Open Web Application Security Project (OWASP) Vulnerabilities & WAF

S7 / S5 B.Tech: Computer Security Lab

Due Date: 09 September 2021 10:00 PM

1 Introduction

A vulnerability is a hole or a weakness in the application, which can be a design flaw or an implementation bug, that allows an attacker to cause harm to the stakeholders of an application. Stakeholders include the application owner, application users, and other entities that rely on the application. The OWASP Top 10 is a standard awareness document for developers and web application security. The OWASP Top 10 list consists of the 10 most seen application vulnerabilities as listed below.

- **Injection:** Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
- **Broken Authentication:** Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
- **Sensitive Data Exposure:** Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
- **XML External Entities (XXE):** Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

- **Broken Access Control:** Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
- **Security Misconfiguration:** Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.
- **Cross-Site Scripting XSS:** XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
- **Insecure Deserialization:** Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.
- **Using Components with Known Vulnerabilities:** Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
- **Insufficient Logging & Monitoring:** Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

2 DVWA (Damn Vulnerable Web Application)

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficulty, with a simple straightforward interface. There are both documented and undocumented vulnerabilities with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

Damn Vulnerable Web Application is damn vulnerable! Do not upload it to your hosting provider's public html folder or any Internet facing servers, as they will be compromised. It is recommended using a virtual machine, which is set to NAT networking mode. DVWA could be installed as a docker container or as VM in Windows / Linux or directly from Kali Linux as VM. Please download the source code of DVWA from https://github.com/ethicalhack3r/DVWA

2.1 Attacks Covered in DVWA

- Brute Force
- Command Execution
- CSRF
- File Inclusion
- SQL Injection
- SQL Injection (Blind)
- Shell Uploading
- XSS (Reflected)
- XSS (Stored)

You are suppose to load DVWA as VM, and check all attacks documented as well as non-documented ones. Extra credits would be given for indicating the non-documented attacks.

2.2 Web Application Firewall (WAF)

A Web Application Firewall (WAF) is an application firewall for HTTP applications. It applies a set of rules to an HTTP conversation. Generally, these rules cover common attacks such as Cross-site Scripting (XSS) and SQL Injection. While proxies generally protect clients, WAFs protect servers. A WAF is deployed to protect a specific web application or set of web applications. A WAF operates through a set of rules often called policies. These policies aim to protect against vulnerabilities in the application by filtering out malicious traffic.

2.2.1 PHPIDS

PHPIDS (PHP-Intrusion Detection System) is a simple to use, well structured, fast and state-of-the-art security layer for your PHP based web application. The IDS neither strips, sanitizes nor filters any malicious input, it simply recognizes when an attacker tries to break your site and reacts in exactly the way you want it to. Based on a set of approved and heavily tested filter rules any attack is given a numerical impact rating which makes it easy to decide what kind of action should follow the hacking attempt. This could range from simple logging to sending out an emergency mail to the development team, displaying a warning message for the attacker or even ending the user's session. PHPIDS enables you to see who's attacking your site and how.

You are requested to enable PHPIDS in DVWA, and have to demonstrate us with all the attacks in different severity levels like "low", "medium", "high" and how it is getting detected by PHPIDS

2.2.2 ModSecurity

ModSecurity is an open source, cross-platform web application firewall (WAF) module. Known as the "Swiss Army Knife" of WAFs. It enables web application defenders to gain visibility into HTTP(S) traffic and provides a power rules language and API to implement advanced protections.

You are suppose to demonstrate ModSecurity by logging in the attacks and demonstrating its prevention capabilities.

3 Readings

1. SQL Injection Walkthrough.
2. Assignment Five: Web Security (Group Assignment).
3. OWASP Top Ten
4. OWASP Top 10 Security Risks & Vulnerabilities
5. OWASP Top 10 -2017 The Ten Most Critical Web Application Security Risks