reporting accuracy and audit trails.

3. Data Storage Architecture

```
┌──(aboodi㉿kali)-[~]
└─$ sudo systemctl status  nessusd
[sudo] password for aboodi:
Sorry, try again.
[sudo] password for aboodi:
● nessusd.service - The Nessus Vulnerability Scanner
     Loaded: loaded (/usr/lib/systemd/system/nessusd.service; enabled; preset: disabled)
     Active: active (running) since Tue 2025-10-28 09:56:28 EEST; 3 weeks 1 day ago
 Invocation: 627fafe294664a85af9480a1c9533bee
   Main PID: 60470 (nessus-service)
      Tasks: 21 (limit: 4498)
     Memory: 1.7G (peak: 2.7G, swap: 0B, swap peak: 169.1M)
        CPU: 1h 31min 33.150s
     CGroup: /system.slice/nessusd.service
             ├─ 60470 /opt/nessus/sbin/nessus-service -q
             └─100775 nessusd -q

Oct 28 09:56:28 kali systemd[1]: Started nessusd.service - The Nessus Vulnerability Scanner.
Oct 28 09:56:28 kali nessus-service[60470]: nessus-service [60470][INFO] : Nessus 19.15.0 [build 20152] Started
Nov 19 15:42:17 kali nessus-service[60470]: nessus-service [60470][INFO] : Service upgrade detected: starting upgrade.
Nov 19 15:42:17 kali nessus-service[60470]: nessus-service [60470][INFO] : Nessus 19.15.1 [build 20010] Started

┌──(aboodi㉿kali)-[~]
└─$
```



a. Central Database (GVMD)

The Greenbone Vulnerability Manager Daemon (GVMD) serves as the central data management system.
It stores all scan-related information in a PostgreSQL database, including:

- Target definitions and configurations
- Scan tasks and schedules
- Collected results and logs
- User accounts and permissions
- Generated reports and historical data

This database is the main storage location for OpenVAS data, ensuring that all collected information is organized and retrievable.

b. NVT Feed Storage

All the Network Vulnerability Tests (NVTs) are stored locally in a structured directory under the OpenVAS installation.
These are small scripts written in the NASL language, and each test file contains:

- Instructions for data collection
- Detection logic for specific vulnerabilities
- References to CVE, CPE, and CVSS data

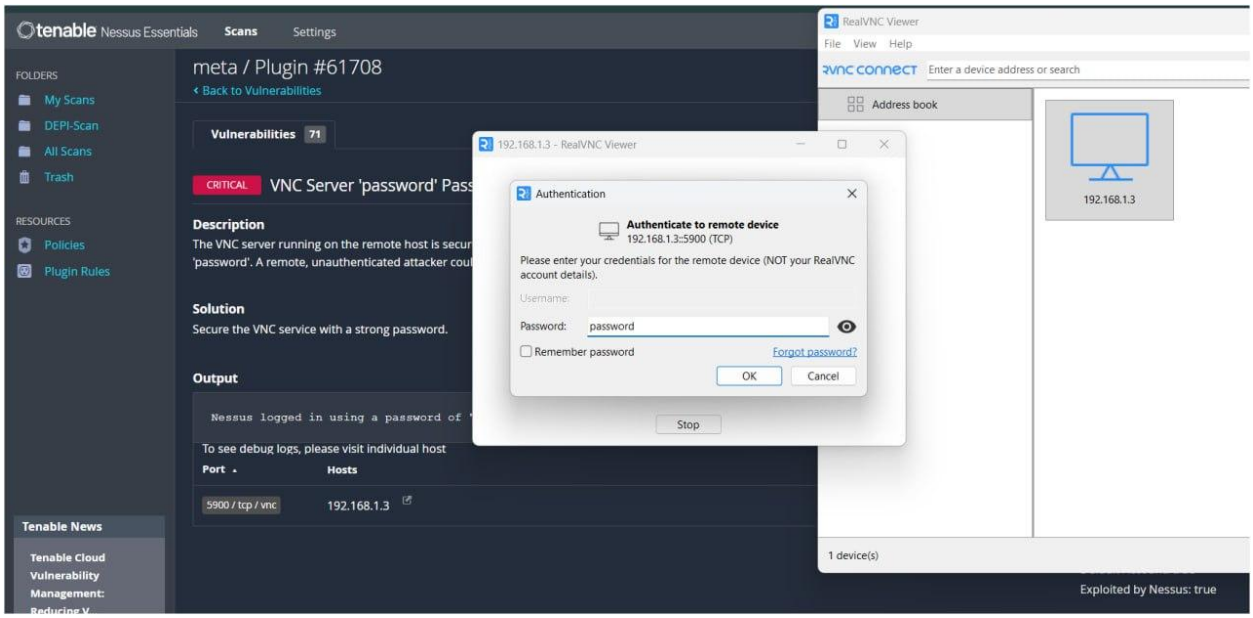
When updates occur, the feed synchronizes automatically and stores new or modified NVTs.

