OWASP TOP 10 APPLICATION SECURITY RISKS - 2017

What is OWASP?

The **Open Web Application Security Project** known as **OWASP** is an international non-profit organization dedicated to web application security, OWASP has many community driven open source software projects and materials available online at “https://owasp.org/”. Materials include documentation, tools, videos and forums. One of the famous projects is the OWASP Top 10. (CLOUDFLARE, 2020) (OWASP, Who is the OWASP Foundation?, 2020)

What is the OWASP Top 10?

The **OWASP Top 10** is a report that outlines the type of security concerns for web application security focusing on the 10 most critical risks. Next, the report is compiled by a team of security experts globally. OWASP refers to the Top 10 as “awareness document” and they recommend that all companies to incorporate the report into the business implementations to minimize and/or mitigate security risks. (CLOUDFLARE, 2020)

Figure 1 below displays the type of Security risk reported in the OWASP Top 10 2017 report and rank from A1 to A10.

Figure 1: OWASP Top 10 Application Security Risks (2017).

A1: Injection

Injection attacks occur when untrusted data is sent to a code interpreter through a form input or some other data submission to a web application. For example, an attacker injects SQL database code onto a website login form that expects a plaintext username. If that form is not secured, it would result in that SQL code being executed. This process is known as **SQL injection attack.**

Injection attacks can be prevented through validating and sanitizing user-submitted data. (Validation of data refers to rejecting suspicious-looking data, and sanitizing data refers to cleaning up the suspicious-looking parts of the data). An example of preventing SQL injection in MySQL database is to use "mysql_real_escape_string()" to ensure that any dangerous characters such as ' are not passed to a SQL query in data. (Rubens, 2018)

A2: Broken Authentication

Login systems that have vulnerabilities provides a major risk as attackers are able to access user accounts and compromise the entire system through utilizing an admin account. For example, an attacker saves a list containing various known username and passwords during a data breach and use a script to try all the combinations on another login system to gain access. (CLOUDFLARE, 2020) An example of a password list is rockyou.txt which is a set of compromised passwords obtained from a social media application developer RockYou. (Adam, 2017)

Various strategies can be employed to mitigate authentication vulnerabilities such as 2-factor authentication (2FA) which requires user to provide authentication code generated and received from a different device, and limiting the number of login attempts a user can perform. (CLOUDFLARE, 2020)

A3: Sensitive Data Exposure

Web applications that do not protect sensitive data such as financial, healthcare and identity properly enable attackers to steal or modify weakly protected data through attacks such as "**Man-in-the-middle-attack (MitM)**".

Data exposure risk can be minimized by encrypting all sensitive data and disabling the caching of any sensitive information. Next, web developers had to take into consideration not to store any sensitive data when designing web applications. (CLOUDFLARE, 2020) (OWASP, OWASP Top Ten, 2017)

A4: XML External Entities (XXE)

XML External Entities (XXE) is a type of attack that takes advantage of the XML parsers in a web application that might process and execute some payload included as an external reference in the XML document. If an attacker adds or modifies entities in an XML file and points them to a malicious source, this could cause a **Denial of Service (DoS)** attack or an **SSRF** attack. (What Is OWASP? What Are the OWASP Top 10 Vulnerabilities?, 2019)

The code below is an example of XXE attack from sectigostore.com that obtained the reference from OWASP where the attacker attempts to extract data from the server:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```

Steps to prevent XXE:

- Implement server-side input validation, sanitization checks, etc. to prevent hostile data within XML documents.
- Disable XML external entity and DTD processing.
- Use timeouts, and test any place where uploads are made.
- Use less complicated formats like JSON, avoid serialization of sensitive data, and patch all XML processors and libraries

A5: Broken Access Control

Access control refers to a system that controls access to information or functionality. Broken access controls enables attackers to bypass authorization and perform tasks as privileged users such as administrators. For example, a web application allows a user to change into another account through changing sections of the url without verification.

