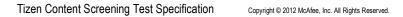


Document version 1.0.4



Copyright (c) 2013, McAfee, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of McAfee, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# 1 Contents

1	Con	tents	4
	1.1	Document History	6
	1.2	References	
	1.3	Glossary and definitions	6
2	Pur	pose and Scope	
	-	· •	
3	Con	nponent Description	8
4	Test	Environment Description	10
5	Test	Cases Specifications	11
C	5.1	Test Case TC_SEC_CS_TCSLibraryOpen_0001	
	5.2	Test Case TC_SEC_CS_TCSLibraryOpen_0002	
	5.3	Test Case TC_SEC_CS_TCSLibraryOpen_0003	
	5.4	Test Case TC_SEC_CS_TCSLibraryOpen_0004	
	5.5	Test Case TC_SEC_CS_TCSGetLastError_0001	
	5.6	Test Case TC_SEC_CS_TCSLibraryClose_0001	
	5.7	Test Case TC_SEC_CS_TCSScanData_0001	
	5.8	Test Case TC_SEC_CS_TCSScanData_0002	
	5.9	Test Case TC_SEC_CS_TCSScanData_0003	
	5.10	Test Case TC_SEC_CS_TCSScanData_0004	
	5.11	Test Case TC_SEC_CS_TCSScanData_0005	
	5.12	Test Case TC_SEC_CS_TCSScanData_0006	
	5.12	Test Case TC_SEC_CS_TCSScanData_0007	
	5.14	Test Case TC_SEC_CS_TCSScanData_0008	
	5.15	Test Case TC_SEC_CS_TCSScanData_0009	
	5.16	Test Case TC_SEC_CS_TCSScanData_0010	
	5.17	Test Case TC_SEC_CS_TCSScanData_0011	
	5.18	Test Case TC_SEC_CS_TCSScanData_0012	
	5.19	Test Case TC_SEC_CS_TCSScanData_0013	
	5.20	Test Case TC_SEC_CS_TCSScanData_0014	
	5.21	Test Case TC_SEC_CS_TCSScanData_0015	
	5.22	Test Case TC_SEC_CS_TCSScanData_0016	
	5.23	Test Case TC_SEC_CS_TCSScanData_0017	
	5.24	Test Case TC_SEC_CS_TCSScanData_0018	
	5.25	Test Case TC_SEC_CS_TCSScanData_0019	
	5.26	Test Case TC_SEC_CS_TCSScanData_0020	
	5.27	Test Case TC_SEC_CS_TCSScanData_0021	
	5.28	Test Case TC SEC CS TCSScanData 0022	
	5.29	Test Case TC_SEC_CS_TCSScanData_0023	
	5.30	Test Case TC SEC CS TCSScanData 0024	39
	5.31	Test Case TC_SEC_CS_TCSScanData_0025	
	5.32	Test Case TC_SEC_CS_TCSScanData_0026	
	5.33	Test Case TC_SEC_CS_TCSScanData_0027	
	5.34	Test Case TC_SEC_CS_TCSScanData_0028	
	5.35	Test Case TC_SEC_CS_TCSScanData_0029	
	5.36	Test Case TC_SEC_CS_TCSScanData_0030	
	5.37	Test Case TC_SEC_CS_TCSScanData_0031	
	5.38	Test Case TC_SEC_CS_TCSScanData_0032	47
	5.39	Test Case TC_SEC_CS_TCSScanData_0033	48
	5.40	Test Case TC_SEC_CS_TCSScanData_0034	
	5.41	Test Case TC_SEC_CS_TCSScanData_0035	50

5	5.42 Test Case TC_SEC_CS_TCSScanData_003	6
5	5.43 Test Case TC_SEC_CS_TCSScanData_003	7
5		8
5		9
5		0
5		1
5		2
5		3
		4
		5
	5.52 Test Case TC_SEC_CS_TCSScanData_004	6
		7
		8
		9
		0
		1
		2
	5.73 Test Case TC_SEC_CS_TCSScanFile 0015	
5	5.84 Test Case TC_SEC_CS_TCSScanFile_0026	
5	5.85 Test Case TC_SEC_CS_TCSScanFile_0027	
5	5.86 Test Case TC_SEC_CS_TCSScanFile_0028	
5	5.87 Test Case TC_SEC_CS_TCSScanFile_0029	
5	5.88 Test Case TC_SEC_CS_TCSScanFile_0030	
5		
5		
5		
5	5.92 Test Case TC_SEC_CS_TCSScanFile_0034	
6	Test Guide	
7	Test Contents	

### 1.1 Document History

Version	Date	Reason	
1.0.0	11/05/2012	First draft from McAfee	
1.0.1	11/07/2012	Added more test cases for stub funtions	
1.0.2	11/08/2012	Correct some test statement and wording	
1.0.3	11/12/2012	Add library replacement test cases, add test contents and test guide.	
1.0.4	01/26/2013	Add license	

### 1.2 References

Ref	Document	Issue	Title
[1]	Tizen Content Screening API Specification	1.0.2	Tizen Content Screening API Specification

### 1.3 Glossary and definitions

- API Application Programming Interface
- TCS Tizen Content Screening

## 2 Purpose and Scope

The overall purpose of this document is to describe the conformance test cases for the Tizen Content Screening framework.

This document shall include:

- 1. Tizen Content Screening Test Configuration
- 2. Test Case procedures

The scope of this document is the Tizen Content Screening Foundation API functions that are common to all Content Screening implementations. Specific functions of the Content Screening plug-in are not tested. All TCS implementations must include and meet the test cases defined in this document.

TCS validation plug-in

• A security plug-in for Tizen Content Screening Framework validation. Includes the functionalities required for the validation, including scanning, and conforms to the TCS framework API specification.

# **3** Component Description

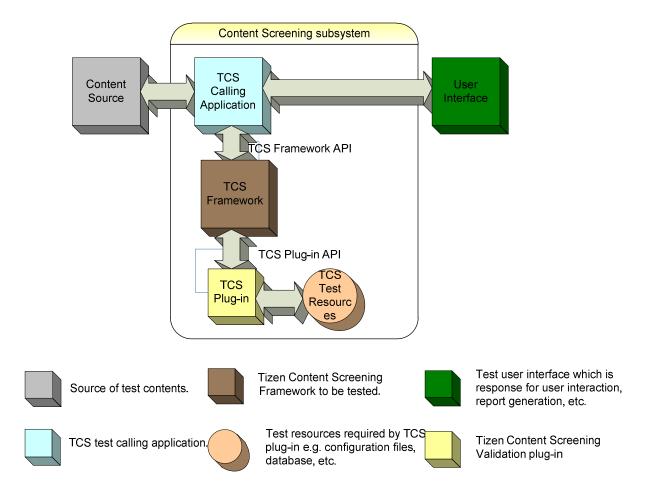


Figure 1: Tiezn Content Screening Architecture

The TCS framework (here on will be referenced as "tizen content screening library", "TCS library") works (interacts) with the calling application through an interface identified as one of the main elements to be tested in this test specification.

TCS plug-in is the content screening function implementation interfacing the TCS framework via Tizen Content screening Framework API functions.

"TCS Test Resources" is the resource data used by the TCS plug-in for test purposes (e.g. configurations, signatures for test content, etc.).

For testing purposes, the TCS library can be interchanged with a test tool. Rather than using software to analyze the content from the calling application and return the result of the scanning, a test tool is used to return the desired result matching the input content and the test case under execution. The test tool should also analyze the request from the calling application implementation to check that the process and the implementation is successful in both of the following ways:

- 1. The input content received from the calling application triggers the scanning process according to the content type (the request to the engine/test tool could be different if the content is an e-mail, a HTML document, a binary file, etc.).
- 2. The result of the scanning APIs must be understood by the calling application which should take an action with the received content:
  - a) Do nothing if the content is correct, or
  - b) Request more information from the TCS library (by the test tool).

This test tool can generate a log file with the result of the performed tests for checking purposes.

# 4 Test Environment Description

The test environment used is on Tizen platform.

The following requirements apply to all test cases defined in this document:

- 1. Any resources required by Tizen Content Screening subsystem in runtime should be installed in the test environment.
- 2. Test samples required by test suite should be installed in the test environment.

# 5 Test Cases Specifications

# 5.1 Test Case TC\_SEC\_CS\_TCSLibraryOpen\_0001

TC_SEC_CS_TCSLibraryOpen_0001 TCS library interface initialization test.		
API Function(s) covered:		
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>		
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>		
Test Objectives:		
This test case verifies that the calling application can correctly initialize the TCS library handle.		
Test pre-conditions:		
validation plug-in		
Test Procedure:		
1. Call TCSLibraryOpen().		
2. Verify the API return value.		
Test PASS Condition:		
Step 2 should return valid TCSLIB_HANDLE instead of INVALID_TCSLIB_HANDLE.		
Test Clean-up procedure:		
Call TCSLibraryClose() with the TCS library handle returned by TCSLibraryOpen().		

## 5.2 Test Case TC\_SEC\_CS\_TCSLibraryOpen\_0002

TC_SEC_CS_TCSLibraryOpen_0002	TCS library interface initialization test.	
API Function(s) covered:		
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>		
Test Objectives:		
This test case verifies that the calling application can get proper error when there is no TCS plugin found in system.		
Test pre-conditions:		
Stub functions		
Test Procedure:		
1. Call TCSLibraryOpen().		
2. Verify it returns INVALID_TCSLIB_HANDLE.		