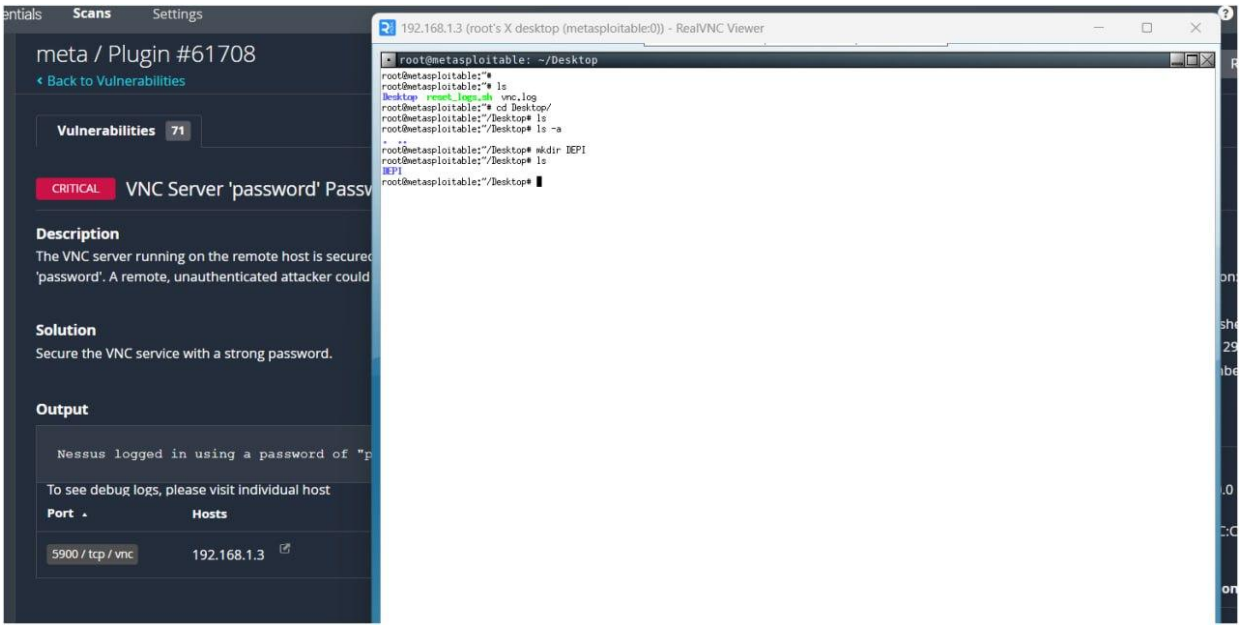
c. Report Storage

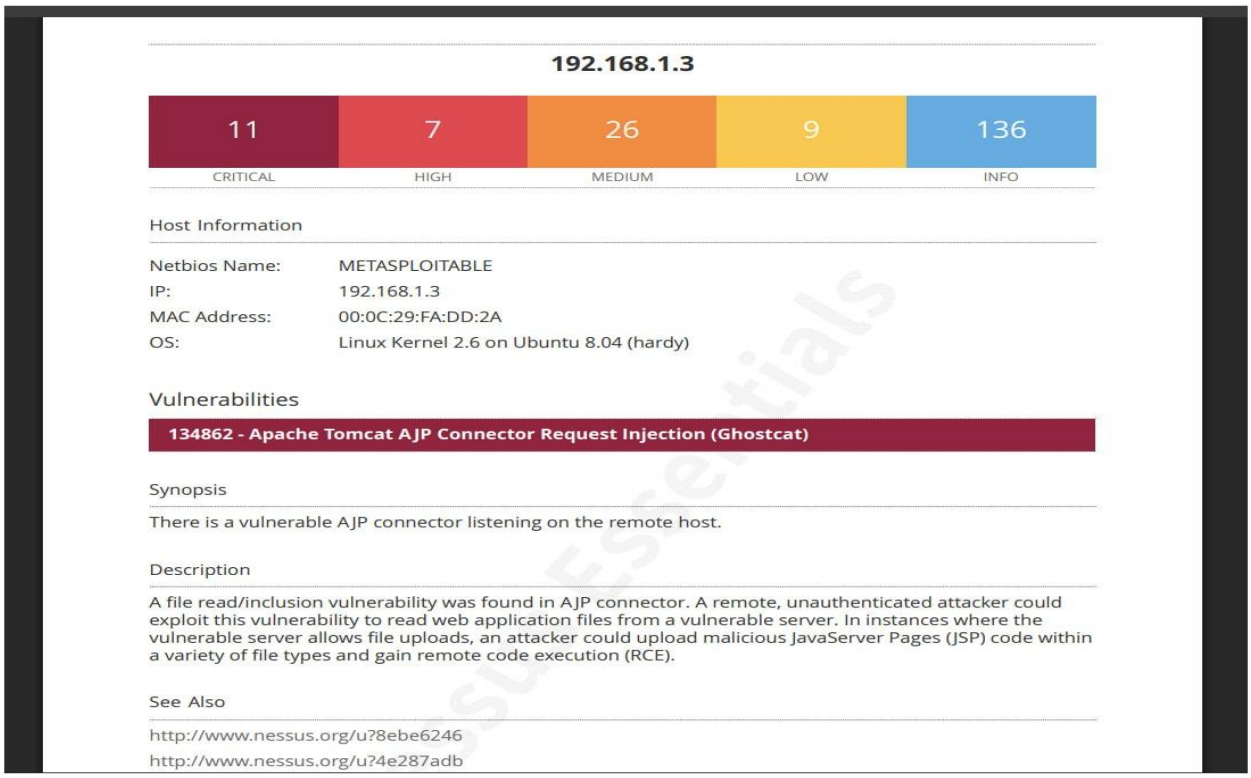
Once a scan is complete, reports are generated and saved in multiple formats within the GVM system.
Each report includes:

