Date: 12 June 2014

**Some Project Code Review**

**Prepared by**

**Some Organization**

**For**

**Some Organization**

Version 1.0

Background

In order for SOME SOFTWARE to pass the annual STIG review, a code review of the entire application was done. The main focus of this document will cover the testing results for potential vulnerabilities such as SQL Injection, XML Injection (APP3810) and Fuzz Testing (APP5100).

In addition to the descriptions of how we tested below, Threat Model information is described below and the Retina scan results for our lab server have been inserted at the end of this document.

SQL Injection

The testing for SQL injection was three-fold. First, we ran an automated source code analysis using SonarQube. Second, we used Retina to scan our live systems in an attempt to identify additional vulnerabilities that could have been missed by static code analysis. Third, we manually attempted SQL Injection in various parts of the system (login pages, search boxes, product registration boxes, etc).

During our testing, no vulnerabilities were identified. Although search results are returned for SQL injections attempts, these results are being returned based off single word matches. For example, if someone attempts to search for “SELECT password FROM user;” some results will be returned. These results are simply products that contain one or more of the words in the search string. This is not a vulnerability.

SonarQube identified a number of false positive potential vulnerabilities relating to prepared statements generated from nonconstant Strings. These findings, if they were true positives, could be potential SQL Injection attack vectors. The findings are outlined below, and were also previously detailed in the document entitled “SonarQube Run Against SOME SOFTWARE” located at <provide location>. For additional information on how we handled our manual testing, please see <https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OWASP-DV-005)>

|  |  |  |
| --- | --- | --- |
| **File** | **Lines affected** | **Description** |
| Queries.java | 378, 543, 658, 758 | Not a vulnerability because the string is nonconstant due to appending table name and filter information that are environment specific from configuration. No user input. |
| 826 | Not a vulnerability because the string is nonconstant due to appending bounding box information from a bounding box object. No user input. |
| 937 | Not a vulnerability because the string is nonconstant due to appending a list of uuids. No user input. |
| ImsMetadataAdminDao.java | 194, 1040, 1583 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| ImsMetadataProxyDao.java | 326 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HjLoadAllRequest.java | 99 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HeSelectOneRequest.java | 83 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HeUpdateRequest.java | 99 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HeSelectRequest.java | 180 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HeDeleteRequest.java | 109 | Not a vulnerability because the string is nonconstant due to appending a list of uuids. No user input. |
| HjCreateRequest.java | 160, 223 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HjGetNextRequest.java | 117, 195 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HjCancelRequest.java | 83 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HjCompleteRequest.java | 142 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HjWithdrawRequest.java | 74 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HrDeleteRequest.java | 125 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HrHarvestRequest.java | 232 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| HrSelectRequest.java | 342 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| SingleIndexingServlet.java | 239 | Not a vulnerability because the string is nonconstant due to appending table name and UUIDs that are environment specific from configuration. No user input. |
| LuceneIndexSynchronizer.java | 245 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| MmdQueryRequest.java | 216 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| GptRepository.java | 417 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| RepositoriesServlet.java | 120 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| ThumbnailServlet.java | 88 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| DataMigration.java | 610 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |
| ErosQueryServlet.java | 174 | Not a vulnerability because the string is nonconstant due to appending table names that are environment specific from configuration. No user input. |

XML Injection – APP3810

The testing for XML Injection was very similar to the testing for SQL Injection. As done in SQL Injection testing, our testing strategy was three-fold. First, we ran an automated source code analysis using SonarQube. Second, we used Retina to scan our live systems. Third, we manually attempted XML Injection in various areas of the system (login pages, search boxes, product registration boxes, etc).

During our testing, no vulnerabilities were identified. We were able to correctly process any attempt to inject XML injection and the resulting XML was still valid.

In addition to the above, we also make use of XSDs to prevent incorrect XML from being inserted.

For additional information on how we handled our manual testing, please see <https://www.owasp.org/index.php/Testing_for_XML_Injection_(OWASP-DV-008)>

Fuzz Testing – APP5100

Fuzz testing was all handled by running Retina against our lab servers. After a run against each server, we inspected the results for potential problems discovered by Fuzz testing. During our testing, no problems were found.