TC_SEC_CS_TCSLibraryOpen_0002	TCS library interface initialization test.	
Test PASS Condition:		
Step 2 should return valid INVALID_TCSLIB_HANDLE.		
Test Clean-up procedure:		
None.		

### 5.3 Test Case TC\_SEC\_CS\_TCSLibraryOpen\_0003

TC_SEC_CS_TCSLibraryOpen_0003 TCS library replacement test.		
API Function(s) covered:		
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>		
<pre>int TCSLibraryClose(void);</pre>		
Test Objectives:		
This test case verifies that the calling application can get always get the latest TCS library API call after close/open.		
Test pre-conditions:		
Stub functions		
Test Procedure:		
1. Call TCSLibraryOpen().		
2. Verify it returns INVALID_TCSLIB_HANDLE.		
3. Copy validation plug-in to "/opt/usr/share/sec_plugin"		
<ol> <li>Call TCSLibraryOpen().</li> </ol>		
5. Verify it returns valid TCS library handle.		
6. Call TCSLibraryClose().		
Test PASS Condition:		
Step 2 should pass.		
Step 5 should pass.		
Test Clean-up procedure:		
None.		

## 5.4 Test Case TC\_SEC\_CS\_TCSLibraryOpen\_0004

TC\_SEC\_CS\_TCSLibraryOpen\_0004

TCS library replacement test.

TC_SEC_CS_TCSLibraryOpen_0004 TCS library replacement test.		
API Function(s) covered:		
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>		
<pre>int TCSLibraryClose(void);</pre>		
Test Objectives:		
This test case verifies that the calling application can get always get the latest TCS library API call after close/open.		
Test pre-conditions:		
validation plug-in		
Test Procedure:		
1. Call TCSLibraryOpen().		
2. Verify it returns valid TCS library handle.		
3. Delete validation plug-in from "/opt/usr/share/sec_plugin"		
4. Call TCSLibraryClose().		
5. Call TCSLibraryOpen().		
6. Verify it returns INVALID_TCSLIB_HANDLE.		
Test PASS Condition:		
Step 2 should pass.		
Step 6 should pass.		
Test Clean-up procedure:		
None.		

# 5.5 Test Case TC\_SEC\_CS\_TCSGetLastError\_0001

TC_SEC_CS_TCSGetLastError_0001	Stub TCS function error return.	
API Function(s) covered:		
<pre>int TCSGetLastError(TCSLIB_HANDLE hLib);</pre>		
<u>Test Objectives:</u> This test case verifies that the calling application can get proper error code from TCS stub functions.		
Test pre-conditions:		
Stub functions		
Test Procedure:		

TC_SEC_CS_TCSGetLastError_0001	Stub TCS function error return.		
1. Call TCSGetLastError() with INVALID_TCS	LIB_HANDLE.		
2. Verify it returns TCS_ERROR_NOT_IMPLEMENT	2. Verify it returns TCS_ERROR_NOT_IMPLEMENTED.		
Test PASS Condition:			
Step 2 should passed.			
Test Clean-up procedure:			
None.			

# 5.6 Test Case TC\_SEC\_CS\_TCSLibraryClose\_0001

TC_SEC_CS_TCSLibraryClose_0001 TCS library interface finalization.	
API Function(s) covered:	
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>	
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>	
Test Objectives:	
This test case verifies that the calling application can close the TCS library handle.	
Test pre-conditions:	
validation plug-in.	
Test Procedure:	
1. Call TCSLibraryOpen().	
2. Verify that the API returns valid TCSLIB_HANDLE instead of INVALID_TCSLIB_HANDLE.	
3. Call TCSLibraryClose() with the TCS library handle returned by TCSLibraryOpen().	
4. Verify that the return value of the TCSLibraryClose() is 0.	
Test PASS Condition:	
Step 2 should pass verification.	
Step 4 should pass verification.	
Test Clean-up procedure:	
No specific cleanup required.	

### 5.7 Test Case TC\_SEC\_CS\_TCSScanData\_0001

### TC\_SEC\_CS\_TCSScanData\_0001Call TCS interface to scan benign content data.

#### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanData(TCSLIB\_HANDLE hLib, TCSScanParam \*pParam,

TCSScanResult \*pResult);

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

#### **Test Objectives:**

This test case tests the scan request interface and verifies that the TCS interface returns the expected return value in the case of benign content data.

#### Test pre-conditions:

validation plug-in.

#### Test Procedure:

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with benign data, TCS\_SA\_SCANONLY as the scan action ID, TCS\_DTYPE\_UNKNOWN as the data type identifier and set pfCallback to NULL.
- 3. Verify that the return value of TCSScanData() is 0.
- 4. Verify that the number of the detected malware is 0.
- 5. Call pfFreeResult() to release the resource returned by TCS library.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

### 5.8 Test Case TC\_SEC\_CS\_TCSScanData\_0002

### TC\_SEC\_CS\_TCSScanData\_0002Call TCS interface to scan benign content data.

#### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanData(TCSLIB\_HANDLE hLib, TCSScanParam \*pParam,

TCSScanResult \*pResult);

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

#### **Test Objectives:**

This test case tests the scan request interface and verifies that the TCS interface returns the expected return value in the case of benign content data.

#### Test pre-conditions:

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with benign, TCS\_SA\_SCANONLY as the scan action ID, TCS\_DTYPE\_UNKNOWN as the data type identifier and pfCallback is not NULL.
- 3. Verify that the pfCallback is not called.
- 4. Verify that the return value of TCSScanData() is 0.
- 5. Verify that the number of the detected malware is 0.
- 6. Call pfFreeResult () to release the resource returned by TCS library.

7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

### 5.9 Test Case TC\_SEC\_CS\_TCSScanData\_0003

### TC\_SEC\_CS\_TCSScanData\_0003 Call TCS interface to scan infected content data.

#### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanData(TCSLIB\_HANDLE hLib, TCSScanParam \*pParam,

TCSScanResult \*pResult);

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

#### **Test Objectives:**

This test case verifies that the expected return value is returned when the TCS interface is called to scan infected content data

#### Test pre-conditions:

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with infected data, TCS\_SA\_SCANONLY as the scan action ID, TCS\_DTYPE\_UNKNOWN as the data type identifier and set pfCallback to NULL.
- 3. Verify that the return value of TCSScanData() is 0.
- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call pfFreeResult () to release the resource returned by TCS library.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

### 5.10 Test Case TC\_SEC\_CS\_TCSScanData\_0004

### TC\_SEC\_CS\_TCSScanData\_0004Call TCS interface to scan infected content data.

#### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanData(TCSLIB\_HANDLE hLib, TCSScanParam \*pParam,

TCSScanResult \*pResult);

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

#### **Test Objectives:**

This test case verifies that the expected return value is returned when the TCS interface is called to scan infected content data

#### Test pre-conditions:

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with infected data, TCS\_SA\_SCANONLY as the scan action ID, TCS\_DTYPE\_UNKNOWN as the data type identifier and where pfCallback is not NULL.
- 3. Verify that pfCallback is called and that the malware name or variant name is as expected and the severity/behaviour is as expected.
- 4. Verify that the return value of TCSScanData() is 0.
- 5. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 6. Call pfFreeResult () to release the resource returned by TCS library.

7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

# 5.11 Test Case TC\_SEC\_CS\_TCSScanData\_0005

TC_SEC_CS_TCSScanData_0005 Call TCS interface to scan benign HTML formatted content data.		
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void);		
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
TCSScanResult *pResult);		
int TCSLibraryClose(TCSLIB_HANDLE hLib);		
Test Objectives:		
This test case verifies that the TCS interface returns the expected return value when it is called to scan benign HTML formatted content data		
Test pre-conditions:		
For validation plug-in only.		
Test Procedure:		
1. Call TCSLibraryOpen().		
<ol> <li>Call TCSScanData() with a buffer filled with benign HTML formatted data, TCS_SA_SCANONLY as the scan action ID, and TCS_DTYPE_HTML as the data type identifier. Set pfCallback to NULL.</li> </ol>		
3. Verify that the return value of TCSScanData() is 0.		
4. Verify that the number of the detected malware is 0.		
5. Call pfFreeResult() to release the resource returned by TCS library.		
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().		
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Test Clean-up procedure:		

# 5.12Test Case TC\_SEC\_CS\_TCSScanData\_0006

TC_SEC_CS_TCSScanData_0006 Call TCS interface to scan benign HTML formatted content data.		
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void);		
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
TCSScanResult *pResult);		
int TCSLibraryClose(TCSLIB_HANDLE hLib);		
Test Objectives:		
This test case verifies that the TCS interface returns the expected return value when it is called to scan benign HTML formatted content data		
Test pre-conditions:		
For validation plug-in only.		
Test Procedure:		
1. Call TCSLibraryOpen().		
2. Call TCSScanData() with a buffer filled with benign HTML formatted data, TCS_SA_SCANONLY as the scan action ID, TCS_DTYPE_HTML as the data type identifier and pfCallback is not NULL.		
3. Verify that pfCallback is not called.		
4. Verify that the return value of TCSScanData() is 0.		
5. Verify that the number of the detected malware is 0.		
6. Call pfFreeResult() to release the resource returned by TCS library.		
7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().		
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Step 5 should pass verification.		
Test Clean-up procedure:		

### 5.13Test Case TC\_SEC\_CS\_TCSScanData\_0007

TC_SEC_CS_TCSScanData_	_0007 Call TCS interface to scan infected HTML formatted content data.
API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryO	<pre>open(void);</pre>
int TCSScanData(TCSLIB_HA	NDLE hLib, TCSScanParam *pParam,
TCSScanRe	esult *pResult);
int TCSLibraryClose(TCSLI	B_HANDLE hLib);
Test pre-conditions: For validation plug-in only.	
Test Procedure:	
1. Call TCSLibraryOpen()	
	h a buffer filled with infected HTML formatted data, TCS_SA_SCANONLY as the E_HTML as the data type identifier and set pfCallback to NULL.
3. Verify that the return value of	of TCSScanData() is 0.
4 Varify that the number of the	e detected malware is as expected, the malware name or variant name is as

- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call  ${\tt pfFreeResult}$  ( ) to release the resource returned by TCS library.

