NIST Special Publication 500-270 v1.1

Source Code Security Analysis Tool Test Plan Version 1.1

Michael Koo

Romain Gaucher

Charline Cleraux

Jenise Reyes Rodriguez

Software and Systems Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

July 2011

U.S. Department of Commerce
Rebecca M. Blank, Acting Secretary

National Institute of Standards and Technology
Dr. Patrick D. Gallagher, Director

Abstract

This document provides a set of metrics, including test suites and methods, to determine how well a particular source code security analysis tool conforms to the requirements specified in *Source Code Security Analysis Tool Functional Specification Version 1.0* [SCA]. Each relevant programming language in [SCA] has a corresponding set of test suites. The test suites are intended to be used by tool developers and tool users alike to increase their level of confidence in product quality. Each test suite consists of test cases that are designed to evaluate against various requirements of [SCA], including mandatory features and optional features. Each test case contains a test description, a description of the weakness contained in the test case, the expected result and the test code. Detailed information on the test case, such as start parameters, procedures for executing the test file and the test file itself can be retrieved from the SAMATE Reference Dataset (SRD) at http://samate.nist.gov/SRD/.

As this document evolves, new versions will be posted to the web site at http://samate.nist.gov/index.php/Source_Code_Security_Analysis.html.

Keywords

Source code security analysis tool; test plan; test methodology; test suite.

Changes to this version

This version 1.1 updates version 1.0 by modifying C and C++ test suites and adding Java test suites.

Table of Contents

1 Introduction

There is a critical need in information technology to ensure that software assurance tools produce accurate, repeatable and objective results. The Software Assurance Metrics and Tool Evaluation (SAMATE) project at the National Institute of Standards and Technology (NIST) [1] is working to establish a methodology for measuring the functionality and capability of software assurance tools by developing functional specifications, test procedures, test criteria, and test suites. The results provide the information necessary for the toolmakers to improve tools, for users to make informed choices about acquiring and using software assurance tools, and for interested parties to understand the tools' capabilities.

This document updates a test plan, including the test methodology and test suites, for Source Code Security Analysis Tools that examine program source code to detect and report weaknesses that can lead to security vulnerabilities. Other static analysis tools, for examples tools that scan bytecode or binary code, are not covered.

2 Purpose and Scope

This document, along with some specific test suites in the SAMATE Reference Dataset (SRD) [SRD], provides a means to test the functionality and capability of a Source Code Security Analysis Tool based on the requirements asserted in *Source Code Security Analysis Tool Functional Specification Version 1.0* [SCA].

The test plan is generic in that it can be applied to any programming language. However, each language will have its own specific set of test suites for measuring the tool.

The test methodology described in this document focuses only on the functional specification in [SCA]. Testing the performance robustness, scalability, usability, etc. of a source code security analysis tool is outside of the scope of this document.

3 Test Methodology

[1] This project is sponsored by the U.S. Department of Homeland Security (DHS) National Cyber Security Division and NIST. Detailed information about this project is described at http://samate.nist.gov/.

3.1 Requirements from [SCA]

3.1.1 Requirements for Mandatory Features

To meet a minimum level of capability, a source code security analysis tool or set of tools must be able to accomplish the tasks described below. The tool(s) shall:

SCSA-RM-1: Identify all of the classes of weaknesses listed in Annex A of [SCA].

SCSA-RM-2: Textually report any weaknesses that it identifies.

SCSA-RM-3: For any identified weaknesses in classes listed in Annex A of [SCA], report the class using a semantically equivalent name.

SCSA-RM-4: Report the location of any weaknesses by providing the directory path, file name and line number.

SCSA-RM-5: Identify weaknesses despite the presence of the coding complexities listed in Annex B of [SCA].

SCSA-RM-6: Have an acceptably low false positive rate.

3.1.2 Requirements for Optional Features

The following requirements apply to optional tool features. If the tool supports an optional feature, then the requirement for that feature applies, and the tool can be tested against it. A specific tool might optionally provide none, some, or all of the features described by these requirements. Optionally, the tool(s) shall:

SCSA-RO-1: Produce an XML-formatted report.

SCSA-RO-2: Not report a weakness instance that has been suppressed.

