OWASP Top Ten

OWASP, or the Open Web Application Security Project, is a nonprofit organization focused on software security. Their projects include a number of open-source software development programs and toolkits, local chapters and conferences, among other things. One of their projects is the maintenance of the OWASP Top 10, a list of the top 10 security risks faced by web applications.

OWASP Top 10 Vulnerabilities

So, what are the top 10 risks according to OWASP? We break down each item, its risk level, how to test for them, and how to resolve each.

01. Broken Access Control

If authentication and access restriction are not properly implemented, it's easy for attackers to take whatever they want. With broken access control flaws, unauthenticated or unauthorized users may have access to sensitive files and systems, or even user privilege settings.

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.

Weak access controls and issues with credentials management in applications are preventable with secure coding practices, as well as preventative measures like locking down administrative accounts and controls and using multi-factor authentication.

Example: An application allows a primary key to be changed, and when this key is changed to another user's record, that user's account can be viewed or modified.

Solution: An interactive application security testing (IAST) solution, such as Seeker®, can help you effortlessly detect cross-site request forgery or insecure storage of your sensitive data. It also pinpoints any bad or missing logic being used to handle JSON Web Tokens. Penetration testing can serve as a manual supplement to IAST activities, helping to detect unintended access controls. Changes in architecture and design may be warranted to create trust boundaries for data access.

02: Cryptographic Failures

Common errors such as using hardcoded passwords, outdated cryptographic algorithms, or weak cryptographic keys can result in the exposure of sensitive data (the previous name for this category).

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.

Example: A financial institution fails to adequately protect its sensitive data and becomes an easy target for credit card fraud and identity theft.

Solution: Seeker's checkers can scan for both inadequate encryption strength and weak or hardcoded cryptographic keys, and then identify any broken or risky cryptographic algorithms. The Black Duck® cryptography module surfaces the cryptographic methods used in open source software (OSS) so they

can be further evaluated for strength. Both Coverity® static application security testing (SAST) and Black Duck software composition analysis (SCA) have checkers that can provide a "point in time" snapshot at the code and component levels. However, supplementing with IAST is critical for providing continuous monitoring and verification to ensure that sensitive data isn't leaked during integrated testing with other internal and external software components.

03: 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.

Example: An application uses untrusted data when constructing a vulnerable SQL call.

Solution: Including SAST and IAST tools in your continuous integration / continuous delivery (CI/CD) pipeline helps identify injection flaws both at the static code level and dynamically during application runtime testing. Modern application security testing (AST) tools such as Seeker can help secure the software application during the various test stages and check for a variety of injection attacks (in addition to SQL injections). For example, it can identify NoSQL injections, command injections, LDAP injections, template injections, and log injections. Seeker is the first tool to provide a new, dedicated checker designed to specifically detect Log4Shell vulnerabilities, determine how Log4J is configured, test how it actually behaves, and validate (or invalidate) those findings with its patented Active Verification engine.

04: Insecure Design

Insecure design is a new category in the 2021 OWASP Top Ten which 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 of insecure designs creating critical vulnerabilities.

Example: A movie theater chain that allows group booking discounts requires a deposit for groups of more than 15 people. Attackers threat model this flow to see if they can book hundreds of seats across various theaters in the chain, thereby causing thousands of dollars in lost income.

Solution: Seeker IAST detects vulnerabilities and exposes all the inbound and outbound API, services, and function calls in highly complex web, cloud, and microservices-based applications. By providing a visual map of the data flow and endpoints involved, any weaknesses in the design of the app design are made clear, aiding in pen testing and threat modeling efforts.

05: 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.

The 2023 Veracode State of Software Security reported that misconfiguration errors were reported in 70% or more applications that had introduced a new vulnerability in the last year.

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.

Example: A default account and its original password are still enabled, making the system vulnerable to exploit.

Solution: Solutions like Coverity SAST include a checker that identifies the information exposure available through an error message. Dynamic tools like Seeker IAST can detect information disclosure and inappropriate HTTP header configurations during application runtime testing.

06: 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. In a modern application there may be orders of magnitude more code from libraries and components than written by an organization's developers.

As might be expected with any software, vulnerabilities in libraries and components will routinely be discovered, patched, and new versions released. The challenges of identifying all the components in use, keeping track of their vulnerability status, and routinely rebuilding and testing deployed software is both essential and onerous. Perhaps this is why so many organizations are still running vulnerable software in production.

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.

Example: Due to the volume of components used in development, a development team might not know or understand all the components used in their application, and some of those components might be out-of-date and therefore vulnerable to attack.

Solution: Software composition analysis (SCA) tools like Black Duck can be used alongside static analysis and IAST to identify and detect outdated and insecure components in an application. IAST and SCA work well together, providing insight into how vulnerable or outdated components are actually being used. Seeker IAST and Black Duck SCA together go beyond identifying a vulnerable component, uncovering details like whether that component is currently loaded by an application under test. Additionally, metrics such as developer

activity, contributor reputation, and version history can give users an idea of the potential risk that a stale or malicious component may pose.

07 Identification and Authentication Failures

Identifying and authorizing users and non-human clients is a fundamental security practice. It goes without saying that weaknesses in a way an application allows access or identifies users is a critical vulnerability.

While mitigation starts with secure coding practices, tools to detect and prevent credential stuffing and brute force attacks are also useful protections.

Example: A web application allows the use of weak or easy-to-guess passwords (i.e., “password1”).

Solution: Multifactor authentication can help reduce the risk of compromised accounts, and automated static analysis is highly useful in finding such flaws, while manual static analysis can add strength when evaluating custom authentication schemes. Coverity SAST includes a checker that specifically identifies broken authentication vulnerabilities. Seeker IAST can detect hardcoded passwords and credentials, as well improper authentication or missing critical steps in authentication.

08: Software and Data Integrity Failures

The tools used to build, manage, or deploy software are increasingly common vectors of attack. A CI’CD pipeline that routinely builds, tests and deploys software can also be used to inject malicious code (or libraires), create insecure deployments, or steal secrets.

As discussed above in ‘Vulnerable and Outdated Components’ modern applications use many third-party components, often pulling them from third party repositories.

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. Ensuring the build and

Example: An application deserializes attacker-supplied hostile objects, opening itself to vulnerability.

Solution: Application security tools help detect deserialization flaws, and penetration testing can validate the problem. Seeker IAST can also check for unsafe deserialization and help detect insecure redirects or any tampering with token access algorithms.

09: Security Logging and Monitoring Failures

Having adequate logging and monitoring in place is essential in both detecting a breach early, hopefully limiting the damage, and in incident forensics to establish the scope of the breach, and to determine the method of compromise.

Simply generating the data is obviously insufficient, organizations must have adequate collection, storage, alerting and escalation processes. Organizations should also verify that these processes are working correctly –

using Dynamic Application Security Testing (DAST) tools like Veracode DAST, for instance, should produce significant logging and alerting events.

Example: Events that can be audited, like logins, failed logins, and other important activities, are not logged, leading to a vulnerable application.

Solution: After performing penetration testing, developers can study test logs to identify possible shortcomings and vulnerabilities. Coverity SAST and Seeker IAST can help identify unlogged security exceptions.

10: 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.

Example: If a network architecture is unsegmented, attackers can use connection results or elapsed time to connect or reject SSRF payload connections to map out internal networks and determine if ports are open or closed on internal servers.

Solution: Seeker is one of the modern AST tools that can track, monitor, and detect SSRF without the need for additional scanning and triaging. Due to its advanced instrumentation and agent-based technology, Seeker can pick up any potential exploits from SSRF as well.