4.0Vulnerability Scanning Automation + Web Reporting

4.1 System Overview:

The system automates the entire vulnerability assessment lifecycle. It performs network scanning on cloud-hosted virtual machines (VMs), identifies vulnerabilities using industry-standard scanners (OpenVAS and Nessus), processes scan results through a custom script, and displays the findings in a structured and interactive web interface.

4.2 System Components:

4.2.1Scanning Engine:

We used two scanning tools:

(1) OpenVAS

- Performs **full system vulnerability assessment**
- Provides results in **XML/JSON format**
- Detects outdated software, misconfigurations, weak services, etc.

(2) Nessus

- Detects **software vulnerabilities**, **weak passwords**, and **cloud misconfigurations**
- Provides detailed severity scores (CVSS)

Both scanners were installed on our cloud environment and configured with:

- Scan targets (Cloud VMs IPs)
- Scan profiles (Basic / Full / Authenticated)
- Scheduled scans
- Exporting results automatically to a shared directory

4.3 Automation Script:

We developed a Python-based script responsible for automating three main tasks:

4.3.1 Triggering the Scan

- The script sends API requests to OpenVAS/Nessus
- It starts full scans on predefined targets
- It polls (checks repeatedly) until the scan is completed

4.3.2 Parsing Scan Reports

After the scanner finishes:

- OpenVAS results exported in XML
- Nessus results exported in Nessus (XML-based)

The script reads these files and extracts:

- Vulnerability name
- Severity (Critical / High / Medium / Low)
- CVSS score
- Affected port and service
- Description of the vulnerability
- Recommended remediation to parse results.

4.3.3 Storing Results

The script then stores extracted data into a **database**, such as:

- SQLite (simple)
- MySQL (if production-level)

Each vulnerability is saved with:

- ID
- VM name / IP

- Plugin ID
- Severity
- Description
- Fix recommendation
- Timestamp

4.4 Web Interface:

We created a simple but structured web dashboard to display vulnerabilities.

4.4.1 Backend

We used either:

- Node.js (Express)

Backend functionality:

1. Retrieve vulnerability data from the database
2. Provide API endpoints like:
    - vulnerabilities
    - vulnerabilities/{id}
3. Send data to the frontend in JSON format

4.4.2 Frontend

Using HTML + CSS + JavaScript:

- A main dashboard page shows all vulnerabilities
- A filter (dropdown) allows filtering by **severity**, **VM**, or **date**
- Each vulnerability row includes:
    - Name
    - Severity (color-coded)
    - Affected VM
    - Port
    - CVSS Score
    - Recommended Fix

4.4.3 Additional Features:

- Search bar
- Sorting vulnerabilities
- Export to PDF / Excel
- Re-scan button (triggering scan job)

5.0 Vulnerability Types:

During our scanning, the system detected multiple categories:

5.1 Vulnerable Software

Examples of findings:

- Outdated Apache version
- Old OpenSSH server
- PHP vulnerabilities
- Unsupported Linux kernel Remediation Steps
- Update software to latest stable version
- Apply security patches from vendor
- Remove unused or deprecated services

5.2 Weak Passwords

Nessus/OpenVAS detected:

- Default passwords
- Simple passwords
- Services with no password policies

Remediation Steps

- Enforce password policy (complexity, length, rotation)
- Disable default accounts
- Enable MFA (Multi-Factor Authentication)

5.3 Cloud VM Misconfigurations (GCP)

Detected issues:

- Public VM exposure
- Open SSH/RDP to the internet
- Unrestricted firewall rules
- Missing IAM role restrictions

Remediation Steps

- Restrict firewall rules (allow only necessary ports)
- Remove public IP if not required
- Apply IAM least privilege
- Enable VPC Service Controls
- Use Google Cloud Security Command Center (SCC)

6.0 Workflow Explanation:

1) User clicks "Start Scan" button on the web page
2) Backend calls Python script
3) Script sends API requests → OpenVAS / Nessus
4) Scanner performs scan on Cloud VMs
5) Results exported to XML / JSON directory
6) Script parses vulnerabilities and saves them to the database
7) Frontend requests data from backend
8) Dashboard displays vulnerabilities in a clean UI

7. Summary

This system automates vulnerability scanning across cloud-based infrastructure. It integrates OpenVAS and Nessus, processes the results with a custom automation script, and provides a web-based dashboard for visualization.

The solution helps organizations quickly identify risks such as vulnerable software, weak passwords, and cloud misconfigurations (especially in GCP), and provides actionable remediation steps.