6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

# 5.14Test Case TC\_SEC\_CS\_TCSScanData\_0008

TC_SE	C_CS_TCSScanData_0008	Call TCS interface to scan infected HTML formatted content data.		
<u>API Fu</u>	API Function(s) covered:			
TCSLIE	TCSLIB_HANDLE TCSLibraryOpen(void);			
int TC	int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,			
	TCSScanResult *pR	esult);		
int TC	CSLibraryClose(TCSLIB_HANDLE	hLib);		
Test Ob	jectives:			
	case verifies that the expected return va d content data.	lue is returned when the TCS interface is called to scan infected HTML		
<u>Test pre</u>	e-conditions:			
For valie	lation plug-in only.			
Test Pro	ocedure:			
1.	Call TCSLibraryOpen().			
2.	<ol> <li>Call TCSScanData() with a buffer filled with infected HTML formatted data, TCS_SA_SCANONLY as the scan action ID, TCS_DTYPE_HTML as the data type identifier and where pfCallback is not NULL.</li> </ol>			
3.	Verify that pfCallback is called, the n is as expected.	malware name or variant name is as expected and the severity/behaviour		
4.	Verify that the return value of TCSScan	Data() is 0.		
5.	Verify that the number of the detected m and the severity/behaviour is as expected	alware is as expected, the malware name or variant name is as expected l.		
6.	Call pfFreeResult() to release the r	resource returned by TCS library.		
7.	Call TCSLibraryClose() with the T	CS library handle returned by the TCSLibraryOpen().		
Test PA	SS Condition:			
Step 3 sl	nould pass verification.			
Step 4 should pass verification.				
Step 5 should pass verification.				
Test Ck	Test Clean un procedure:			

### Test Clean-up procedure:

# 5.15Test Case TC\_SEC\_CS\_TCSScanData\_0009

TC_SEC_CS_TCSScanData_0009	Call TCS interface to scan benign URL formatted content data.	
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void	);	
int TCSScanData(TCSLIB_HANDLE hLi	b, TCSScanParam *pParam,	
TCSScanResult *pR	esult);	
int TCSLibraryClose(TCSLIB_HANDLE	hLib);	
Test Objectives:		
This test case verifies that the expected value is reformatted content data.	eturned from the interface when it is called to scan benign URL	
Test pre-conditions:		
For validation plug-in only.		
Test Procedure:		
1. Call TCSLibraryOpen().		
	lled with benign URL formatted data, TCS_SA_SCANONLY as the as the data type identifier. Set pfCallback to NULL.	
3. Verify that the return value of TCSScar	nData() is 0.	
4. Verify that the number of the detected m	nalware is 0.	
5. Call pfFreeResult() to release the	resource returned by TCS library.	
6. Call TCSLibraryClose() with the T	TCS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Test Clean-up procedure:		

# 5.16Test Case TC\_SEC\_CS\_TCSScanData\_0010

TC_SEC_CS_TCSScanData_0010	Call TCS interface to scan benign URL formatted content data.
API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryOpen(void)	);
int TCSScanData(TCSLIB_HANDLE hLik	o, TCSScanParam *pParam,
TCSScanResult *pRe	esult);
int TCSLibraryClose(TCSLIB_HANDLE	hLib);
Test Objectives:	
This test case verifies that the expected value is reformatted content data.	turned from the interface when it is called to scan benign URL
Test pre-conditions:	
For validation plug-in only.	
Test Procedure:	
1. Call TCSLibraryOpen().	
	ed with benign URL formatted data, TCS_SA_SCANONLY as the edata type identifier and where pfCallback is not NULL.
3. Verify that pfCallback is not called.	
4. Verify that the return value of TCSScan	Data() is 0.
5. Verify that the number of the detected ma	alware is 0.
6. Call pfFreeResult() to release the rele	esource returned by TCS library.
7. Call TCSLibraryClose() with the T	CS library handle returned by the TCSLibraryOpen().
Test PASS Condition:	
Step 3 should pass verification.	
Step 4 should pass verification.	
Step 5 should pass verification.	
Test Clean-up procedure:	
No specific cleanup required.	

# 5.17Test Case TC\_SEC\_CS\_TCSScanData\_0011

TC_SEC_CS_TCSScanData_0011 Call TCS interface to scan infected URL formatted content data.
API Function(s) covered:
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,
TCSScanResult *pResult);
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>
Test Objectives:         This test case verifies that the expected return value is returned when the interface is called to scan infected URL formatted content data.
Test pre-conditions:         For validation plug-in only.         Test Procedure:
1. Call TCSLibraryOpen().
2. Call TCSScanData() with a buffer filled with infected URL formatted data, TCS_SA_SCANONLY as the scan action ID, and TCS_DTYPE_URL as the data type identifier. Set pfCallback to NULL.
3. Verify that the return value of TCSScanData() is 0.
4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
5. Call pfFreeResult() to release the resource returned by TCS library.
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().
Test PASS Condition:
Step 3 should pass verification.
Step 4 should pass verification.
Test Clean-up procedure:

# 5.18Test Case TC\_SEC\_CS\_TCSScanData\_0012

TC_SE	C_CS_TCSScanData_0012	Call TCS interface to scan infected URL formatted content data.	
API Fun	action(s) covered:		
TCSLIB	TCSLIB_HANDLE TCSLibraryOpen(void);		
int TC	int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
	TCSScanResult *p	Result);	
int TC	SLibraryClose(TCSLIB_HANDL	E hLib);	
Test Ob	jectives:		
	case verifies that the expected return va d content data.	alue is returned when the interface is called to scan infected URL	
<u>Test pre</u>	-conditions:		
For valid	lation plug-in only.		
<u>Test Pro</u>	cedure:		
1.	Call TCSLibraryOpen().		
	Call TCSScanData() with a buffer filled with infected URL formatted data, TCS_SA_SCANONLY as the scan action ID, TCS_DTYPE_URL as the data type identifier and where pfCallback is not NULL.		
	. Verify that pfCallback is called, the malware name or variant name is as expected and the severity/behaviour is as expected.		
4.	Verify that the return value of TCSScanData() is 0.		
	Verify that the number of the detected expected and the severity/behaviour is	malware is as expected, the malware name or variant name is as as expected.	
6.	Call pfFreeResult() to release the	e resource returned by TCS library.	
7.	Call TCSLibraryClose() with the	TCS library handle returned by the TCSLibraryOpen().	
Test PA	SS Condition:		
Step 3 should pass verification.			
Step 4 should pass verification.			
Step 5 should pass verification.			
Test Clean-up procedure:			
No specific cleanup required.			

# 5.19Test Case TC\_SEC\_CS\_TCSScanData\_0013

TC_S	EC_CS_TCSScanData_0013 Call TCS interface to scan benign Email formatted content data.
<u>API Fu</u>	nction(s) covered:
TCSLI	B_HANDLE TCSLibraryOpen(void);
int T	CSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,
	TCSScanResult *pResult);
int T	CSLibraryClose(TCSLIB_HANDLE hLib);
Test Ol	bjectives:
	t case verifies that the expected return value is returned when the interface is called to scan benign Email ed content data.
	dation plug-in only. <b>ocedure:</b>
1.	
2.	Call TCSScanData() with a buffer filled with benign Email formatted data, TCS_SA_SCANONLY as the scan action ID, and TCS_DTYPE_EMAIL as the data type identifier. Set pfCallback to NULL.
3.	Verify that the return value of TCSScanData() is 0.
4.	Verify that the number of the detected malware is 0.
5.	Call pffreeResult() to release the resource returned by TCS library.
6.	Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().
<u>Test P</u> A	ASS Condition:
Step 3 s	hould pass verification.
	hould pass verification.

