Reporting

1. Present reports to your clients/employers
2. Score the severity of your findings.

1. Overview of Reports in Penetration Testing

A report is not just about the look and feel. A good report will have the following criteria:

- Accurate vulnerabilities severity scoring (not exaggerating the severity of a vulnerability).
- No false positives.
- Evidence (e.g., screenshots, or PoC) and not just links or definitions.
- Instructions for how to remediate the flaw. This is where a security professional will shine.
- A clear definition of how to fix the issue is a turning point in your reports. (I've seen a lot of reports where the remediation part is just a link to OWASP, a CVE reference, etc.)
- Be clear and not too wordy.
- Must be divided into two reports:
  - A technical report that comprises all the evidence and details.
  - A summary report addressed to management (executive summary) that shows the real vulnerabilities and outcomes of the security tests.
  - Well-formatted and concise so the reader will enjoy reading it. Generally, companies have their own template to create reports. Feel free to step out of your comfort zone and change it depending on your test's style.

2. Severities

Example:

During a penetration test, you found a SQL injection using Burp Suite. By default, the severity is High for a SQL injection. So, you escalate it to upper management. But, digging further (going beyond the severity level), the application tested contains no confidential information, and it's only accessible from the intranet (it's not visible to the internet).

In summary, multiple factors changed the game. Let's see together how to fix this kind of issue by using the CVSS scoring system.

Common Vulnerability Scoring System:

What is CVSS?

It is a scoring system that considers multiple factors and calculates the score (e.g., low, medium, high, or critical) to estimate your findings in a better, more accurate way.

CVSS version 3.1 uses the following factors to calculate the score of a flaw/vulnerability found during a pentest:

1. **Attack vector (AV):** This score evaluates how the attacker can exploit a vulnerability, e.g., remotely versus locally.
    - **Network (N):** This option is selected when the vulnerability is accessible from the internet.
    - **Adjacent (A):** This option is selected when the exploit can be successful in the same intranet network (not accessible from the internet).
    - **Local (L):** This option is selected when the network stack is excluded from the choices, for example, opening a document (through social engineering), a keylogger, etc.
    - **Physical (P):** This option is selected when the exploitation is done physically on the host (e.g., the attacker has to enter the server room).

2. **Attack complexity (AC):** This score evaluates how difficult it is to exploit a vulnerability.
    - **High (H):** This option is selected when the exploit is hard to execute.
    - **Low (L):** This option is selected when it's easy to exploit the flaw.

3. **Privilege required (PR):** This score will mention the level of privilege needed to exploit a vulnerability.
    - **None (N):** The attacker will not need any credentials to access the flaw.
    - **Low (L):** The attacker will need a limited account to access the flaw.
    - **High (H):** The attacker will need a high privilege account to access the flaw.
4. **User interaction (UI):** This score evaluates if a user (the victim) is required or not to interact with the flaw (e.g., through social engineering).
    - **None (N):** The flaw can be exploited without the victim interaction.
    - **Required (R):** The flaw involves some sort of victim interaction.

5. **Scope (S):** This metric will evaluate if the flaw impacts another system with a different security scope. For example, if the defect is exploited in the testing environment, then it will allow the attacker to exploit the production network.
    - **Unchanged (U):** This option is picked when the affected resources are in the same security scope.
    - **Changed (C):** This option is selected when other security scope assets will be impacted by this flaw.

6. **Confidentiality (C):** This metric will evaluate if the attacker will be able to read confidential data.

- **High (H):** Use this if the attacker can read the whole confidential data that an organization has on premises.
- **Low (L):** Use this if the attacker can read some part of the confidential data that an organization has on premises.
- **None (N):** Use this if the attacker will not be able to read any confidential data when the flaw is exploited.

7. **Integrity (I):** This metric will evaluate if the attacker will be able to write data into the exploited system (e.g., adding or altering records inside the database).
    - **High (H):** This option is selected when the attacker has full write access if they exploit the vulnerability.
    - **Low (L):** This option is selected when the attacker has some limited write access if they exploit the vulnerability.
    - **None (N):** This option is selected when the attacker has no write per□missions if the flaw is exploited.
8. **Availability (A):** This metric evaluates if the exploitation will affect the availability of the exploited assets (servers, routers, laptops, Wi-Fi AP, etc.).
    - **High (H):** This option is selected if the exploitation will greatly impact the availability of the exploited asset.
    - **Low (L):** This option is selected if the exploitation will have some sort of impact on the availability of the exploited asset.
    - **None (N):** This option is selected if the exploitation has no impact on the availability of the exploited asset.

Based on the previous example of SQL injection (where you were happy to find it), if you visit the CVSS calculator, your score should look like Figure 1

Table: CVSS Score rating

RATING	CVSS 3.1 SCORE
None	0
Low	0.1–3.9
Medium	4.0–6.9
High	7.0–8.9
Critical	9.0–10.0

Report Presentation

In general, four inputs are feeding our reports:

- Infrastructure flaws found by a scanner (e.g., OpenVAS, Nessus, Qualys, etc.) or manual findings.
- Web application flaws found by a scanner (e.g., Burp Suite) or manual findings.
- Source code flaws found by a static application security testing (SAST) tool (e.g., Checkmarx, Veracode, etc.) or findings using a manual code review.
- Open-source library flaws found by a library scanner (e.g., Nexus Sonatype lifecycle).

1. Cover Page

On this page, you should show the title of your report (e.g., “Penetration Test Report” or “Cybersecurity Report”). Make sure to include the following items:

- Company name
- Project name tested.
- The name of the person in charge (penetration tester; sometimes you have to include the name of your company, if you are hired by a third party)
- Report title

2. History Logs

Sometimes there are multiple people working on the same report. Also, there is always the possibility that the contents will change, so a history log will help keep track of the efforts done. A history log should contain the following items:

- The version number.
- The date of the change
- The name of the person who modified it.
- A short description of what the change is all about

3. Report Summary

This section is the first part that will summarize the contents of the vulnerabilities found during your engagements. First, take your time to explain all the tasks executed to get the work done. Then, explain the scoring that you used to evaluate the severity of each flaw. For the visuals, a professional report summary will contain the following items:

- Number of flaws based on their severity (graph charts are nice to have)
- Number of flaw categories and occurrences (e.g., five occurrences of reflected XSS flaws, two occurrences of SQL injection, etc.)