SCSA-RO-3: Use the Common Weakness Enumeration [CWE] name of the weakness class it reports.

3.2 Measurement of Fulfillment of Requirements

In general, measuring source code security analysis tool takes three steps:

1. Preparation: Set up testing environment, study functionalities of the tool, become familiar with the material in Annex A of [SCA], review test suites outlined in this test plan, etc. Specific examples of tasks that may be part of this step include:
    - install tool
    - understand the capacity of the tool
    - establish the corresponding relationship between weaknesses that the tool can detect and the weakness listed in Annex A of [SCA]
    - choose appropriate test suites, (see Section 4).
2. Run tool on test cases in test suites – determine if results are as expected.

3. Summarize results.

This test plan includes 9 (nine) test suites. Each test suite evaluates the source code security tool for one or multiple requirement specified in Sections 3.1.1 and 3.1.2. Each test suite consists of a set of test cases in one of the languages, i.e., C, C++, or Java. A test case demonstrates a weakness in a specific code construct; or a test case is a fixed version of code that has the weakness removed to measure the capability of the tool to avoid generating false positives.

The expected result of a test case will be a report of weakness, if a weakness is seeded in the code; or it will not report the weakness, if a weakness is not seeded or reporting the weakness is suppressed. If a weakness is reported, the expected result must be a weakness from classes listed in Annex A of [SCA], or a weakness from that class reported using a semantically equivalent name. The report must also include the location of the weakness.

3.2.1 Naming Convention

Annex A of [SCA] covers three computer languages, C, C++ and Java. Each computer language has three corresponding test suites. Test suite SCA-TS-1-*name*, where *name* is the name of the language, consists of test cases in that language. For instance, SCA-TS-1-CPP is for the C++ language and SCA-TS-1-Java is for Java. Test suite SCA-TS-1 tests features that are described in requirements SCA-RM-1 through SCA-RM-5. It can also test requirements SCA-RO-1 and SCA-RO-3. Test suite SCA-TS-2-*name* tests requirement SCA-RM-6, false alarms or false positives. Test suite SCA-TS-3 tests requirement SCA-RO-2, suppression of warnings.

Dependent on the language and requirements to be tested, download the appropriate test suite from the SRD, see Section 4.

Since there may be up to several hundred individual test results to check, we encourage users to write a harness for their own needs. The “More Downloads” section of the SRD provides sample scripts. All the tests could be run, and results saved, then the determination made if the results are as expected. Alternatively the determination could be made as each test is run.

Note that the results from running the tool on test suite SCA-TS-1 are used for many requirements. The results of SCA-TS-2 and SCA-TS-3 are used for one requirement each.

3.2.2 Mandatory Features

- To determine if SCA-RM-1 is met from the results of SCA-TS-1 …

  To determine if SCA-RM-2 is met from the results of SCA-TS-1 …

  To determine if SCA-RM-3 is met from the results of SCA-TS-1 …

  To determine if SCA-RM-4 is met from the results of SCA-TS-1 …

  To determine if SCA-RM-5 is met from the results of SCA-TS-1 …

  For each test case, the tool under test is expected to generate a report that identifies the seeded weakness with a name semantically equivalent to one of those in Annex A of [SCA] and its location (e.g., the path name and line number(s) of the weakness).

- To determine what to report for SCA-RM-6 from the results of SCA-TS-2 …

  For each test case, the original seeded weakness is fixed or removed. The tool under test should not report any weakness. If it does, practical considerations require the *false positive rate* [Fleiss] to be acceptably low for the domain.

3.2.3 Optional Features

- To test requirement SCA-RO-1, turn on that feature if necessary and then run test suite SCA-TS-1 for the appropriate language. The tool under test should generate a report as described in 3.2.2 in the appropriate XML-format.
- To test requirement SCA-RO-2, run test suite SCA-TS-3 first to prove the tool can identify the weaknesses. Then run SCA-TS-3 again after the weaknesses described in Annex A of [SCA] are suppressed in the tool under test. The tool should generate no reports for any weakness that is suppressed.
- To test requirement SCA-RO-3, run test suite SCA-TS-1 for the appropriate language. For each test case, the tool under test is expected to generate a report that identifies the incorporated weakness with the correct CWE name.