Test Clean-up procedure:

### 5.20 Test Case TC\_SEC\_CS\_TCSScanData\_0014

TC_S	EC_CS_TCSScanData_0014	Call TCS interface to scan benign Email formatted content data.
<u>API Fu</u>	nction(s) covered:	
TCSLI	B_HANDLE TCSLibraryOpen(void)	;
int T	CSScanData(TCSLIB_HANDLE hLik	o, TCSScanParam *pParam,
	TCSScanResult *pRe	esult);
int T	CSLibraryClose(TCSLIB_HANDLE	hLib);
	e-conditions: dation plug-in only.	
	ocedure:	
1.	Call TCSLibraryOpen().	
2.		ed with benign Email formatted data, TCS_SA_SCANONLY as the the data type identifier and where pfCallback is not NULL.
3.	Verify that pfCallback is not called.	
4.	Verify that the return value of TCSScan	Data() is 0.
_		

- 5. Verify that the number of the detected malware is 0.
- 6. Call pfFreeResult() to release the resource returned by TCS library.
- 7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

### 5.21 Test Case TC\_SEC\_CS\_TCSScanData\_0015

TC_S	EC_CS_TCSScanData_0015	Call TCS interface to scan infected Email formatted content data.	
API Fu	nction(s) covered:		
TCSLI	TCSLIB_HANDLE TCSLibraryOpen(void);		
int T	CSScanData(TCSLIB_HANDLE hLi	b, TCSScanParam *pParam,	
	TCSScanResult *pR	esult);	
int T	CSLibraryClose(TCSLIB_HANDLE	hLib);	
Test O	<u>pjectives:</u>		
	t case verifies that the expected return val ed content data.	ue is returned when the interface is called to scan infected Email	
<u>Test pr</u>	e-conditions:		
For val	dation plug-in only.		
<u>Test Pr</u>	ocedure:		
1.	Call TCSLibraryOpen().		
2.		lled with infected Email formatted data, TCS_SA_SCANONLY as the IL as the data type identifier. Set pfCallback to NULL.	

- 3. Verify that the return value of TCSScanData() is 0.
- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call pfFreeResult () to release the resource returned by TCS library.

6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

### 5.22Test Case TC\_SEC\_CS\_TCSScanData\_0016

TC\_SEC\_CS\_TCSScanData\_0016

formatted content data.
API Function(s) covered:
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,
<pre>TCSScanResult *pResult);</pre>
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>
This test case verifies that the expected return value is returned when the interface is called to scan infected Email formatted content data.
Test pre-conditions:
For validation plug-in only.
Test Procedure:
1. Call TCSLibraryOpen().
2. Call TCSScanData() with a buffer filled with infected Email formatted data, TCS_SA_SCANONLY as the scan action ID, TCS_DTYPE_EMAIL as the data type identifier and where pfCallback is not NULL.

Call TCS interface to scan infected Email

- 3. Verify that pfCallback is called, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 4. Verify that the return value of TCSScanData() is 0.
- 5. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 6. Call pfFreeResult() to release the resource returned by TCS library.
- 7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

# 5.23 Test Case TC\_SEC\_CS\_TCSScanData\_0017

<pre>API Function(s) covered: TCSLIB_HANDLE TCSLibraryOpen(void); int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,</pre>	TC_SEC_CS_TCSScanData_0017	Call TCS interface to scan benign phone number formatted content data.			
<pre>int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,</pre>	API Function(s) covered:				
<pre>TCSScanResult *pResult); int TCSLibraryClose(TCSLIB_HANDLE hLib); Test Objectives: This test case verifies that the expected return value is returned when the interface is called to scan benign phone number formatted content data. Test pre-conditions: For validation plug-in only. Test Procedure: 1. Call TCSLibraryOpen(). 2. Call TCSLibraryOpen(). 3. Call TCSScanData() with a buffer filled with benign phone number formatted data, TCS_SA_SCANONI as the scan action ID, and TCS_DTYPE_PHONE as the data type identifier. Set pfCallback to NULL. 3. Verify that the return value of TCSScanData() is 0. 4. Verify that the return value of TCSScanData() is 0. 5. Call pfFreeResult() to release the resource returned by TCS library. 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen(). Test PASS Condition: Step 3 should pass verification.</pre>					
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib); Test Objectives: This test case verifies that the expected return value is returned when the interface is called to scan benign phone number formatted content data. Test pre-conditions: For validation plug-in only. Test Procedure: 1. Call TCSLibraryOpen(). 2. Call TCSScanData() with a buffer filled with benign phone number formatted data, TCS_SA_SCANOND as the scan action ID, and TCS_DTYPE_PHONE as the data type identifier. Set pfCallback to NULL. 3. Verify that the return value of TCSScanData() is 0. 4. Verify that the return value of TCSScanData() is 0. 5. Call pfFreeResult() to release the resource returned by TCS library. 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen(). Test PASS Condition: Step 3 should pass verification.</pre>	int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,				
Test Objectives:         This test case verifies that the expected return value is returned when the interface is called to scan benign phone number formatted content data.         Test pre-conditions:         For validation plug-in only.         Test Procedure:         1. Call TCSLibraryOpen().         2. Call TCSScanData() with a buffer filled with benign phone number formatted data, TCS_SA_SCANONI as the scan action ID, and TCS_DTYPE_PHONE as the data type identifier. Set pfCallback to NULL.         3. Verify that the return value of TCSScanData() is 0.         4. Verify that the number of the detected malware is 0.         5. Call pfFreeResult() to release the resource returned by TCS library.         6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().         Test PASS Condition:         Step 3 should pass verification.	TCSScanResult *pResult);				
This test case verifies that the expected return value is returned when the interface is called to scan benign phone number formatted content data.          Test pre-conditions:         For validation plug-in only.         Test Procedure:         1. Call TCSLibraryOpen().         2. Call TCSScanData() with a buffer filled with benign phone number formatted data, TCS_SA_SCANONNA as the scan action ID, and TCS_DTYPE_PHONE as the data type identifier. Set pfCallback to NULL.         3. Verify that the return value of TCSScanData() is 0.         4. Verify that the number of the detected malware is 0.         5. Call pfFreeResult() to release the resource returned by TCS library.         6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().         Test PASS Condition:         Step 3 should pass verification.	int TCSLibraryClose(TCSLIB_HANDLE hLib);				
<pre>number formatted content data. Test pre-conditions: For validation plug-in only. Test Procedure: 1. Call TCSLibraryOpen(). 2. Call TCSScanData() with a buffer filled with benign phone number formatted data, TCS_SA_SCANONI as the scan action ID, and TCS_DTYPE_PHONE as the data type identifier. Set pfCallback to NULL. 3. Verify that the return value of TCSScanData() is 0. 4. Verify that the number of the detected malware is 0. 5. Call pfFreeResult() to release the resource returned by TCS library. 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen(). Test PASS Condition: Step 3 should pass verification.</pre>	Test Objectives:				
<ul> <li>For validation plug-in only.</li> <li>Test Procedure: <ol> <li>Call TCSLibraryOpen().</li> <li>Call TCSScanData() with a buffer filled with benign phone number formatted data, TCS_SA_SCANONE as the scan action ID, and TCS_DTYPE_PHONE as the data type identifier. Set pfCallback to NULL.</li> <li>Verify that the return value of TCSScanData() is 0.</li> <li>Verify that the number of the detected malware is 0.</li> <li>Call pfFreeResult() to release the resource returned by TCS library.</li> <li>Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().</li> </ol> </li> <li>Test PASS Condition: Step 3 should pass verification.</li></ul>					
<ol> <li>Test Procedure:         <ol> <li>Call TCSLibraryOpen().</li> <li>Call TCSScanData() with a buffer filled with benign phone number formatted data, TCS_SA_SCANONI as the scan action ID, and TCS_DTYPE_PHONE as the data type identifier. Set pfCallback to NULL.</li> <li>Verify that the return value of TCSScanData() is 0.</li> <li>Verify that the number of the detected malware is 0.</li> <li>Call pfFreeResult() to release the resource returned by TCS library.</li> <li>Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().</li> </ol> </li> <li>Test PASS Condition:         Step 3 should pass verification.         </li> </ol>	Test pre-conditions:				
<ol> <li>Call TCSLibraryOpen().</li> <li>Call TCSScanData() with a buffer filled with benign phone number formatted data, TCS_SA_SCANONI as the scan action ID, and TCS_DTYPE_PHONE as the data type identifier. Set pfCallback to NULL.</li> <li>Verify that the return value of TCSScanData() is 0.</li> <li>Verify that the number of the detected malware is 0.</li> <li>Call pfFreeResult() to release the resource returned by TCS library.</li> <li>Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().</li> </ol> Test PASS Condition: Step 3 should pass verification.	For validation plug-in only.				
<ol> <li>Call TCSScanData() with a buffer filled with benign phone number formatted data, TCS_SA_SCANONI as the scan action ID, and TCS_DTYPE_PHONE as the data type identifier. Set pfCallback to NULL.</li> <li>Verify that the return value of TCSScanData() is 0.</li> <li>Verify that the number of the detected malware is 0.</li> <li>Call pfFreeResult() to release the resource returned by TCS library.</li> <li>Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().</li> </ol> Test PASS Condition: Step 3 should pass verification.	Test Procedure:				
<ul> <li>as the scan action ID, and TCS_DTYPE_PHONE as the data type identifier. Set pfCallback to NULL.</li> <li>3. Verify that the return value of TCSScanData() is 0.</li> <li>4. Verify that the number of the detected malware is 0.</li> <li>5. Call pfFreeResult() to release the resource returned by TCS library.</li> <li>6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().</li> </ul> Test PASS Condition: Step 3 should pass verification.	1. Call TCSLibraryOpen().				
<ul> <li>4. Verify that the number of the detected malware is 0.</li> <li>5. Call pfFreeResult() to release the resource returned by TCS library.</li> <li>6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().</li> </ul> Test PASS Condition: Step 3 should pass verification.					
<ol> <li>Call pfFreeResult() to release the resource returned by TCS library.</li> <li>Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().</li> </ol> Test PASS Condition: Step 3 should pass verification.	3. Verify that the return value of TCSScar	3. Verify that the return value of TCSScanData() is 0.			
<ul> <li>6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().</li> <li><u>Test PASS Condition:</u> Step 3 should pass verification.</li> </ul>	4. Verify that the number of the detected malware is 0.				
Test PASS Condition: Step 3 should pass verification.	5. Call pfFreeResult() to release the	resource returned by TCS library.			
Step 3 should pass verification.	6. Call TCSLibraryClose() with the	TCS library handle returned by the TCSLibraryOpen().			
	Test PASS Condition:				
Step 4 should pass verification	Step 3 should pass verification.				
Step 4 should pass verification.	Step 4 should pass verification.				

