***What Is OWASP?***

*OWASP, or the Open Web Application Security Project, is a nonprofit organization focused on software security. One of their projects is the maintenance of the OWASP Top 10, a list of the top 10 security risks faced by web applications.*

***A Guide to OWASP Top 10 Testing***

*Testing for OWASP vulnerabilities is a crucial part of secure application development.*

* *Build security into your development process, rather than making it an afterthought*
* *Test your code against security standards repeatedly throughout development*
* *Use IDE and CI Pipeline integrations to automate testing*
* *Identify known vulnerabilities in third-party code to ensure your program does not rely on insecure dependencies*

*OWASP Top 10 Vulnerabilities –*

***A01. Broken Access Control*** *- If authentication and access restriction are not properly implemented, it's easy for attackers to take whatever they want. Penetration testing can detect missing authentication but cannot determine the misconfigurations that lead to the exposure. One of the benefits of the increasing use of Infrastructure as Code (IaC) tools is the ability to use scanning tools to detect configuration errors leading to access control failures.*

***A02: Cryptographic Failures*** *- Common errors such as using hardcoded passwords, outdated cryptographic algorithms, or weak cryptographic keys can result in the exposure of sensitive data. Scanning images for hardcoded secrets, and ensuring that data is properly encrypted at rest and in transit can help mitigate exposing sensitive data to attackers.*

***A03: Injection*** *- Injection attacks occur when attackers exploit vulnerabilities in web applications that accept untrusted data. Common types include SQL injection and OS command injection. This category now also includes Cross Site Scripting (XSS). By inserting malicious code into input fields, attackers can execute unauthorized commands, access sensitive databases, and potentially gain control over systems.*

*Application security testing can reveal injection flaws and suggest remediation techniques such as stripping special characters from user input or writing parameterized SQL queries.*

***A04: Insecure Design*** *- Insecure design focusses on fundamental design flaws and ineffective controls as opposed to weak or flawed implementations. Creating secure designs and secure software development lifecycles requires a combination of culture, methodologies and tools. Developer training, robust threat modelling, and an organizational library of secure design patterns should all be implemented to reduce the risks.*

***A05: Security Misconfiguration*** *- Application servers, frameworks, and cloud infrastructure are highly configurable, and security misconfigurations such as too broad permissions, insecure default values left unchanged, or too revealing error messages can provide attackers easy paths to compromise applications. To reduce misconfiguration risks organizations should routinely harden deployed application and infrastructure configurations and should scan all infrastructure as code components as part of a secure SDLC.*

***A06: Vulnerable and Outdated Components*** *- Modern applications are built using a large number of third-party libraries (which themselves are dependent on other libraries), and frequently run on open-source frameworks. As might be expected with any software, vulnerabilities in libraries and components will routinely be discovered, patched, and new versions released. A critical mitigation step is to build a Software Bill of Materials (SBoM) for all the software deployed or supplied to customers.* ***Veracode Software Composition analysis*** *and* ***Container Scanning*** *tools can produce SBoMs in standardized formats to give organizations a view of their exposure to vulnerabilities in third-party components.*

***A07 Identification and Authentication Failures*** *- Identifying and authorizing users and non-human clients is a fundamental security practice. While mitigation starts with secure coding practices, tools to detect and prevent credential stuffing and brute force attacks are also useful protections.*

***A08: Software and Data Integrity Failures*** *- The tools used to build, manage, or deploy software are increasingly common vectors of attack. A CI’CD pipeline can also be used to inject malicious code (or libraires), create insecure deployments, or steal secrets.* *Organizations can mitigate this threat by ensuring both the security of the build process, and the components pulled into the build. Adding in code scanning and software component analysis steps into a software build pipeline can identify malicious code or libraries.*

***A09: Security Logging and Monitoring Failures*** *- Having adequate logging and monitoring in place is essential in both detecting a breach early and in incident forensics to establish the scope of the breach, and to determine the method of compromise. Organizations must have adequate collection, storage, alerting and escalation processes. Using Dynamic Application Security Testing (DAST) tools like Veracode DAST, should produce significant logging and alerting events.*

***A10: Server-Side Request Forgery (SSRF)*** *- Modern web applications commonly fetch additional content or data from a remote resource. If an attacker can influence the destination resource, and the application does not validate the supplied URL, then a crafted request may be sent to a target destination.*

*Mitigating SSRF attacks is done using familiar methods such as sanitizing user input, using explicit allow lists, and inspecting request responses before they are returned to clients.*

| **HTTP** | **HTTPS** |
| --- | --- |
| HTTP stands for HyperText Transfer Protocol. In HTTP, the URL begins with “http://”. | HTTPS stands for HyperText Transfer Protocol Secure. In HTTPS, the URL starts with “https://”. |
| HTTP uses port number 80 for communication. | HTTPS uses port number 443 for communication. |
| Hyper-text exchanged using HTTP goes as plain text i.e. anyone between the browser and server can read it relatively easily if one intercepts this exchange of data and due to which it is Insecure. | HTTPS is considered to be secure but at the cost of processing time because Web Server and Web Browser need to exchange encryption keys using Certificates before actual data can be transferred. |
| HTTP Works at the [Application Layer](https://www.geeksforgeeks.org/application-layer-in-osi-model/). | HTTPS works at [Transport Layer](https://www.geeksforgeeks.org/transport-layer-responsibilities/). |
| HTTP does not use encryption, which results in low security in comparison to HTTPS. | HTTPS uses Encryption which results in better security than HTTP. |
| HTTP speed is faster than HTTPS. | HTTPS speed is slower than HTTP. |
| HTTP does not use data hashtags to secure data. | HTTPS will have the data before sending it and returning it to its original state on the receiver side. |
| HTTP is used to transfer text, video, and images via web pages. | HTTPS is used to transfer data securely via a network. |

*FAQs*

*1. Which is better: HTTP or HTTPS?*

*HTTPS is better than HTTP because HTTPS provides security. HTTP can also be hacked where as HTTPS cannot be hacked. HTTP does not help in search ranking whereas HTTPS helps in search ranking.*

*2. Whose performance is better: HTTP or HTTPS?*

*The speed of HTTP is faster than the speed of HTTPS. Due to the presence of SSL Protocol in HTTPS, the webpages become slower than HTTP.*