4 Test Suites

A test suite is a collection of test cases explicitly selected for a special purpose. Each test case is an atomic program that tests a specific tool functionality required by [SCA]. Test suites and their test cases are stored in the SRD [SRD]. Each SRD test case entry provides a test file, a description of the test case, the CWE classification of the incorporated weakness, any type(s) of code complexity present, the location of the weakness, and other test case related information. This test plan includes test suites for C, C++ and Java languages.

4.1 Test Suites for the C Language

4.1.1 Test Suite SCA-TS-1-C (SRD Test Suite 45)

http://samate.nist.gov/SRD/view.php?tsID=45

This test suite tests the capability of a source code security analysis tool's handling of weaknesses in the C language. It covers the source code weaknesses in C listed in Annex A of *Source Code Security Analysis Tool Functional Specification* [SCA].

Source Code Weakness	**CWE ID**	**Code Complexity**	**SRD Test case ID**	**Remark**
Basic XSS	80	Basic	1794	
		Scope	1781	
		Address alias level	1919	
		Container	1921	
		Loop complexity	2198	
Resource Injection	99	Basic	1897	
		Scope	1901	
		Address alias level	1895	
		Container	1899	
OS Command Injection	78	Basic	111	Test function 'system()'.
		Scope	1885	
		Local control flow	1881	
		Loop structure	1883	
SQL Injection	89	Basic	1796	
		Array index complexity	1798	
		Scope	1800	
Stack Overflow	121	Basic	2009	
		Array index complexity	1544	
		Scope	1548	
		Basic	1563	Test function gets()
		Basic	1565	Test function fgets()
		Array index complexity	1751	
		Array length/limit complexity	1905	
		Index alias level	1907	
		Loop Structure	1909	
Heap Overflow	122	Basic	1611	
		Scope	1612	
		Array address complexity	1843	
		Array index complexity	1845	
Format string	134	Basic	10	

vulnerability				
		Address alias level	92	
		Scope	93	
		Container	1831	
		Local control flow	1833	
Improper Null Termination	170	Basic	1849	
		Taint	1857	
		Buffer address type	2010	
		Container	1854	
		Address alias level	1850	
Heap Inspection	244	Basic	1737	
Often Misused: String Management	251	Basic	1865	
		Taint	1873	
		Scope	1871	
		Address alias level	1867	
		Container	1869	
Hard-coded Password	259	Basic	1810	
		Local control flow	1839	
		Loop structure	1841	
		Container	1837	
		Array Index Complexity	1835	
Time-of-check Time-of-use race condition	367	Basic	102	
		Basic	1806	
		Local Control Flow	1808	
Unchecked Error Condition	391	Basic	1928	
Memory leak	401	Basic	1585	
		Scope	1588	
Unrestricted Critical Resource Lock	412	Basic	2109	
Double Free	415	Basic	2199	
		Buffer address type	99	
		Loop structure	1829	

		Local control flow	1827	
		Scope	1590	
Use After Free	416	Basic	2200	
		Scope	2201	
		Container	2202	
		Buffer address type	2203	
Uninitialized variable	457	Data type	2019	Data type is integer
		Data type	2003	Data type is int *
		Loop Structure	1757	
Unintentional pointer scaling	468	Basic	1782	
Null Dereference	476	Basic	2193	
		Address alias level	1875	
		Local control flow	1877	
		Scope	1879	
Leftover Debug Code	489	Basic	1861	

4.1.2 Test Suite SCA-TS-2-C (SRD Test Suite 46)

http://samate.nist.gov/SRD/view.php?tsID=46

This test suite can be used to assess the false positive ratio of the tool under test for C applications.

