September 30, 2014

Bromium
20813 Stevens Creek Boulevard
Cupertino, CA 95014

Bromium **vSentry** v2.4 Assessment

In September 2014, IOActive performed an in-depth source code review, architecture audit, and penetration test of Bromium vSentry v2.4. The goal of the assessment was to evaluate the security of the system including its architecture and implementation, and to audit the source code.

IOActive established the following properties of the system:

- Architectural Design: Secure
- Source Code: Secure
- Runtime Functionality: Secure
- Penetration Test Results: Secure

Overall Results: The Bromium vSentry v2.4 system architecture and code implementation practices are sound, and no vulnerabilities were identified. Exhaustive penetration testing of the vSentry environment failed to breach the containment of a micro-VM or permit an attacker to compromise the Microvisor.

701 5TH AVENUE SUITE 6850, SEATTLE, WA 98104 **T** 206 784 4313 866 760 0222 toll free **F** 206 784 4367

Source Code Review and Architecture Audit

Methodology: IOActive reviewed the source code and architecture of Bromium vSentry v2.4 with the goal of identifying vulnerabilities in the logic of the source code that could enable an attacker to compromise vSentry or the Windows desktop that vSentry protects.

Before starting the actual source code review, we performed an entry-point analysis looking for trust boundaries between high-priority features by:

- Evaluating the system architecture in consultation with Bromium
- Evaluating threat model data flow diagrams (DFDs)
- Evaluating the vSentry source code
- Reviewing the system design documentation
- Discussing the architecture and implementation with Bromium developers

The results of the entry point analysis allowed IOActive to identify areas that warranted deeper analysis. As we analyzed the system, we performed these types of checks:

- Developed a basic dataflow analysis
- Performed text and token matching on the code base, including comments
- Built on the entry-point analysis to map the attack surface
- Analyzed communication protocols
- Analyzed component use
- Evaluated configuration of applications and libraries
- Evaluated authorization logic
- Evaluated communication security
- Evaluated encryption-key management
- Evaluated input validation and sanitization
- Evaluated output encoding
- Evaluated how session management is implemented
- Reviewed logging subsystems for security, privacy, and data leakage
- Evaluated Operating System error notification routines for information leakage
- Verified that direct-object references are protected
- Identified issues that may pose a denial-of-service risk

Results: No significant vulnerabilities or errors were found in the source code or architecture design.

Runtime Functionality and Penetration Testing

Concepts: IOActive identifies vulnerabilities as points where applications or network components exhibit errors; the following list gives examples of the conceptual context in which these errors are generally seen.

Concept	Description
Authentication	Confirming a user's identity or ensuring that a program can be trusted
Access Controls	Methods used to authenticate the identity of a user, such as username and password combinations
Broken Authentication and Session Management	Account credentials/session tokens are not protected properly, so attackers compromise passwords or keys to assume identities
Configuration	How securely servers, devices, and software are chosen and implemented or deployed
Cross-site Request Forgery (CSRF)	A browser is forced to send a pre-authenticated request to a vulnerable application, which then forces the browser to perform a hostile action that benefits the attacker
Cross-site Scripting (XSS)	When an application accepts user-supplied data and sends it to a web browser without first validating or encoding that content
Cryptography and Insecure Storage	Applications rarely use mathematical data protections properly; attackers can conduct identity theft and credit card fraud
Data Validation	Ensuring that a program operates on clean, correct, useful, and secure data
Denial of Service	Anything that makes a computer resource unavailable to its intended users
Failure to Restrict URL Access	When an application protects sensitive functionality by preventing its display as opposed to restricting access
Information Leakage and Improper Error Handling	When an application exposes information about its configuration or internal function, or violates user privacy
Insecure Communication	When an application fails to encrypt sensitive network traffic
Insecure Direct Object Reference	When a reference to an internal implementation object (file, directory, database record, key, URL, etc.) is exposed
Malicious File Execution	Code that is vulnerable to remote file inclusion allows attackers to include hostile code and data
Session Management	The process of tracking a user's activity across sessions of interaction with a computer system

Methodology: For the runtime review, IOActive consultants installed vSentry 2.4.0.11033 in a VMware Fusion® hosted Windows® 7 virtual machine.

Host Testing

After installing vSentry, IOActive reviewed the installed system files and examined the system to understand communication paths between code isolated in a micro-VM and the Microvisor, and code in a micro-VM and components of the vSentry system that run on the protected desktop. IOActive used these paths in its attempts to attack the Microvisor and the desktop. In addition, IOActive consultants attempted to bypass the "Untrusted" file protections used by vSentry for untrusted document isolation.

Testing the Isolation of a micro-VM

An attacker that is exploring Bromium on their own system, may need to install a vulnerable version of Internet Explorer, Adobe® Acrobat, Chrome, or other application that processes untrusted content. In addition an attacker will need a method to deliver an attack, such as a malicious file that spawns cmd.exe – for example from a malicious web site. This is the method used by IOActive.

Results: Extensive testing showed that IOActive was unable to initiate communications with the protected desktop from a micro-VM. The isolation of code contained within the micro-VM functioned as designed. Efforts to bypass "Untrusted File" protections were also unsuccessful.

Matt Rahman

Vice President of Business Development and Sales