OFFICE OF INFORMATION SECURITY

PASS: This application has successfully completed the V&V Secure Code Review Validation Process.

Secure Code Review Validation Report

TAS VistaLink v02.00.2

Application-ID: C3BDB4FB-492B-4660-A04B-31E43F472B70

Filename: VA SwA Code Validation TAS VistaLink v02.00.2 2019-05-14 PASS.pdf

MAY 14, 2019

U.S. Department of Veterans Affairs

Office of Information and Technology

Software Assurance Program Office

Table of Contents

1 Secure Code Review Validation Report Introduction

This document contains the results of the validation by the VA Software Assurance Program Office of a secure code review of TAS VistaLink performed by the developer.

This document contains the following additional sections:

Section 2. Secure Code Review Validation Results

This section summarizes the results of the validation of the developer secure code review.

Section 3. Secure Code Review Validation Process Details

This section describes how the validation of the developer secure code review was performed.

Section 4. Secure Code Review Validation Findings and Recommendations

This section provides residual secure code review validation findings that should have already been fixed prior to the validation. Recommendations are also provided.

Section 5. Secure Code Review Validation Report Conclusion

This section provides additional recommendations to build security in during development.

1.1 Application Information

The version of TAS VistaLink for which static analysis tool scan results were provided was v02.00.2. The following was provided by the developer for review:

1. Completed V&V Secure Code Review Validation Request Form
2. VA_VISTALINK_2_fortify_scan.fpr – Fortify Static Code Analyzer (SCA) static analysis tool scan result file
3. VistaLink_02.00.02.zip – TAS VistaLink v02.00.2 source code

2 Secure Code Review Validation Results

This document contains the results of a Verification and Validation (V&V) review, conducted by the VA Software Assurance Program Office, of developer-provided Fortify SCA static analysis tool scan result files, and of any provided custom scan tool custom rule files, as well as the TAS VistaLink v02.00.2 source code. Goals of performing secure code reviews at the VA include ensuring that risk-based activities in applications are performed in a secure manner. Goals of V&V secure code review validations include ensuring that secure code reviews performed by VA software developers have been done correctly and consistently.

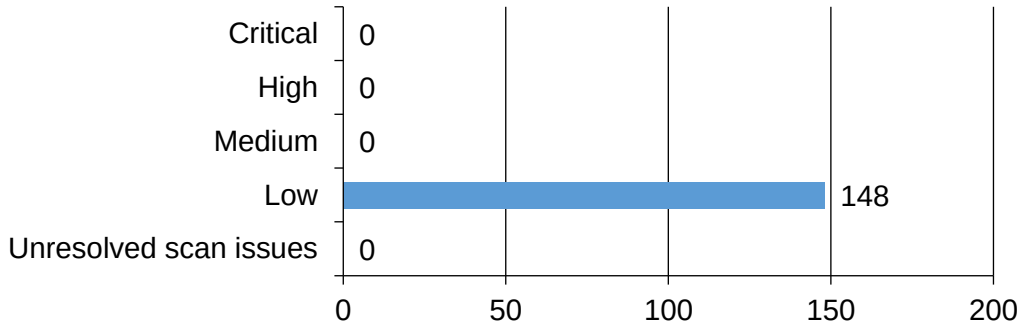
The V&V secure code review validation conducted by the VA Software Assurance Program Office covered provided materials to ensure that:

1. Application information in secure code review validation request packages is accurate and complete, and
2. Application scan results demonstrate that VA standards have been met, and
3. Application scan results demonstrate that mitigations must have been made for issues reported by the Fortify SCA static analysis tool, and
4. There are justifications provided for cases where Fortify SCA static analysis tool rules are disabled, or scan results are marked as false positives.

For more information about the validation process, see Section 3.

The V&V secure code review validation conducted by the VA Software Assurance Program Office identified a total of 0 residual vulnerabilities that were considered Critical in severity. There was a total of 0 residual High severity vulnerabilities. There was a total of 0 residual Medium severity vulnerabilities. There was a total of 148 residual Low severity vulnerabilities. There was a total of 0 unresolved scan issues.

Figure 1. Summary of Residual Vulnerabilities & Unresolved Scan Issues



For more information about residual vulnerabilities and unresolved scan issues that were identified during the secure code review validation, see Section 4.

3 Secure Code Review Validation Process Details

The secure code review validation was performed overall as follows:

Step 1. Perform initial planning

The first step that was performed was to perform initial planning. This included developing a strategy for performing the review and identifying considerations that should be taken into account during the review, such as any Fortify SCA static analysis tool custom rule files provided by the developer.

Step 2. Review source code

The next step is to perform the review. A combination of using Fortify SCA to review scan result files and manual analysis was used. The scan results were reviewed to ensure that best practices for performing secure code review have been followed, and that VA standards have been met, as noted in the previous section.

Step 3. Write report

The last step in the secure code review validation process is to write up the report, after working with the VA application developer to resolve any issues identified during review.