Source Code Weakness	**CWE ID**	**Code Complexity**	**SRD Test case ID**	**Remark**
Basic XSS	80	Basic	1795	
		Scope	1924	
		Address alias level	1920	
		Container	1922	
		Loop complexity	2204	
Resource Injection	99	Basic	1898	
		Address alias level	1896	
		Container	1900	
		Scope	1902	
OS Command Injection	78	Basic	2139	Test function 'system()'.
		scope	2138	

		Local control flow	2136	
		loop structure	2137	
SQL Injection	89	Basic	1797	
		Array index complexity	1799	
		Scope	1801	
		Loop structure	1930	
Stack Overflow	121	Basic	1547	
		Array index complexity	1545	
		Scope	1549	
		Basic	1566	
		Array length/limit complexity	1906	
		Basic	1602	
		Index alias level	1908	
		Loop Structure	1910	
Heap Overflow	122	Basic	2134	
		Scope	1615	
		Array index complexity	1844	
		Memory location	1848	
		Array index complexity	1574	
		Scope	1613	
Format string vulnerability	134	Scope	1562	
		Address alias level	1560	
		Local control flow	1834	
		Container	1832	
		Scope	1556	
Improper Null Termination	170	Basic	1856	
		Taint	1858	
		Buffer address type	2012	
		Container	1855	
Often Misused: String Management	251	Basic	1866	
		Taint	1874	
		Scope	1872	
		Address alias level	1868	
		Container	1870	

Hard-coded Password	259	Basic	2131	
		Local control flow	2132	
		Loop structure	2133	
		Array Address Complexity	2130	
Time-of-check Time-of-use race condition	367	Basic	1892	
		Local Control Flow	1894	
Unchecked Error Condition	391	Basic	1929	
Memory leak	401	Basic	1933	
		Scope	1586	
		Address alias level	1589	
		Container	1925	
		Loop structure	1926	
Unrestricted Critical Resource Lock	412	Basic	2205	
Double Free	415	Basic	2271	
		Loop structure	1830	
		local control flow	1828	
		Scope	1591	
Use After Free	416	Scope	2269	
		Address alias level	2270	
		Container	2135	
		Buffer address type	1914	
Uninitialized variable	457	Basic	2186	
Unintentional pointer scaling	468	Data type	1927	
Null Dereference	476	Basic	2195	
		Scope	1880	
		Address alias level	1876	
		Local control flow	2194	
Leftover Debug Code	489	Basic	1862	

4.1.3 Test Suite SCA-TS-3-C (SRD Test Suite 47)

http://samate.nist.gov/SRD/view.php?tsID=47

This test suite can be used to examine whether the tool generates weakness reports after reporting of each weakness has been suppressed for C applications.

Source Code Weakness	**CWE ID**	**Code Complexity**	**SRD Test case ID**	**Remark**
Basic XSS	80	Basic	1794	
Resource Injection	99	Basic	1897	
OS Command Injection	78	Basic	1885	Test function 'system()'.
SQL Injection	89	Basic	1796	
Stack Overflow	121	Basic	1563	
Heap Overflow	122	Basic	1611	
Format string vulnerability	134	Basic	10	
Improper Null Termination	170	Basic	1849	
Heap Inspection	244	Basic	1737	
Often Misused: String Management	251	Basic	1865	
Hard-coded Password	259	Basic	1810	
Time-of-check Time-of-use race condition	367	Basic	102	
Unchecked Error Condition	391	Basic	1928	
Memory leak	401	Basic	1585	
Unrestricted Critical Resource Lock	412	Basic	2109	
Double Free	415	Basic	2199	
Use After Free	416	Basic	2200	
Uninitialized variable	457	Data type	2019	Data type is integer
Unintentional pointer scaling	468	Basic	1782	
Null Dereference	476	Basic	2193	

Leftover Debug Code	489	Basic	1861	

4.2 *Test Suites for the C++ Language*

4.2.1 Test Suite SCA-TS-1-CPP (SRD Test Suite 57)

http://samate.nist.gov/SRD/view.php?tsID=47

This test suite tests the capability of a source code security analysis tool's handling weaknesses in the C++ language. Coupled with SCA-TS-1-C (SRD Test Suite 45), it covers the source code weaknesses in C++ listed in Annex A of *Source Code Security Analysis Tool Functional Specification* [SCA].

