Task 7

1.Understand OWASP Top 10 vulnerabilities.

ID	Vulnerability	Simple Explanation	Technical Detail (For the Report)
A01	**Broken Access Control**	Like a hotel where your key opens every door.	Users can view or modify data they don't own. **SSRF** (A10:2021) has now been merged into this category.
A02	**Security Misconfiguration**	Leaving the default "admin/admin" login on a router.	**Jumped to #2.** Includes unpatched software, unnecessary open ports (like Telnet on Metasploitable), and weak cloud permissions.
A03	**Software Supply Chain Failures**	Using a "free" car part that turns out to be broken.	**Broadened for 2025.** Covers risks from 3rd-party libraries, insecure CI/CD pipelines, and untrusted software updates.
A04	**Cryptographic Failures**	Sending a secret message on a postcard.	Lack of encryption for sensitive data (PII), using weak algorithms like MD5/SHA1, or missing TLS (HTTPS).
A05	**Injection**	A soda order that tells the waiter to empty the register.	Untrusted data is executed as a command. Includes **SQL Injection**, **Command Injection**, and **XSS**.
A06	**Insecure Design**	Building a vault with a screen door on the back.	Flaws in the app's logic or architecture (e.g., a "forgot password" flow that is too easy to guess).
A07	**Authentication Failures**	A club that doesn't check IDs properly.	Weak passwords, lack of Multi-Factor Authentication (MFA), or session IDs that don't expire after logout.
A08	**Software/Data Integrity Failures**	Accepting a delivery with a broken seal.	Failure to verify the source or integrity of data/updates. Includes "Insecure Deserialization."
A09	**Logging & Alerting Failures**	A burglar breaks in, but the alarm is turned off.	Name changed to include **"Alerting."** It’s not enough to just keep logs; you must have

ID	Vulnerability	Simple Explanation	Technical Detail (For the Report)
			an alert system to stop the attack.
A10	**Mishandling of Exceptions**	**NEW.** The app crashes and reveals the database password in the error message.	Focuses on resilience and "failing safely." Poor error handling can leak info or cause a Denial of Service (DoS).

---

2. Burp Suite

as a "Man-in-the-Middle" that sits between your web browser and the target server. Usually, when you click a button, the request goes straight to the server. With Burp, you catch that request in mid-air, read it, change it, and then decide whether to let it pass or "drop" it.

1. The "Big Three" Tools (Simple & Detailed)

You will use these three tabs 90% of the time in your lab.

A. The Proxy (The Interceptor)

- **Simple:** A "Pause" button for the internet.
- **Detailed:** It captures HTTP/S requests. You can turn "Intercept ON," click a button on a site, and see the raw text—including cookies, headers, and hidden fields—before the server sees it.
- **Internship Use:** Capturing a login request to see if the password is being sent in plain text or if there are hidden "admin=false" flags you can flip.

B. The Repeater (The Lab)

- **Simple:** A "Redo" button for testing different ideas quickly.
- **Detailed:** Once you catch a request in the Proxy, you "Send to Repeater." Here, you can change one small thing (like a User ID or a price) and hit "Send" over and over to see how the server reacts without re-typing everything in your browser.
- **Internship Use:** Testing for **A01: Broken Access Control**. Change `User_ID=100` to `User_ID=1` in a request, hit Send, and see if the server returns the Admin's private data.

C. The Intruder (The Automator)

- **Simple:** A "Rapid Fire" machine gun for attacks.
- **Detailed:** It takes a single request and runs it hundreds of times with different "payloads" (variables). It is used for fuzzing and brute-forcing.

- **Internship Use: Brute Forcing** a login. You give it a list of 1,000 common passwords, and Intruder tries them all in seconds to see which one returns a "200 OK" instead of a "401 Unauthorized."

3.What is DVWA?

- **Simple:** It is a "practice range" for hackers. It is a real web application built with PHP and MySQL that is intentionally filled with the biggest security holes on the internet.
- **Detailed:** It is an open-source PHP/MySQL-based training platform used by penetration testers to practice identifying and exploiting vulnerabilities. It mimics a real-world environment where security controls are either missing or poorly implemented.

4. The Unique "Security Level" System

This is the most important feature to explain in your report. DVWA allows you to toggle the "strength" of the website's armor:

Level	Security Status	Goal for your Internship
Low	**No Security.** Input is trusted blindly.	Use this to understand the "Pure" version of an exploit (e.g., basic SQLi).
Medium	**Poor Security.** Developers tried to filter input but did it poorly.	Practice "Bypass" techniques. Use Burp Suite to change data in transit.
High	**Strict Security.** Very hard to exploit; filters almost everything.	For advanced testing. Often requires finding "logical" flaws or very specific payloads.
Impossible	**Secure Code.** The app is fixed and safe.	**Compare this code** to the "Low" version to write your "Remediation" report.

---

5. Key Vulnerabilities to Demonstrate

For your submission, pick these "Big Three" as they are the easiest to document with screenshots:

A. SQL Injection (SQLi)

- **The Bug:** The app takes your input and injects it directly into the database command.
- **Proof (Low):** Type `' OR 1=1 #` into the User ID box.
- **Result:** The database is tricked into returning every user in the system.

B. Command Injection

- **The Bug:** The app takes an IP address to "ping" it but allows you to run other system commands too.
- **Proof (Low):** Type `127.0.0.1; whoami`
- **Result:** You get back the name of the server's user (e.g., `www-data`), proving you can run code on the operating system.

C. Cross-Site Scripting (XSS)

- **The Bug:** The app displays whatever text you type without checking it.
- **Proof (Low):** Type `<script>alert('Hacked')</script>` in the name field.
- **Result:** A pop-up appears in the browser, proving you can run malicious JavaScript on other people's screens.

---

6. The "View Source" Feature (Pro Tip)

Inside DVWA, there is a button at the bottom of every page called **"View Source."** * In your report, include side-by-side screenshots of the **Low** source code (vulnerable) and the **Impossible** source code (secure).

- Explain that the secure version uses **"Parameterized Queries"** or **"Input Sanitization."** This shows your internship supervisor that you understand the **Fix**, not just the **Hack**.

---

7. SQL Injection (SQLi)

- **Target:** The **Database** (Server-side).
- **Simple Analogy:** Imagine a bank teller who follows *any* written instruction. You give them a check that says: "Pay me $100... **and also show me the vault's master password.**" If the teller does both, that is an Injection.
- **Detailed Explanation:** SQLi occurs when an application takes user input (from a login box, URL, or search bar) and inserts it directly into a database query. Because the input isn't "cleaned," the database treats the attacker's data as a **command**.

8. Cross-Site Scripting (XSS)

- **Target:** The **End User** (Client-side / Browser).
- **Simple Analogy:** Like leaving a sticky note on a public bulletin board that says "Click here for free cookies," but when someone clicks it, it secretly steals their wallet.
- **Detailed Explanation:** XSS happens when an attacker "injects" malicious JavaScript into a website. When a victim visits that site, their own browser executes the script,

thinking it’s a legitimate part of the page.