(https://www.synopsys.com)

Web Application Security

What is web application security?

Web application security (also known as Web AppSec) is the idea of building websites to function as expected, even when they are under attack. The concept involves a collection of security controls engineered into a Web application to protect its assets from potentially malicious agents.

Web applications, like all software, inevitably contain defects. Some of these defects constitute actual vulnerabilities that can be exploited, introducing risks to organizations. Web application security defends against such defects. It involves leveraging secure development practices and implementing security measures throughout the software development life cycle (SDLC) (/software-integrity/resources/knowledge-database/software-development-life-cycle.html), ensuring that design-level flaws and implementation-level bugs are addressed.

Why is web security testing important?

Web security testing aims to find security vulnerabilities in Web applications and their configuration. The primary target is the application layer (i.e., what is running on the HTTP protocol). Testing the security of a Web application often involves sending different types of input to provoke errors and make the system behave in unexpected ways. These so called “negative tests” examine whether the system is doing something it isn’t designed to do.

It is also important to understand that Web security testing is not only about testing the security features (e.g., authentication and authorization) that may be implemented in the application. It is equally important to test that other features are implemented in a secure way (e.g., business logic and the use of proper input validation and output encoding). The goal is to ensure that the functions exposed in the Web application are secure.

The complete web application security testing checklist

Download the checklist (https://www.synopsys.com/blogs/software-security/complete-web-application-security-testing-checklist/)

What are the different types of security tests?

- Dynamic Application Security Test (DAST) (/software-integrity/managed-services/dynamic-analysis-dast.html). This automated application security test is best for internally facing, low-risk applications that must comply with regulatory security assessments. For medium-risk applications and critical applications undergoing minor changes, combining DAST with some manual web security testing for common vulnerabilities is the best solution.
- Static Application Security Test (SAST) (/software-integrity/security-testing/static-analysis-sast.html). This application security approach offers automated and manual testing techniques. It is best for identifying bugs without the need to execute applications in a production environment. It also enables developers to scan source code and systematically find and eliminate software security vulnerabilities.
- Penetration Test (/software-integrity/managed-services/penetration-testing.html). This manual application security test is best for critical applications, especially those undergoing major changes. The assessment involves business logic and adversary-based testing to discover advanced attack scenarios.
- Runtime Application Self Protection (RASP). This evolving application security approach encompasses a number of technological techniques to instrument an application so that attacks can be monitored as they execute and, ideally, blocked in real time.

How does application security testing reduce your organization's risk?

Majority of Web Application Attacks

- SQL Injection
- XSS (Cross Site Scripting)
- Remote Command Execution
- Path Traversal

Attack Results

- Access to restricted content
- Compromised user accounts
- Installation of malicious code
- Lost sales revenue
- Loss of trust with customers
- Damaged brand reputation
- And much more

A Web application in today's environment can be affected by a wide range of issues. The diagram above demonstrates several of the top attacks used by attackers, which can result in serious damage to an individual application or the overall organization. Knowing the different attacks that make an application vulnerable, in addition to the potential outcomes of an attack, allow your firm to preemptively address the vulnerabilities and accurately test for them.

By identifying the root cause of the vulnerabilities, mitigating controls can be implemented during the early stages of the SDLC to prevent any issues. Additionally, knowledge of how these attacks work can be leveraged to target known points of interest during a Web application security test.

Recognizing the impact of an attack is also key to managing your firm's risk, as the effects of a successful attack can be used to gauge the vulnerability's total severity. If issues are identified during a security test, defining their severity allows your firm to efficiently prioritize the remediation efforts. Start with critical severity issues and work towards lower impact issues to minimize risk to your firm.

Prior to an issue being identified, evaluating the potential impact against each application within your firm's application library can facilitate the prioritization of application security testing. With an established list of high profile applications, wenb security testing can be scheduled to target your firm's critical applications first with more targeted testing to lower the risk against the business.

What features should be reviewed during a web application security test?

The following non-exhaustive list of features should be reviewed during Web application security testing. An inappropriate implementation of each could result in vulnerabilities, creating serious risk for your organization.

- Application and server configuration. Potential defects are related to encryption/cryptographic configurations, Web server configurations, etc.
- Input validation and error handling. SQL injection (/software-integrity/resources/knowledge-database/sql-injection.html), cross-site scripting (XSS) (/software-integrity/resources/knowledge-database/cross-site-scripting.html), and other common injection vulnerabilities are the result of poor input and output handling.
- Authentication and session management. Vulnerabilities potentially resulting in user impersonation. Credential strength and protection should also be considered.
- Authorization. Testing the ability of the application to protect against vertical and horizontal privilege escalations.
- Business logic. These are important to most applications that provide business functionality.
- Client-side logic. With modern, JavaScript-heavy webpages, in addition to webpages using other types of client-side technologies (e.g., Silverlight, Flash, Java applets), this type of feature is becoming more prevalent.

Browse Articles