Source Code Weakness	**CWE ID**	**Code Complexity**	**SRD Test case ID**	**Remark**
Basic XSS	80	Basic	1965	
		Scope	1973	
		Index alias level	1975	
Resource Injection	99	Basic	2013	
		Scope	2023	
		Address alias level	2021	
		Container	2026	
OS Command Injection	78	Basic	2028	
		Scope	2030	
		Local control flow	2032	
		Loop structure	2034	
SQL Injection	89	Basic	1983	
		Array index complexity	1989	
		Scope	1985	
Stack Overflow	121	Basic	1971	
		Container	2038	
Heap Overflow	122	Basic	2062	
		Scope	2063	
		Array address complexity	2064	
		Array index complexity	2065	
Hard-coded	259	Basic	2043	

Password				
		Scope	2044	
		Loop structure	2045	
		Container	2046	
		Data Type	2048	
Unchecked Error Condition	391	Basic	1739	
Memory leak	401	Basic	2054	
		Local Control Flow	2056	
Uninitialized variable	457	Data type	2187	Data type is integer
		Data type	2060	Data type is pointer
		Loop Structure	2188	
Unintentional pointer scaling	468	Basic	1979	
Null Pointer Dereference	476	Basic	1993	
		Index alias level	1999	
		Local control flow	1997	
		Scope	1995	
Leftover Debug Code	489	Basic	2196	

4.2.2 Test Suite SCA-TS-2-CPP (SRD Test Suite 58)

http://samate.nist.gov/SRD/view.php?tsID=58

This test suite can be used to assess the false positive ratio of the tool under test for C++ applications. Coupled with SCA-TS-2-C (SRD Test Suite 46), it covers the source code weaknesses in C++ listed in Annex A of *Source Code Security Analysis Tool Functional Specification* [SCA].

Source Code Weakness	**CWE ID**	**Code Complexity**	**SRD Test case ID**	**Remark**
Basic XSS	80	Basic	1966	
		Scope	1974	
		Index alias level	1976	
Resource Injection	99	Basic	2025	
		Scope	2024	

		Address alias level	2022	
		Container	2040	
OS Command Injection	78	Basic	2029	
		Scope	2036	
		Local control flow	2033	
		Loop structure	2035	
SQL Injection	89	Basic	1984	
		Array index complexity	1990	
		Scope	1986	
Stack Overflow	121	Basic	1972	
		Container	2039	
Heap Overflow	122	Basic	2066	
		Scope	2067	
		Array address complexity	2068	
Hard-coded Password	259	Basic	2047	
		Local control flow	2050	
		Loop structure	2051	
		Container	2052	
		Data Type	2049	
Unchecked Error Condition	391	Basic	1992	
Memory leak	401	Basic	2058	
Uninitialized variable	457	Data type	2191	Data type is integer
		Data type	2061	Data type is pointer
		Loop Structure	2192	
Unintentional Pointer Scaling	468	Basic	1980	
Null Pointer Dereference	476	Basic	1994	
		Index alias level	2000	
		Local control flow	1998	
		Scope	1996	
Leftover Debug Code	489	Basic	2197	

4.2.3 Test Suite SCA-TS-3-CPP (SRD Test Suite 59)

http://samate.nist.gov/SRD/view.php?tsID=59

This test suite can be used to examine whether the tool generates weakness reports after reporting of each weakness has been suppressed for C++ applications. Coupled with SCA-TS-3-C (SRD Test Suite 47), it covers the source code weaknesses in C++ listed in Annex A of *Source Code Security Analysis Tool Functional Specification* [SCA].

Source Code Weakness	**CWE ID**	**Code Complexity**	**SRD Test case ID**	**Remark**
Basic XSS	80	Basic	1965	
Resource Injection	99	Basic	2013	
OS Command Injection	78	Basic	2028	
SQL Injection	89	Basic	1983	
Stack Overflow	121	Basic	1971	
Heap Overflow	122	Basic	2062	
Hard-coded Password	259	Basic	2043	
Unchecked Error Condition	391	Basic	1739	
Memory leak	401	Basic	2054	
Uninitialized variable	457	Data type	2187	Data type is integer
Unintentional pointer scaling	468	Basic	1979	
Null Pointer Dereference	476	Basic	1993	
Leftover Debug Code	489	Basic	2196	

4.3 Test Suites for the Java Language

4.3.1 Test Suite SCA-TS-1-JAVA (SRD Test Suite 63)

http://samate.nist.gov/SRD/view.php?tsID=63

This test suite tests the capability of a source code security analysis tool's handling of weaknesses in the Java language. It covers the source code weaknesses in Java listed in Annex A of *Source Code Security Analysis Tool Functional Specification* [SCA].