### 5.24Test Case TC\_SEC\_CS\_TCSScanData\_0018

TC_SEC_CS_TCSScanData_0018	Call TCS interface to scan benign phone number formatted content data.		
API Function(s) covered:			
TCSLIB_HANDLE TCSLibraryOpen(void)	);		
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,			
<pre>TCSScanResult *pResult);</pre>			
int TCSLibraryClose(TCSLIB_HANDLE hLib);			
Test Objectives:			
This test case verifies that the expected return value is returned when the interface is called to scan benign phone number formatted content data.			
Test pre-conditions:			
For validation plug-in only.			
Test Procedure:			

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with benign phone number formatted data, TCS\_SA\_SCANONLY as the scan action ID, TCS\_DTYPE\_PHONE as the data type identifier and where pfCallback is not NULL.
- 3. Verify that pfCallback is not called.
- 4. Verify that the return value of TCSScanData() is 0.
- 5. Verify that the number of the detected malware is 0.
- 6. Call pfFreeResult () to release the resource returned by TCS library.
- 7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

# 5.25 Test Case TC\_SEC\_CS\_TCSScanData\_0019

TC_S	EC_CS_TCSScanData_0019	Call TCS interface to scan infected phone number formatted content data.	
<u>API Fu</u>	unction(s) covered:		
TCSLI	<pre>ICSLIB_HANDLE TCSLibraryOpen(void);</pre>		
int T	int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
	TCSScanResult *pResult);		
int T	int TCSLibraryClose(TCSLIB_HANDLE hLib);		
formatted content data.         Test pre-conditions:         For validation plug-in only.			
For val	idation plug-in only.		
For vali			
For vali Test Pr 1.	idation plug-in only. cocedure: Call TCSLibraryOpen(). Call TCSScanData() with a buffer fill	ed with infected phone number formatted data, TCS_SA_SCANONLY PHONE as the data type identifier. Set pfCallback to NULL.	
For vali Test Pr 1.	idation plug-in only. cocedure: Call TCSLibraryOpen(). Call TCSScanData() with a buffer fill	PHONE as the data type identifier. Set pfCallback to NULL.	
For vali Test Pr 1. 2.	idation plug-in only. Tocedure: Call TCSLibraryOpen(). Call TCSScanData() with a buffer fill as the scan action ID, and TCS_DTYPE_ Verify that the return value of TCSScan	PHONE as the data type identifier. Set pfCallback to NULL. Data() is 0. Ilware is as expected, the malware name or variant name is as	

6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

### 5.26 Test Case TC\_SEC\_CS\_TCSScanData\_0020

TC_S	EC_CS_TCSScanData_0020	Call TCS interface to scan infected phone number formatted content data.			
<u>API Fu</u>	API Function(s) covered:				
TCSLI	TCSLIB_HANDLE TCSLibraryOpen(void);				
int T	int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,				
	TCSScanResult *pResult);				
int T	CSLibraryClose(TCSLIB_HANDLE	hLib);			
This test case verifies that the expected value is returned when the interface is called to scan infected phone number formatted content data.           Test pre-conditions:           For validation plug-in only.					
Test Procedure:					
1.	Call TCSLibraryOpen().				
2.		ed with infected phone number formatted data, TCS_SA_SCANONLY NE as the data type identifier and where pfCallback is not NULL.			
3.	Verify that pfCallback is called, the n severity/behaviour is as expected.	nalware name or variant name is as expected and the			

- 4. Verify that the return value of TCSScanData() is 0.
- 5. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 6. Call pfFreeResult() to release the resource returned by TCS library.
- 7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

# 5.27Test Case TC\_SEC\_CS\_TCSScanData\_0021

TC_SEC_CS_TCSScanData_0021	Call TCS interface to scan benign Java code formatted content data.			
API Function(s) covered:				
TCSLIB_HANDLE TCSLibraryOpen(void);				
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,				
TCSScanResult *pResult);				
int TCSLibraryClose(TCSLIB_HANDLE hLib);				
<u>Test Objectives:</u> This test case verifies that the expected return value is returned when the interface is called to scan benign Java code formatted content data.				
<u>Test pre-conditions:</u>				
For validation plug-in only.				
Test Procedure:				
1. Call TCSLibraryOpen().				
	ed with benign Java code formatted data, TCS_SA_SCANONLY as VA as the data type identifier. Set pfCallback to NULL.			
3. Verify that the return value of TCSScanE	Data() is 0.			
4. Verify that the number of the detected ma	lware is 0.			
5. Call pfFreeResult() to release the re	source returned by TCS library.			
6. Call TCSLibraryClose() with the TC	CS library handle returned by the <code>TCSLibraryOpen()</code> .			
Test PASS Condition:				
Step 3 should pass verification.				
Step 4 should pass verification.				
Test Clean-up procedure:				

## 5.28 Test Case TC\_SEC\_CS\_TCSScanData\_0022

TC_SEC_CS_TCSScanData_0022	Call TCS interface to scan benign Java code formatted content data.	
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void)	);	
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
TCSScanResult *pRe	esult);	
int TCSLibraryClose(TCSLIB_HANDLE	hLib);	
<u>Test Objectives:</u>		
This test case verifies that the expected return valu formatted content data.	ue is returned when the interface is called to scan benign Java code	
Test pre-conditions:		

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with benign Java code formatted data, TCS\_SA\_SCANONLY as the scan action ID, TCS\_DTYPE\_JAVA as the data type identifier and where pfCallback is not NULL.
- 3. Verify that pfCallback is not called.
- 4. Verify that the return value of TCSScanData() is 0.
- 5. Verify that the number of the detected malware is 0.
- 6. Call pfFreeResult () to release the resource returned by TCS library.
- 7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

# 5.29 Test Case TC\_SEC\_CS\_TCSScanData\_0023

10_3	EC_CS_TCSScanData_0023 Call TCS interface to scan infected Java code formatted content data.
API Fu	nction(s) covered:
TCSLI	B_HANDLE TCSLibraryOpen(void);
int T	CSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,
	TCSScanResult *pResult);
int T	CSLibraryClose(TCSLIB_HANDLE hLib);
This tes	<b>Djectives:</b> t case verifies that the expected value is returned when the interface is called to scan infected Java code ed content data.
<u>Test pr</u>	e-conditions:
For vali	dation plug-in only.
<u>Test Pr</u>	ocedure:
	ocedure: Call TCSLibraryOpen().
1.	
1.	Call TCSLibraryOpen(). Call TCSScanData() with a buffer filled with infected Java code formatted data, TCS_SA_SCANONLY as
1. 2.	Call TCSLibraryOpen(). Call TCSScanData() with a buffer filled with infected Java code formatted data, TCS_SA_SCANONLY as the scan action ID, and TCS_DTYPE_JAVA as the data type identifier. Set pfCallback to NULL.
1. 2. 3.	Call TCSLibraryOpen(). Call TCSScanData() with a buffer filled with infected Java code formatted data, TCS_SA_SCANONLY as the scan action ID, and TCS_DTYPE_JAVA as the data type identifier. Set pfCallback to NULL. Verify that the return value of TCSScanData() is 0. Verify that the number of the detected malware is as expected, the malware name or variant name is as

Step 4 should pass verification.

### Test Clean-up procedure:

# 5.30 Test Case TC\_SEC\_CS\_TCSScanData\_0024

TC_SEC_CS_TCSScanData_0024	Call TCS interface to scan infected Java code formatted content data.		
API Function(s) covered:			
TCSLIB_HANDLE TCSLibraryOpen(void)	);		
int TCSScanData(TCSLIB_HANDLE hLi)	o, TCSScanParam *pParam,		
TCSScanResult *pRe	esult);		
int TCSLibraryClose(TCSLIB_HANDLE	hLib);		
Test Objectives:			
This test case verifies that the expected value is returned when the interface is called to scan infected Java code formatted content data.			
Test pre-conditions:			
For validation plug-in only.			
Test Procedure:			
1. Call TCSLibraryOpen().			
2. Call TCSScanData() with a buffer filled with infected Java code formatted data, TCS_SA_SCANONLY as the scan action ID, TCS_DTYPE_JAVA as the data type identifier and where pfCallback is not NULL.			
3. Verify that pfCallback is called, the malware name or variant name is as expected and the severity/behaviour is as expected.			
4. Verify that the return value of TCSScan	Data() is 0.		
5. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.			
6. Call pfFreeResult() to release the r	6. Call pfFreeResult() to release the resource returned by TCS library.		
7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().			
Test PASS Condition:			
Step 3 should pass verification.			
Step 4 should pass verification.			
Step 5 should pass verification.			
Test Clean-up procedure:			
No specific cleanup required.			

