

Veracode OWASP Top 10 2010 Detection

Introduction

Veracode SecurityReview uses a combination of automated static analysis, automated dynamic analysis, and manual penetration testing to assess web applications against the vulnerabilities listed in the OWASP Top 10.

A1-Injection

Tests used: Automated Static, Automated Dynamic, Manual

Veracode SecurityReview's static analysis creates a detailed data flow model covering 100% of the application code. For each supported platform such as Java, .NET, PHP, Cold Fusion, etc, the analyzer has a list of sources of tainted data and a list of sinks where tainted data will cause an injection vulnerability. Example sources are the OS APIs for network and file reads. Example sinks are database queries, OS command interfaces, or LDAP APIs.

Bytecode and interpreted languages have data flow graphs which are relatively easy for a static analyzer to model. The 100% coverage of sources and sinks combined with an accurate data flow modeler creates a high level of accuracy for the automated static detection of injection flaws.

Dynamic analysis uncovers injection vulnerabilities by crawling the live web site, injecting attack strings into forms, headers, cookies, etc. and then using various heuristics to determine which requests caused an exploitable condition.

Manual injection attacks are also attempted in order to add depth in areas where the automated dynamic scanner may not have been able to reach or exploit fully.

A2-Cross Site Scripting (XSS)

Test used: Automated Static, Automated Dynamic, Manual

Much like the testing for Injection vulnerabilities, static analysis is used for XSS testing. The analyzer looks at the data flow model and finds tainted data that is output back to the web browser through the web applications response. If output encoding is not used on the data then XSS is detected. Automated static accuracy for XSS is high.

Dynamic analysis detects XSS vulnerabilities using a similar methodology to other injection attacks. To prevent false positives, each HTTP response is evaluated using a Javascript engine to verify code execution.

Manual penetration will attempt to find XSS, particularly in cases where other methods did not detect any, but this is not a major focus of manual testing since the automated methods are so effective.

A3-Broken Authentication and Session Management

Tests used: Automated Static, Automated Dynamic, Manual

Static analysis has a limited ability to understand the authentication and session state of an application and what the authorization model of the software is. A few vulnerabilities in this category can be found through static analysis such as session fixation.

Dynamic analysis attempts to detect several issues in this category, including default or weak login credentials, authentication bypass (using SQL injection), and session fixation.

Since automation doesn't cover all classes of vulnerabilities in this category, manual penetration testing must be conducted to sufficiently verify that this category isn't present in the application. Manual authentication-related attacks attempt to exploit or subvert an application's authentication infrastructure. During the penetration test, Veracode attempts a number of authentication attacks. These attacks include session/identity spoofing (reuse of session identifiers), testing for default accounts and credentials, and authentication bypass (testing for content or resources that are available without authentication). Possibilities for session hijacking (switching user/role context without credentials) are also evaluated and attempted.

Manual attacks against session management mechanisms aim to compromise the controls used by an application to maintain user sessions and session state information. If session management is handled insecurely, an application's users may be subject to session hijacking and sensitive information disclosure. To test this class of attack, Veracode examines the strength and implementation of application session identifiers (including uniqueness, expiry, and replay capabilities) and evaluates the use of cookies in the application.

A4-Insecure Direct Object References

Tests used: Manual

It is not possible for automated testing to understand the authorization model of an application or whether the values provided by an end user constitute a direct object reference. This design knowledge requires the insight of a manual penetration tester. During the engagement, Veracode will attempt a variety of authorization attacks, including attempts to access resources and data outside of the current user's role and/or community and attempts to discover and access unprotected pages and resources. While Veracode does not limit its authorization testing to insecure direct object references, some authorization vulnerabilities may be due to this condition, which would be noted in the report. Veracode's testing also includes evaluations of the application to determine whether the users could access any functionality or resources not normally permitted.

A5-Cross Site Request Forgery (CSRF)

Tests used: Manual

It is not possible for automation to reliably detect CSRF vulnerabilities without generating an unacceptable rate of false positives. It is also difficult to determine in an automated fashion which functionality within the application need to be protected from such attacks in the first place. Manual penetration testing is required to detect CSRF accurately. Testers look for standard CSRF defenses such as the use of unique tokens accompanying requests for sensitive application functions.

A6-Security Misconfiguration

Tests used: Automated Static, Automated Dynamic, Manual

Security misconfiguration is a broad category that typically requires inspection of the deployed environment. Dynamic analysis inspects the configuration of the run time environment (application servers, web servers, OS, etc.) to detect common misconfigurations such as directory indexing, backup files, and extraneous HTTP methods.

Manual attacks against the system environment and configuration involve discovering and exploiting deficiencies in the deployed system and infrastructure used to support the application. This includes access to administrative interfaces, weak or default passwords, and unnecessary services running on the system. To evaluate this class of attack, Veracode examines the target systems for insecure system settings, out of date software, and other configuration-related items that could lead to system compromise or data exposure. Manual testing also examines third-party software deployed alongside the target application, as vulnerabilities in these products can often be leveraged to attack the primary target.

A7-Insecure Cryptographic Storage

Tests used: Automated Static

Static analysis is uniquely positioned to detect the unsafe usage of encryption APIs and poor management of passwords and keys. Certain variables are automatically flagged as containing potentially sensitive information, and the downstream data flows can be analyzed to determine if the data is ever stored in the clear in a database or in the filesystem.

A8-Failure to Restrict URL Access

Tests used: Manual

Manual analysis is required because the tester needs to determine which pages should be protected by understanding the design of the application. During the engagement, Veracode will attempt a variety of access control attacks, including attempts to access resources and data outside of the current user's role and/or community and attempts to discover and access unprotected pages and resources. Veracode's

testing also includes evaluations of the application to determine whether the users could access any functionality or resources not normally permitted.

A9-Insufficient Transport Layer Protection

Tests used: Automated Static, Automated Dynamic

Automated static testing can be used to detect when a cookie is issued without the Secure attribute. However, it does not know whether the site will actually be deployed using SSL.

Dynamic testing is used to detect SSL configuration issues, such as allowing weak ciphers. It will also detect when cookies are missing the Secure attribute.

A10-Unvalidated Redirects and Forwards

Tests used: Automated Static, Manual

Automated static testing uses the same mechanisms to detect unvalidated redirects as it uses for injection and XSS attacks. If user-supplied data is ever used by the application to construct the target of an HTTP redirect, it is marked as a flaw.

Manual testing is used to inspect for unvalidated redirects as the application is navigated.