Source Code Weakness	**CWE ID**	**Code Complexity**	**SRD Test case ID**	**Remark**
Basic XSS	80	Basic	2153	
		Scope	2154	
		Container	2155	
		Loop complexity	2156	
Resource Injection	99	Basic	2088	
		Scope	2089	
		Container	2090	
OS Command Injection	78	Basic	2084	
		Scope	2085	
		Local control flow	2086	
		Loop structure	2087	
SQL Injection	89	Basic	2161	
		Array index complexity	2162	
		Scope	2163	
Hard-coded Password	259	Basic	2091	
		Local control flow	2092	
		Loop structure	2093	
		Container	2094	
		Array Index Complexity	2095	
Time-of-check Time-of-use race condition	367	Basic	2096	
Unchecked Error Condition	391	Basic	2103	
Unrestricted Critical Resource Lock	412	Basic	2104	
Null Dereference	476	Basic	2099	
		Address alias level	2105	
		Local control flow	2106	
		Scope	2107	

Leftover Debug Code	489	Basic	2098	

4.3.2 Test Suite SCA-TS-2-JAVA (SRD Test Suite 64)

http://samate.nist.gov/SRD/view.php?tsID=64

This test suite can be used to assess the false positive ratio of the tool under test for JAVA applications.

Source Code Weakness	**CWE ID**	**Code Complexity**	**SRD Test case ID**	**Remark**
Basic XSS	80	Basic	2157	
		Scope	2158	
		Container	2159	
		Loop complexity	2160	
Resource Injection	99	Basic	2114	
		Scope	2115	
		Container	2116	
OS Command Injection	78	Basic	2110	
		Scope	2111	
		Local control flow	2112	
		Loop structure	2113	
SQL Injection	89	Basic	2164	
		Array index complexity	2165	
		Scope	2166	
Hard-coded Password	259	Basic	2117	
		Local control flow	2118	
		Loop structure	2119	
		Container	2120	
		Array Index Complexity	2121	
Time-of-check Time-of-use race condition	367	Basic	2122	
Unchecked Error Condition	391	Basic	2123	
Unrestricted	412	Basic	2124	

Critical Resource Lock				
Null Dereference	476	Basic	2125	
		Address alias level	2128	
		Local control flow	2126	
		Scope	2127	
Leftover Debug Code	489	Basic	2129	

4.3.3 Test Suite SCA-TS-3-JAVA (SRD Test Suite 65)

http://samate.nist.gov/SRD/view.php?tsID=65

This test suite can be used to examine whether the tool generates weakness reports after the reporting of each weakness has been suppressed for JAVA applications.

Source Code Weakness	**CWE ID**	**Code Complexity**	**SRD Test case ID**	**Remark**
Basic XSS	80	Basic	2153	
Resource Injection	99	Basic	2088	
OS Command Injection	78	Basic	2084	
SQL Injection	89	Basic	2161	
Hard-coded Password	259	Basic	2091	
Time-of-check Time-of-use race condition	367	Basic	2096	
Unchecked Error Condition	391	Basic	2103	
Unrestricted Critical Resource Lock	412	Basic	2104	
Null Dereference	476	Basic	2099	
Leftover Debug Code	489	Basic	2098	

5 References

[CWE]	Common Weakness Enumeration – a community-developed dictionary of software weakness types, sponsored and managed by The MITRE Corporation. http://cwe.mitre.org/
[Fleiss]	Fleiss, Joseph L. (1981). *Statistical Methods for Rates and Proportions*, 2nd ed., John Wiley and Sons, New York, pp 4-8.
[SCA]	Source Code Security Analysis Tool Functional Specification Version 1.0. http://samate.nist.gov/docs/source_code_security_analysis_spec_SP500-268.pdf
[SRD]	SAMATE Reference Dataset http://samate.nist.gov/SRD/ To retrieve test suite, click on "Test Suites" on the tool bar. A list of test suite will display. Select the required test suite.