## 5.31 Test Case TC\_SEC\_CS\_TCSScanData\_0025

TC_	_SEC_	_CS_	TCSScanData_	_0025
-----	-------	------	--------------	-------

Call TCS interface to scan benign JavaScript code formatted content data.

#### **API Function(s) covered:**

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanData(TCSLIB\_HANDLE hLib, TCSScanParam \*pParam,

TCSScanResult \*pResult);

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

#### **Test Objectives:**

This test case verifies that the expected return value is returned when the interface is called to scan benign JavaScript code formatted content data.

### Test pre-conditions:

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with benign JavaScript code formatted data, TCS\_SA\_SCANONLY as the scan action ID, and TCS\_DTYPE\_JAVAS as the data type identifier. Set pfCallback to NULL.
- 3. Verify that the return value of TCSScanData() is 0.
- 4. Verify that the number of the detected malware is 0.
- 5. Call pfFreeResult () to release the resource returned by TCS library.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

## 5.32Test Case TC\_SEC\_CS\_TCSScanData\_0026

TC_SEC_CS_TCSScanData_0026	Call TCS interface to scan benign JavaScript code formatted content data.
API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryOpen(void	);
int TCSScanData(TCSLIB_HANDLE hLi	b, TCSScanParam *pParam,
TCSScanResult *pR	esult);
int TCSLibraryClose(TCSLIB_HANDLE	hLib);
Test Objectives:	
This tast asso varifies that the expected return val	us is returned when the interface is called to seen benign IsveScript

This test case verifies that the expected return value is returned when the interface is called to scan benign JavaScript code formatted content data.

#### **Test pre-conditions:**

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with benign Java code formatted data, TCS\_SA\_SCANONLY as the scan action ID, TCS\_DTYPE\_JAVAS as the data type identifier and where pfCallback is not NULL.
- 3. Verify that pfCallback is not called.
- 4. Verify that the return value of TCSScanData() is 0.
- 5. Verify that the number of the detected malware is 0.
- 6. Call pfFreeResult() to release the resource returned by TCS library.
- 7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

## 5.33 Test Case TC\_SEC\_CS\_TCSScanData\_0027

	code formatted content data.
API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryOpen(void)	;
int TCSScanData(TCSLIB_HANDLE hLib,	, TCSScanParam *pParam,
TCSScanResult *pRes	sult);
int TCSLibraryClose(TCSLIB_HANDLE }	nLib);
Test Objectives:	
This test case verifies that the expected value is retu formatted content data.	rrned when the interface is called to scan infected JavaScript code
Test pre-conditions:	
For validation plug-in only.	

Call TCS interface to scan infected JavaScript

#### Test Procedure:

1. Call TCSLibraryOpen().

TC\_SEC\_CS\_TCSScanData\_0027

- 2. Call TCSScanData() with a buffer filled with infected JavaScript code formatted data, TCS\_SA\_SCANONLY as the scan action ID, and TCS\_DTYPE\_JAVAS as the data type identifier. Set pfCallback to NULL.
- 3. Verify that the return value of TCSScanData() is 0.
- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call pfFreeResult () to release the resource returned by TCS library.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

## 5.34Test Case TC\_SEC\_CS\_TCSScanData\_0028

TC_SEC_CS_TCSScanData_0028	Call TCS interface to scan infected JavaScript code formatted content data.	
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void)	;	
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
TCSScanResult *pResult);		
int TCSLibraryClose(TCSLIB_HANDLE hLib);		
<u>Test Objectives:</u>		
This test case verifies that the expected value is ret formatted content data.	turned when the interface is called to scan infected JavaScript code	
Test pre-conditions:		
For validation plug-in only.		
- · · ·		

#### Test Procedure:

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with infected JavaScript code formatted data, TCS\_SA\_SCANONLY as the scan action ID, TCS\_DTYPE\_JAVAS as the data type identifier and where pfCallback is not NULL.
- 3. Verify that pfCallback is called, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 4. Verify that the return value of TCSScanData() is 0.
- 5. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 6. Call pfFreeResult() to release the resource returned by TCS library.
- 7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

# 5.35Test Case TC\_SEC\_CS\_TCSScanData\_0029

TC_SEC_CS_TCS	ScanData_0029	Call TCS interface to scan benign text content data.
API Function(s) cove	red:	
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>		
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
TCSScanResult *pResult);		
int TCSLibraryC	lose(TCSLIB_HANDLE	hLib);
This test case verifies Test pre-conditions:	hat the expected return valu	e is returned when interface is called to scan benign text content data.
For validation plug-in	only.	
Test Procedure:	·	
1. Call TCSLib	raryOpen().	
<ol> <li>Call TCSScanData() with a buffer filled with benign text data, TCS_SA_SCANONLY as the scan action ID, and TCS_DTYPE_TEXT as the data type identifier. Set pfCallback to NULL.</li> </ol>		
3. Verify that the return value of TCSScanData() is 0.		
4. Verify that the number of the detected malware is 0.		
5. Call pfFreeResult() to release the resource returned by TCS library.		
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().		
Test PASS Condition	<u>:</u>	
Step 3 should pass ver	fication.	
Step 4 should pass verification.		

Test Clean-up procedure:

# 5.36 Test Case TC\_SEC\_CS\_TCSScanData\_0030

TC_SEC_CS_TCSScanData_0030 Call TCS interface to scan benign text content data.		
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void);		
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
TCSScanResult *pResult);		
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>		
Test Objectives:		
This test case verifies that the expected return value is returned when interface is called to scan benign text content data.		
Test pre-conditions:		
For validation plug-in only.		
Test Procedure:		
1. Call TCSLibraryOpen().		
2. Call TCSScanData() with a buffer filled with benign text data, TCS_SA_SCANONLY as the scan action ID, TCS_DTYPE_TEXT as the data type identifier and where pfCallback is not NULL.		
3. Verify that pfCallback is not called.		
4. Verify that the return value of TCSScanData() is 0.		
5. Verify that the number of the detected malware is 0.		
6. Call pfFreeResult() to release the resource returned by TCS library.		
7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().		
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Step 5 should pass verification.		
Test Clean-up procedure:		

# 5.37Test Case TC\_SEC\_CS\_TCSScanData\_0031

TC_SEC_CS_TCSScanData_0031       Call TCS interface to scan infected text content data.		
API Function(s) covered:		
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>		
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
TCSScanResult *pResult);		
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>		
Test Objectives:		
This test case verifies that the expected return value is returned when the interface is called to scan infected text content data.		
Test pre-conditions:		
For validation plug-in only.		
Test Procedure:		
1. Call TCSLibraryOpen().		
<ol> <li>Call TCSScanData() with a buffer filled with infected text data, TCS_SA_SCANONLY as the scan action ID, and TCS_DTYPE_TEXT as the data type identifier. Set pfCallback to NULL.</li> </ol>		
3. Verify that the return value of TCSScanData() is 0.		
<ol> <li>Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.</li> </ol>		
5. Call pfFreeResult() to release the resource returned by TCS library.		
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().		
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Test Clean-up procedure:		
No specific cleanup required.		

# 5.38Test Case TC\_SEC\_CS\_TCSScanData\_0032

TC_SE	_SEC_CS_TCSScanData_0032 Call TCS interface to s data.	can infected text content	
API Fu	Function(s) covered:		
TCSLI	LIB_HANDLE TCSLibraryOpen(void);		
int T	int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
	<pre>TCSScanResult *pResult);</pre>		
int T	TCSLibraryClose(TCSLIB_HANDLE hLib);		
<u>Test Ob</u>	t Objectives:		
This test data.	s test case verifies that the expected return value is returned when the interface .	is called to scan infected text content	
Test pro	t pre-conditions:		
For vali	validation plug-in only.		
Test Pro	t Procedure:		
1.	1. Call TCSLibraryOpen().		
2.	<ol> <li>Call TCSScanData() with a buffer filled with infected text data, TCS_ ID, TCS_DTYPE_TEXT as the data type identifier and where pfCallba</li> </ol>		
3.	3. Verify that pfCallback is called. The malware name or variant name i severity/behaviour is as expected.	s as expected and the	
4.	4. Verify that the return value of TCSScanData() is 0.		
5.	5. Verify that the number of the detected malware is as expected, the malwa expected and the severity/behaviour is as expected.	re name or variant name is as	
6.	6. Call pfFreeResult() to release the resource returned by TCS library.		
7.	7. Call TCSLibraryClose() with the TCS library handle returned by th	eTCSLibraryOpen().	
<u>Test PA</u>	t PASS Condition:		
Step 3 s	3 should pass verification.		
Step 4 s	4 should pass verification.		
Step 5 s	5 should pass verification.		
Test Clo	t Clean-up procedure:		
No spec	specific cleanup required.		