Broken access control can be prevented through the web application utilizing authorization tokens. The authorization token is generated upon users logging in. Performing a privileged request will require authorization tokens to be present and restrict normal users from gaining privileged user controls. (CLOUDFLARE, 2020)

A6: Security Misconfiguration

Security misconfiguration is the most common vulnerability and is due to the application using default configurations or displaying excessively verbose errors. For example, an application displays descriptive errors to the user which may reveal vulnerabilities in the application. This can be mitigated through removing any unused features in the code and ensure error messages are displayed generally without specific info. (OWASP, OWASP Top Ten, 2017)

A7: Cross-Site Scripting (XSS)

Cross-site scripting vulnerabilities occur when web applications allow users to add custom code into a url path or onto a website that will be seen by other users. This vulnerability can be exploited to run malicious JavaScript code on a victim's browser. For example, an attacker sends an email to a victim that appears to be from a trusted site with a link to the website. The link could contain malicious JavaScript code tagged onto the end of the url. If the website is not configured or properly protected against cross-site scripting, the malicious code will be executed in the victim's browser when the user clicks on the link.

Mitigation strategies for cross-site scripting involve implementing data escaping techniques, applying context-sensitive encoding and enabling security policy (CSP). Using modern web development frameworks such as ReactJS and Ruby on Rails provides built-in cross-site scripting protection. (CLOUDFLARE, 2020) (What Is OWASP? What Are the OWASP Top 10 Vulnerabilities?, 2019)

A8: Insecure Deserialization

The concept of serialization is converting an object from the application code into stream of bytes or into a format to be used for other purposes such as sending over the wire or storing it on a disk. Deserialization is the opposite process of serialization by converting serialized data back into objects usable by the application. (What Is OWASP? What Are the OWASP Top 10 Vulnerabilities?, 2019)

An insecure deserialization exploit is the result of deserializing data from untrusted sources and may result in serious consequences such as **DDoS** attacks and **remote code execution** attacks. Mitigation steps to prevent this vulnerability is to prohibit desrialization of data from untrusted sources, monitoring deserialization and implementing type checks. (CLOUDFLARE, 2020)

A9: Using Components with Known Vulnerabilities

This vulnerability refers to the usage of libraries to implement a specific functionality without first verifying the legitimacy or without using updated versions of those components. The exploitability scores varies depending on the component used and what type of vulnerability is present. Attackers will search for security flaws in components that are widely used and exploit the same vulnerability across different websites.

This problem can be fixed through using components from official sources, removing unused dependencies and features, monitor for version updates and patches for both client

and server-side, and continuously search for CVE to identify vulnerabilities in the components. (What Is OWASP? What Are the OWASP Top 10 Vulnerabilities?, 2019)

A10: Insufficient Logging & Monitoring

Organizations have to take effective measures to log events in order to detect data breaches. According to https://sectigostore.com/ where the average discovery time for a security breach is more than 6 months after the breach has happened. The long period of time taken to discover a data breach enables attacker to perform various attacks. Organizations have to log events such as repeated failed login attempts from the same IP and constantly monitor for suspicious activities and reporting it to incident response (IR) team to prevent the issue from escalating.

This vulnerability risk can be handled through ensuring all suspicious activities (failed logins, access control failures, input validation failures, etc) are logged to identify malicious accounts and IP addresses. Next, organizations have to maintain a detailed audit trails for important transactions to prevent tampering or deletion. Furthermore, organizations have to establish an incident response and recovery plan. (What Is OWASP? What Are the OWASP Top 10 Vulnerabilities?, 2019)

Bibliography

What Is OWASP? What Are the OWASP Top 10 Vulnerabilities? (10 October, 2019). Retrieved 9 February, 2020, from INFOSEC INSIGHTS by SECTIGO Store: https://sectigostore.com/blog/what-is-owasp-what-are-the-owasp-top-10-vulnerabilities/

Adam. (24 December, 2017). *Cracking everything with John the Ripper*. Retrieved 9 February, 2020, from https://bytesoverbombs.io: https://bytesoverbombs.io/cracking-everything-with-john-the-ripper-d434f0f6dc1c

CLOUDFLARE. (2020). *What is OWASP? What Are The OWASP Top 10?* Retrieved 9 February , 2020, from cloudflare: https://www.cloudflare.com/learning/security/threats/owasp-top-10/

OWASP. (2017). *OWASP Top Ten*. Retrieved 9 February, 2020, from OWASP: https://owasp.org/www-project-top-ten/

OWASP. (2020). *Who is the OWASP Foundation?* Retrieved 9 February, 2020, from OWASP: https://owasp.org/

Rubens, P. (2 May, 2018). *How to Prevent SQL Injection Attacks*. Retrieved 9 February, 2020, from eSecurity Planet: https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks.html