- Host summaries
- Vulnerability findings
- Severity scores
- Recommended remediations

Reports can also be exported to formats like PDF, XML, HTML, or CSV, which can then be stored externally for long-term documentation.


tenable Nessus Essentials
Scans
Settings
FOLDERS
My Scans
DEPI-Scan
All Scans
Trash
RESOURCES
Policies
Plugin Rules
meta / Plugin #61708
Back to Vulnerabilities
Vulnerabilities 71
CRITICAL
VNC Server 'password' Pass
Description
The VNC server running on the remote host is secur
'password'. A remote, unauthenticated attacker coul
Solution
Secure the VNC service with a strong password.
Output
Nessus logged in using a password of
To see debug logs, please visit individual host
Port
Hosts
5900 / tcp / vnc
192.168.1.3
Tenable News
Tenable Cloud Vulnerability Management:
192.168.1.3 - RealVNC Viewer
Authentication
Authenticate to remote device
192.168.1.3::5900 (TCP)
Please enter your credentials for the remote device (NOT your RealVNC account details).
Username:
Password:
password
Remember password
Forgot password?
OK
Cancel
Stop
RealVNC Viewer
File View Help
vnc connect
Enter a device address or search
Address book
192.168.1.3
1 device(s)
Exploited by Nessus: true


Scans
Settings
meta / Plugin #61708
Back to Vulnerabilities
Vulnerabilities 71
CRITICAL
VNC Server 'password' Passv
Description
The VNC server running on the remote host is secured
'password'. A remote, unauthenticated attacker could
Solution
Secure the VNC service with a strong password.
Output
Nessus logged in using a password of "p
To see debug logs, please visit individual host
Port
Hosts
5900 / tcp / vnc
192.168.1.3
192.168.1.3 (root's X desktop (metasploitable:0)) - RealVNC Viewer
root@metasploitable: ~/Desktop
root@metasploitable:"#
root@metasploitable:"# ls
Desktop reset_logs.sh vnc.log
root@metasploitable:"# cd Desktop/
root@metasploitable:"/Desktop# ls
root@metasploitable:"/Desktop# ls -a
. ..
root@metasploitable:"/Desktop# mkdir DEPI
root@metasploitable:"/Desktop# ls
DEPI
root@metasploitable:"/Desktop#



192.168.1.3

11	7	26	9	136
CRITICAL	HIGH	MEDIUM	LOW	INFO

Host Information

Netbios Name:	METASPLOITABLE
IP:	192.168.1.3
MAC Address:	00:0C:29:FA:DD:2A
OS:	Linux Kernel 2.6 on Ubuntu 8.04 (hardy)

Vulnerabilities

134862 - Apache Tomcat AJP Connector Request Injection (Ghostcat)

Synopsis

There is a vulnerable AJP connector listening on the remote host.

Description

A file read/inclusion vulnerability was found in AJP connector. A remote, unauthenticated attacker could exploit this vulnerability to read web application files from a vulnerable server. In instances where the vulnerable server allows file uploads, an attacker could upload malicious JavaServer Pages (JSP) code within a variety of file types and gain remote code execution (RCE).

See Also

http://www.nessus.org/u?8ebe6246
http://www.nessus.org/u?4e287adb

4. Data Integrity and Security

To maintain accuracy and confidentiality, OpenVAS:

- Uses database authentication and access control for users.
- Encrypts communication between components via TLS/SSL.
- Logs all activities (e.g., who ran which scans).
- Validates feed updates using digital signatures to ensure authenticity.

This ensures that collected data remains trustworthy and protected from tampering.

5. Data Lifecycle

The general lifecycle of OpenVAS data can be described as:

1. Acquisition: Gathering data from scans and targets.
2. Storage: Saving data in databases and local feed directories.
3. Analysis: Interpreting scan results for vulnerability detection.
4. Reporting: Generating structured, exportable reports.
5. Archival or Deletion: Keeping results for compliance or removing them after the project ends.

6. Application in a Project

In our project report, we explain:

- How OpenVAS systematically collects, processes, and stores vulnerability information.
- The role of the GVMD database in managing all scanning data.
- The importance of organized data storage for generating reliable and repeatable results.
- How this structured approach supports research, analysis, and security improvement.