## 5.39 Test Case TC\_SEC\_CS\_TCSScanData\_0033

TC_SEC_CS_TCSScanData_0033	Call TCS interface to scan content data infected by multiple malware.
API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryOpen(void)	;
int TCSScanData(TCSLIB_HANDLE hLib	), TCSScanParam *pParam,
TCSScanResult *pRe	esult);
int TCSLibraryClose(TCSLIB_HANDLE	hLib);
Test Objectives:	

### Test Objectives:

This test case verifies that the expected return value is returned when the interface is called to scan content data infected by multiple malware.

#### **Test pre-conditions:**

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with data infected by multiple malwares, TCS\_SA\_SCANONLY as the scan action ID, and TCS\_DTYPE\_UNKNOWN as the data type identifier. Set pfCallback to NULL.
- 3. Verify that the return value of TCSScanData() is 0.
- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call pfFreeResult () to release the resource returned by TCS library.

6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

## 5.40 Test Case TC\_SEC\_CS\_TCSScanData\_0034

TC_SEC_CS_TCSScanData_0034	Call TCS interface to scan content data infected by multiple malware.
API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryOpen(void)	;

int TCSScanData(TCSLIB\_HANDLE hLib, TCSScanParam \*pParam,

TCSScanResult \*pResult);

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

#### **Test Objectives:**

This test case verifies that the expected return value is returned when the interface is called to scan content data infected by multiple malware.

#### Test pre-conditions:

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with data infected by multiple malwares, TCS\_SA\_SCANONLY as the scan action ID, TCS\_DTYPE\_UNKNOWN as the data type identifier and where pfCallback is not NULL.
- 3. Verify that pfCallback is called, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 4. Verify that the return value of TCSScanData() is 0.
- 5. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 6. Call pfFreeResult () to release the resource returned by TCS library.
- 7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### Test PASS Condition:

Step 3 should pass verification.

Step 4 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

# 5.41 Test Case TC\_SEC\_CS\_TCSScanData\_0035

TC_SEC_CS_TCSScanData_0035 Call TCS interface to repair infected content data.	
API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryOpen(void);	
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,	
TCSScanResult *pResult);	
int TCSLibraryClose(TCSLIB_HANDLE hLib);	
Test Objectives:	
This test case verifies the expected return value is returned when TCS interface is called to repair infected content data	
Test pre-conditions:	
For validation plug-in only.	
Repairing functionality is required in validation plug-in.	
<u>Test Procedure:</u>	
1. Call TCSLibraryOpen().	
<ol> <li>Call TCSScanData() with a buffer filled with infected data, TCS_SA_SCANREPAIR as the scan action ID and TCS_DTYPE_UNKNOWN as the data type identifier.</li> </ol>	
3. Verify that the return value of TCSScanData() is 0.	
4. Verify that the content data is repaired by comparing with prepared clean data.	
5. Call pfFreeResult() to release the resource returned by TCS library.	
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:	
Step 3 should pass verification.	
Step 4 should pass verification.	
Test Clean-up procedure:	

# 5.42Test Case TC\_SEC\_CS\_TCSScanData\_0036

TC_SI	EC_CS_TCSScanData_0036	Call TCS interface to repair infected HTML formatted content data.
API Function(s) covered:		
TCSLI	<pre>ICSLIB_HANDLE TCSLibraryOpen(void);</pre>	
int T	int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,	
	TCSScanResult *pResult);	
int T	CSLibraryClose(TCSLIB_HANDLE	hLib);
Test Ol	bjectives:	
	st case verifies that the expected return val ed content data.	ue is returned when the TCS interface is called to repair infected HTML
Test pre-conditions:		
Test pr	<u>re-conditions:</u>	
	re-conditions: idation plug-in only.	
For vali		lug-in.
For vali Repairii	idation plug-in only.	lug-in.
For vali Repairin <u>Test Pr</u>	idation plug-in only. ng functionality is required in validation pl	lug-in.
For vali Repairin <u>Test Pr</u>	idation plug-in only. ng functionality is required in validation pl rocedure: Call TCSLibraryOpen().	led with infected HTML formatted data, TCS_SA_SCANREPAIR as the
For vali Repairin <u>Test Pr</u> 1.	idation plug-in only. ng functionality is required in validation pl cocedure: Call TCSLibraryOpen(). Call TCSScanData() with a buffer fil	led with infected HTML formatted data, TCS_SA_SCANREPAIR as the as the data type identifier.
For vali Repairin <u>Test Pr</u> 1. 2.	idation plug-in only. ng functionality is required in validation pl rocedure: Call TCSLibraryOpen(). Call TCSScanData() with a buffer fil scan action ID and TCS_DTYPE_HTML	led with infected HTML formatted data, TCS_SA_SCANREPAIR as the as the data type identifier.
For vali <u>Repairin</u> <u>Test Pr</u> 1. 2. 3.	idation plug-in only. ng functionality is required in validation pl cocedure: Call TCSLibraryOpen(). Call TCSScanData() with a buffer fill scan action ID and TCS_DTYPE_HTML Verify that the return value of TCSScan	led with infected HTML formatted data, TCS_SA_SCANREPAIR as the as the data type identifier. Data() is 0. y comparing with prepared clean data.

Step 3 should pass verification.

Step 4 should pass verification.

### Test Clean-up procedure:

# 5.43 Test Case TC\_SEC\_CS\_TCSScanData\_0037

TC_SEC_CS_TCSScanData_0037	Call TCS interface to repair infected URL formatted content data.	
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void	);	
int TCSScanData(TCSLIB_HANDLE hLi)	o, TCSScanParam *pParam,	
TCSScanResult *pResult);		
int TCSLibraryClose(TCSLIB_HANDLE	hLib);	
Test Objectives: This test case verifies that the expected return value formatted content data.	ie is returned when the interface is called to repair infected URL	
Test pre-conditions:		
For validation plug-in only.		
Repairing functionality is required in validation pl	ug-in.	
Test Procedure:		
1. Call TCSLibraryOpen().		
2. Call TCSScanData() with a buffer filled with infected URL formatted data, TCS_SA_SCANREPAIR as the scan action ID and TCS_DTYPE_URL as the data type identifier.		
3. Verify that the return value of TCSScan	Data() is 0.	
4. Verify that the content data is repaired by	comparing with prepared clean data.	
5. Call pfFreeResult() to release the r	esource returned by TCS library.	
6. Call TCSLibraryClose() with the T	CS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Test Clean-up procedure:		

# 5.44 Test Case TC\_SEC\_CS\_TCSScanData\_0038

TC_SEC_CS_TCSScanData_0038	Call TCS interface to repair infected Email formatted content data.		
API Function(s) covered:			
TCSLIB_HANDLE TCSLibraryOpen(void	1);		
int TCSScanData(TCSLIB_HANDLE hLi	b, TCSScanParam *pParam,		
TCSScanResult *pR	TCSScanResult *pResult);		
int TCSLibraryClose(TCSLIB_HANDLE	hLib);		
Test Objectives:			
This test case verifies that the expected return val formatted content data.	ue is returned when the interface is called to repair infected Email		
Test pre-conditions:			
For validation plug-in only.			
Repairing functionality is required in validation p	Repairing functionality is required in validation plug-in.		
Test Procedure:			
1. Call TCSLibraryOpen().	1. Call TCSLibraryOpen().		
2. Call TCSScanData() with a buffer filled with infected Email formatted data, TCS_SA_SCANREPAIR as the scan action ID and TCS_DTYPE_EMAIL as the data type identifier.			
3. Verify that the return value of TCSScanData() is 0.			
4. Verify that the content data is repaired by comparing with prepared clean data.			
5. Call pfFreeResult() to release the resource returned by TCS library.			
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().			
Test PASS Condition:			
Step 3 should pass verification.			
Step 4 should pass verification.			
Test Clean-up procedure:			
No specific cleanup required.			

# 5.45 Test Case TC\_SEC\_CS\_TCSScanData\_0039

		TCS interface to repair infected phone ber formatted content data.
<u>API Fı</u>	Junction(s) covered:	
TCSLI	<pre>IB_HANDLE TCSLibraryOpen(void);</pre>	
int T	TCSScanData(TCSLIB_HANDLE hLib, TCS	SScanParam *pParam,
	TCSScanResult *pResult)	);
int I	TCSLibraryClose(TCSLIB_HANDLE hLib)	);
Test O	<u>Dbjectives:</u>	
	est case verifies that the expected value is returned v tted content data.	when the interface is called to repair infected phone number
Test pi	pre-conditions:	
For val	lidation plug-in only.	
Repairi	ring functionality is required in validation plug-in.	
Test Pi	Procedure:	
1.	Call TCSLibraryOpen().	
2.		infected phone number formatted data, nd TCS_DTYPE_PHONE as the data type identifier.
3.	. Verify that the return value of TCSScanData (	) is 0.
4.	. Verify that the content data is repaired by compa	ring with prepared clean data.
5.	. Call pfFreeResult() to release the resource	returned by TCS library.
6.	Call TCSLibraryClose() with the TCS libration	ary handle returned by the TCSLibraryOpen().
<b>T</b> ( <b>D</b>	PASS Condition:	
Test P		
	should pass verification.	

