VERACODE

Cryptocat

Cryptocat allows users to engage in encrypted chats within a browser. This report summarizes the security flaws identified in the application using manual security analysis techniques, useful for understanding the overall security quality of this application or for comparisons between applications.

The following document represents an attestation of the penetration test by Veracode that was performed on the Cryptocat application.

Customer: Cryptocat

Application: Cryptocat
Manual Analysis Completion Date: 1/11/13

Analysis Technique: Manual Penetration

This application was analyzed manually. The scope included manual testing of the application as presented by the vendor and was determined cooperatively with the Veracode manual testing team to assure maximum coverage of the application's threat landscape within the time constraints of the test and based on available functionality at time of testing. Manual testing simulates real-world attack scenarios and aims to exploit vulnerabilities, identify design flaws, leverage combinations of lower impact flaws into higher impact vulnerabilities, and determine if identified flaws affect the confidentiality, integrity, or availability of the application. Testing includes, but is not limited to:

- Authentication Flaws - This includes testing for session/identity spoofing, testing for default accounts and credentials, and authentication bypass.

- Access Control Flaws - This includes testing for the access to resources and data outside of the current user's role, testing for unprotected pages and resources, testing access to any functionality or resources not normally permitted.

- Session Management Flaws - This includes testing for session fixation and hijacking attack vectors, insecure cookie creation, predictable session identifiers, and long lived sessions.

- Environment Configuration Flaws - This includes testing for access to administrative interfaces, weak or default passwords, unnecessary services running on the system, insecure system settings, out of date software, and other configuration-related items that could lead to system compromise or data exposure.

- Cryptography Flaws - This includes testing for the use of home-grown encryption routines, use of cryptographic functions and algorithms known to be weak or insecure, and insecure file permissions or access controls on cryptographic keys and secrets.

- Data Exposure Flaws - This includes testing for the inappropriate disclosure of sensitive information (user data, passwords, cryptographic secrets) within source code, returned

application responses, and error messages.

- Data Validation and Handling Flaws - This includes testing for weaknesses and issues (Cross-site Scripting, Command/SQL/LDAP/Header Injection, Path Traversal, parameter manipulation, Cross-site request forgery, etc.) in the controls that an application uses to validate data that is input from and returned to the user.

- Error Reporting Flaws - This includes testing how the application logs and reports errors and if any sensitive data that may be useful to an attacker is leaked or disclosed.

Scope and Approach

The test for Cryptocat was a 5-day time-boxed test conducted by a single tester. The 5-day test length was not considered enough for a comprehensive test, and was chosen based on budget and timing restraints. Veracode assigned a senior tester to the test and the following approach was used for the test:

- A thorough discovery process to give the tester context of the application which consisted of reviewing the provided documentation, specifications, threat model, and prior audit results .

- A testing environment which consisted of the supported browsers (Chrome, Firefox, Safari), the installed Cryptocat plugins/extensions, an intercepting proxy to intercept/tamper with the application's requests and responses, and browser debugging tools (such as Firebug and Chrome Developer Tools) to observe and tamper with the client-side execution of Cryptocat's Javascript code

- Tampering with the intercepted XMPP/BOSH responses to attempt to introduce unexpected behavior that may lead to a compromise of the confidentiality/integrity of chat messages

- An audit of selected components of Cryptocat's source code, focusing on basic cryptographic implementation flaws, use of Javascript coding best practices (e.g. the inappropriate use of eval statements, string concatenation, input sanitization and output encoding of tainted user input) that may lead to unauthorized Javascript being executed in a chat user's browser context, and verifying that browser extension best practices were observed

- Dynamically analyzing key data flow paths by setting breakpoints in Cryptocat's Javascript code within browser debugging tools to tamper with Javascript execution in an attempt to compromise the confidentiality/integrity of chat messages

Analysis Result: 100/100

Within the scope and approach illustrated above, Veracode analyzed, scored and rated the Cryptocat product using manual analysis (outlined above). The resulting security quality score is 100 (out of 100). For context, the scoring method begins with a 100, and is decremented based on the flaws discovered.

Disclosure

Penetration tests are a point-in-time identification of vulnerabilities in a given application ("Target"). All products are prone to—and inclusive of—errors, omissions, and defects to varying degrees of severity and quantity. The manual penetration testing team executes a set of pre-defined test cases as bounded by the test engagement's scope. The team uses an approach and tactics based on prior experience and perceived industry best practice at the time of testing. It should be understood that the testing team cannot guarantee that the final deliverable accounts for all possible defects that may or may not be present within the Target. Furthermore, as new vulnerability types and testing approaches arise, it is conceivable that such vulnerability types or discovery of the further existence of previous vulnerability types—made known through use of new testing approaches—can and will occur.

Therefore, Veracode cannot and does not warrant their work product(s) as being "complete" with all instances of every form and kind of vulnerability that may or may not exist within the Target. Rather, the testing team simply performs a "best effort" attempt at identifying vulnerabilities, as limited by the scope (boundary) statement(s) agreed upon by both parties (Veracode and the Application Contact[s]).