3.1 Validation Strategy

The secure code review validation was performed by reviewing Fortify SCA static analysis tool scan result files and any provided Fortify SCA static analysis tool custom rule files. The provided source code was reviewed as need to support analysis of the provided scan result and custom rule files. The secure code review validation included at a minimum the following checks:

Review developer-provided scan file for matching source code

This validation check consists of ensuring that the source code matches the uploaded static analysis tool scan result files. While during the comparison there may be some differences such as build files, source code files should not contain any differences.

Review developer-provided scan file for scanning issues

This validation check consists of reviewing static analysis tool scan result file for any anomalies in the scan. When running the scan there may have been issues reported by the static analysis tool that affected the quality or completeness of the scan that may have been overlooked.

Review developer-provided scan file for residual findings

This validation check consists of ensuring that there are no Critical or High findings in the uploaded static analysis tool scan result file (Fortify SCA ".fpr"

extension file) using Fortify Audit Workbench, after first configuring it to use any provided custom rule files.

Review developer-provided scan file for suppression of issues

This validation check consists of reviewing static analysis tool scan result files to ensure that issues reported by Fortify SCA have not been suppressed, as opposed to adding comments and developing custom rules as might be appropriate.

Review developer-provided custom rule files, if provided

This validation check consists of reviewing any provided static analysis tool custom rule files. Analysis includes examining custom rule files e.g. to ensure that there are no rules to disable built-in Fortify rules, unless those custom rules include documentation justifying their use.

Perform additional supporting analysis, as needed

This validation check consists of performing additional supporting analysis for items that may have been identified during the course of the validation for a particular application. For example, findings in the scan result files have been marked as N/A, checks would be performed to ensure there is some documented justification, and to verify the soundness of the justification. Alternately for example, analysis may be performed to determine the appropriateness of exclusions.

3.2 Tools Used for Validation

The VA Software Assurance Program Office uses the same static analysis tool (Fortify SCA) as VA application developers. The same static analysis tool is used in order to promote confidence in the outcome of the secure code review validation if the tool is in fact being used during development. Fortify SCA version 18.20 was used to review provided static analysis tool scan result and custom rule files. The Audit Workbench tool which is part of Fortify SCA was used to facilitate examining static analysis tool scan result files. Similarly, the Custom Rules Editor tool which is also part of Fortify SCA was used to facilitate examining custom rule files.

3.3 Categorization of Findings

The findings that resulted from performing the secure code review validation are grouped in Section 4 of this report by severity and type of vulnerability. Findings were rated according to severities reported by the Fortify SCA tool, and/or at the discretion of the VA Software Assurance Program Office as follows:

Findings that are Critical in severity

Vulnerabilities in source code that must be fixed immediately, for example exposed passwords or Personally-Identifiable Information (PII).

Findings that are High in severity

Vulnerabilities in source code that allow an attacker immediate access into a machine, allow super user access, or bypass a firewall.

Findings that are Medium in severity

Vulnerabilities in source code that provide information that have a high potential of giving access to an intruder.

Findings that are Low in severity

Vulnerabilities in source code that provide information that potentially could lead to compromise.

Findings that are unresolved scan issues

This finding categorization is reserved for issues having to do with how the scan was conducted, for example, source code not matching the uploaded static analysis tool scan result files.

Additional findings

This finding categorization is reserved for any additional concerns identified by the VA Software Assurance Program Office review that do not correspond to the categories above. For example, new issues may be identified during the course of the validation while reviewing supporting documentation.

4 Secure Code Review Validation Findings and Recommendations

4.1 Residual Critical Findings (0 Total)

Based on the information provided by the developer, it does not appear that vulnerabilities identified by Fortify SCA that were Critical in severity were left unmitigated.

CWE-ID	CWE-Title	Number of Instances	Notes
CWE-325	Weak Encryption: Missing Required Step	0	**Closed** Accept developer analysis The KeyGenerator is initialized in the getKeyGenerator() method. Additionally, this method is overwritten in the PassPhraseEncrypter class. Note that this version of the method is called from the getKey() method which itself is called from the getCipher() method which is called from the public encrypt() method, all within the KeyEncrypter class. It does not appear that KeyEncrypter.encrypt() is used within the codebase, so if it is dead code it should be removed to avoid this potential vulnerability if this method is called in the future.

4.2 Residual High Findings (0 Total)

Based on the information provided by the developer, it does not appear that vulnerabilities identified by Fortify SCA that were High in severity were left unmitigated.

CWE-ID	CWE-Title	Number of Instances	Notes

CWE-247, CWE-292, CWE-558, CWE-807	Often Misused: Authentication	0	**Closed** Accept developer analysis This is mock/test code. Additionally, DNS is not used for application security.

4.3 Residual Medium Findings (0 Total)

Based on the information provided by the developer, it does not appear that vulnerabilities identified by Fortify SCA that were Medium in severity were left unmitigated.

4.4 Residual Low Findings (148 Total)

The vulnerabilities below were identified by Fortify SCA that were Low in severity were left unmitigated and are still being reported by Fortify SCA. It is estimated that it will take approximately 37 hours to address the following issues.

CWE-ID	CWE-Title	Number of Instances	Notes
N/A	Build Misconfiguration: External Maven Dependency Repository	4	This issue has not been audited.
CWE-498	Code Correctness: Class Does Not Implement Cloneable	1	This issue has not been audited.
CWE-398	Code Correctness: Class Does Not Implement equals	3	This issue has not been audited.
N/A	Code Correctness: Constructor Invokes	11	This issue has not been audited.

	Overridable Function		
CWE-486	Code Correctness: Erroneous Class Compare	2	This issue has not been audited.
CWE-570	Dead Code: Expression is Always false	1	This issue has not been audited.
CWE-571	Dead Code: Expression is Always true	1	This issue has not been audited.
CWE-561	Dead Code: Unused Method	11	This issue has not been audited.
N/A	Denial of Service: StringBuilder	4	This issue has not been audited.
CWE-489	J2EE Bad Practices: Leftover Debug Code	2	This issue has not been audited.
CWE-246	J2EE Bad Practices: Sockets	3	This issue has not been audited.
CWE-383	J2EE Bad Practices: Threads	4	This issue has not been audited.
CWE-253, CWE-690	Missing Check against Null	3	This issue has not been audited.
CWE-112	Missing XML Validation	1	This issue has not been audited.
CWE-581	Object Model Violation: Just one of equals() and hashCode() Defined	3	This issue has not been audited.
CWE-615	Password Management: Password in Comment	2	This issue has not been audited.

CWE-391	Poor Error Handling: Empty Catch Block	1	This issue has not been audited.
CWE-396	Poor Error Handling: Overly Broad Catch	6	This issue has not been audited.
CWE-397	Poor Error Handling: Overly Broad Throws	29	This issue has not been audited.
CWE-395	Poor Error Handling: Program Catches NullPointerException	7	This issue has not been audited.
CWE-398	Poor Error Handling: Throw Inside Finally	1	This issue has not been audited.
CWE-398	Poor Logging Practice: Use of a System Output Stream	2	This issue has not been audited.
CWE-398	Poor Style: Confusing Naming	2	This issue has not been audited.
CWE-398	Poor Style: Identifier Contains Dollar Symbol ($)	2	This issue has not been audited.
CWE-493	Poor Style: Non-final Public Static Field	13	This issue has not been audited.
CWE-563	Poor Style: Value Never Read	2	This issue has not been audited.
CWE-476	Redundant Null Check	2	This issue has not been audited.
CWE-497	System Information Leak	15	This issue has not been audited.
CWE-497	System Information Leak: Internal	8	This issue has not been audited.
CWE-470,	Unsafe Reflection	1	This issue has not been audited.

CWE-494			
CWE-776	XML Entity Expansion Injection	1	This issue has not been audited.

4.5 Issues with How Scans Were Performed (0 Total)

4.5.1 Unresolved Scan Issues

Based on the information provided, it does not appear that there were unresolved issues when the scan of the source code was conducted.

4.5.2 Informational Scan Issues

Based on the information provided, it does not appear that there were informational issues when the scan of the source code was conducted.

4.6 Additional Findings (0 Total)

There were no additional findings that were identified during the course of the validation.

5 Secure Code Review Validation Report Conclusion

Designing secure applications is every VA application developer's responsibility. Application-level vulnerabilities generally manifest themselves as one of two types: **design flaws** introduced by weaknesses during the requirements, design, or architecture phase; or **implementation bugs** introduced by weaknesses during the actual coding of the application.

Fortify SCA should be used according to the VA Secure Code Review SOP to minimize design flaws during application implementation. Fortify SCA supports a wide range of programming languages and build environments. The VA-licensed Fortify SCA can also be used as a standalone tool by VA developers, or integrated into for example Continuous Integration (CI) build environments.

The VA Software Assurance Program Office uses the same static analysis tool (Fortify SCA) as VA application developers during the secure code review validation process to ensure consistency and completeness of analysis.

Note that remediation estimates provided in this report are provided to assist with planning remediation work, if or as might be appropriate. Note that some bugs are harder to fix than others. Modifying a single line of code in a self-contained method for example is easier than modifying the result of a sequence of calls. Systems development program and project-specific considerations should also be taken into account when planning remediation work. Read more…

5.1 Resources that you may find helpful

The following resources may be helpful to readers of this report:

VA Software Assurance Support Site

This site provides VA Software Assurance Program Office resources to assist VA application developers with performing secure code reviews and secure design reviews during development and also during Assessment and Authorization (A&A) and continuous monitoring.

VA Secure Code Review Standard Operating Procedures (SOP)

This document provides establishes policies and procedures for performing secure code review (static analysis) of custom-developed applications at the VA.

Fortify product documentation

This documentation is included as part of Fortify software distribution. It includes system requirements documentation and user guides.

OWASP Top Ten

This site provides application security industry insight into common web application security flaws that is compiled by the OWASP Foundation.

CWE/SANS Top 25

This site provides application security industry insight into common application security flaws that is compiled by MITRE and SANS.