# 5.46 Test Case TC\_SEC\_CS\_TCSScanData\_0040

TC_SEC_CS_TCSScanData_0040	Call TCS interface to repair infected Java code formatted content data.	
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void)	;	
int TCSScanData(TCSLIB_HANDLE hLik	int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,	
TCSScanResult *pResult);		
int TCSLibraryClose(TCSLIB_HANDLE	hLib);	
Test Objectives:		
This test case verifies that the expected value is ret formatted content data.	turned when the interface is called to repair infected Java code	
Test pre-conditions:		
For validation plug-in only.		
Repairing functionality is required in validation pl	ug-in.	
Test Procedure:		
1. Call TCSLibraryOpen().		
2. Call TCSScanData() with a buffer fill as the scan action ID and TCS_DTYPE_C	ed with infected Java code formatted data, TCS_SA_SCANREPAIR JAVA as the data type identifier.	
3. Verify that the return value of TCSScan	Data() is 0.	
4. Verify that the content data is repaired by	comparing with prepared clean data.	
5. Call pfFreeResult() to release the re	esource returned by TCS library.	
6. Call TCSLibraryClose() with the T	CS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Test Clean-up procedure:		
No specific cleanup required.		

# 5.47Test Case TC\_SEC\_CS\_TCSScanData\_0041

TC_SEC_CS_TCSScanData_0041	Call TCS interface to repair infected text content data.
API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryOpen(void	);
int TCSScanData(TCSLIB_HANDLE hLi)	b, TCSScanParam *pParam,
TCSScanResult *pR	esult);
int TCSLibraryClose(TCSLIB_HANDLE	hLib);
<u>Test Objectives:</u>	
This test case verifies that the expected return valu content data.	ue is returned when the interface is called to repair infected text
Test pre-conditions:	
For validation plug-in only.	
Repairing functionality is required in validation pl	lug-in.
Test Procedure:	
1. Call TCSLibraryOpen().	
2. Call TCSScanData() with a buffer fill action ID and TCS_DTYPE_TEXT as the	led with infected text data, TCS_SA_SCANREPAIR as the scan e data type identifier.
3. Verify that the return value of TCSScan	Data() is 0.
4. Verify that the content data is repaired by	y comparing with prepared clean data.
5. Call pfFreeResult() to release the r	resource returned by TCS library.
6. Call TCSLibraryClose() with the T	CCS library handle returned by the TCSLibraryOpen().
Test PASS Condition:	
Step 3 should pass verification.	
Step 4 should pass verification.	
Test Clean-up procedure:	
No specific cleanum required	

# 5.48Test Case TC\_SEC\_CS\_TCSScanData\_0042

TC_SEC_CS_TCSScanData_0042	Call TCS interface to repair content data infected by multiple malware.	
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void)	;	
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
TCSScanResult *pResult);		
int TCSLibraryClose(TCSLIB_HANDLE hLib);		
Test Objectives:		
This test case verifies that the expected return valu infected by multiple malware.	e is returned when the interface is called to repair content data	
Test pre-conditions:		
For validation plug-in only.		
Repairing functionality is required in validation pl	ug-in.	
Test Procedure:		
1. Call TCSLibraryOpen().		
<ol> <li>Call TCSScanData() with a buffer fill scan action ID and TCS_DTYPE_UNKNC</li> </ol>	ed with test multiple malware data, TCS_SA_SCANREPAIR as the DWN as the data type identifier.	
3. Verify that the return value of TCSScan	Data() is 0.	
4. Verify that the content data is repaired by	comparing with prepared clean data.	
5. Call pfFreeResult() to release the re	esource returned by TCS library.	
6. Call TCSLibraryClose() with the T	CS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Test Clean-up procedure:		
No specific cleanup required.		

## 5.49 Test Case TC\_SEC\_CS\_TCSScanData\_0043

### TC\_SEC\_CS\_TCSScanData\_0043 Return -1 in pfCallback.

#### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanData(TCSLIB\_HANDLE hLib, TCSScanParam \*pParam,

TCSScanResult \*pResult);

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

#### **Test Objectives:**

This test case verifies that the expected return value is returned when pfCallback returns -1 to the TCS library.

#### **Test pre-conditions:**

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with test malware data, TCS\_SA\_SCANONLY as the scan action ID, TCS\_DTYPE\_UNKNOWN as the data type identifier and where pfCallback is not NULL.
- 3. Return -1 in pfCallback when the detection notify occurrs.
- 4. Verify that the return value of TCSScanData() is -1.
- 5. Call TCSGetLastError().
- 6. Verify that the error code returned from TCSGetLastError() is TCS\_ERROR\_CANCELLED.
- 7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 4 should pass verification.

Step 6 should pass verification.

#### Test Clean-up procedure:

## 5.50 Test Case TC\_SEC\_CS\_TCSScanData\_0044

#### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanData(TCSLIB\_HANDLE hLib, TCSScanParam \*pParam,

TCSScanResult \*pResult);

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

#### **Test Objectives:**

This test case verifies that the expected return value is returned when calling the TCS interface to repair infected content data where the repair functionality is not implemented in the TCS library.

#### **Test pre-conditions:**

For validation plug-in only.

Repairing functionality is required to be not implemented in validation plug-in for this test case.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanData() with a buffer filled with infected data, TCS\_SA\_SCANREPAIR as the scan action ID, TCS\_DTYPE\_UNKNOWN as the data type identifier.
- 3. Verify that the return value of TCSScanData() is -1.
- 4. Call TCSGetLastError() to get error code.
- 5. Verify that the error code returned by TCSGetLastError() is TCS\_ERROR\_NOT\_IMPLEMENTED.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

## 5.51 Test Case TC\_SEC\_CS\_TCSScanData\_0045

TC_SEC_CS_TCSScanData_0045	Call TCS data scan interface with invalid library instance handle.	
API Function(s) covered:		
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,		
TCSScanResult *pRe	esult);	
<u>Test Objectives:</u>		
This test case verifies that -1 is returned when an invalid scanner instance handle is passed to data scan interface.		
Test pre-conditions:		
For validation plug-in only.		
Test Procedure:		
1. Call TCSScanData() with an invalid library instance handle INVALID_TCSLIB_HANDLE.		
2. Verify that the return value of TCSScanData() is -1.		
Test PASS Condition:		
Step 2 should pass verification.		
Test Clean-up procedure:		
No specific cleanup required.		

## 5.52Test Case TC\_SEC\_CS\_TCSScanData\_0046

TC_SEC_CS_TCSScanData_0046	Concurrency TCS data scan test.
----------------------------	---------------------------------

### API Function(s) covered:

int TCSScanData(TCSSCAN\_HANDLE hScan, TCSScanParam \*pParam,

TCSScanResult \*pResult);

#### Test Objectives:

This test case verifies that TCSScanData() can be correctly handled by multiple scanner instance handles in multiple threads.

#### **Test pre-conditions:**

For validation plug-in only.

### Test Procedure:

- 1. Create multiple threads to execute from 2 to 10.
- 2. Call TCSLibraryOpen().

TC_S	EC_CS_TCSScanData_0046 Concurrency TCS data scan test.
3.	Call TCSScanData() with an infected buffer with test malware data, TCS_SA_SCANONLY as the scan action ID, TCS_DTYPE_UNKNOWN as the data type identifier.
4.	Verify that the return value of TCSScanData() is 0.
5.	Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
6.	Call pfFreeResult() to release the resource returned by TCS library.
7.	Call <code>TCSLibraryClose()</code> with the TCS library handle returned by the <code>TCSLibraryOpen()</code> .
8.	Repeat 2 ~ 9 with different parameter for TCSScanData(), other test samples: (html, url, email, phone number, Java code, text) and respective data type identifier.
Test P	ASS Condition:
Step 4	should pass verification.
Step 5	should pass verification.
	lean-up procedure:
No spe	cific cleanup required.

# 5.53Test Case TC\_SEC\_CS\_TCSScanData\_0047

TC_S	EC_CS_TCSScanData_0047 Concurrency TCS data clean test.	
<u>API Fu</u>	unction(s) covered:	
int T	int TCSScanData(TCSSCAN_HANDLE hScan, TCSScanParam *pParam,	
	TCSScanResult *pResult);	
Test O	bjectives:	
This tes threads	st case verifies that TCSScanData() can be correctly handled by multiple scanner instance handles in multiple	
Test pre-conditions:		
For validation plug-in only.		
Repairing functionality is required in validation plug-in.		
<u>Test Pr</u>	ocedure:	
1.	Create multiple threads to execute from 2 to 10.	
2.	Call TCSLibraryOpen().	
3.	Call TCSScanData() with an infected buffer with test malware data, TCS_SA_SCANREPAIR as the scan action ID, TCS_DTYPE_UNKNOWN as the data type identifier.	

TC_S	EC_CS_TCSScanData_0047	Concurrency TCS data clean test.
5.	Verify that the infected data is repaired by a supposed to be infected.	comparing with the respective clean buffer data if the input data is
6.	$Call  {\tt pfFreeResult}$ ( ) to release the reso	ource returned by TCS library.
7.	Call <code>TCSLibraryClose()</code> with the TCS	S library handle returned by the <code>TCSLibraryOpen()</code> .
8.	Repeat 2 ~ 9 with different parameter for T number, java code, text) and respective data	CSScanData(), other test samples: (html, url, email, phone a type identifier.
Test PA	ASS Condition:	
Step 4	should pass verification.	
Step 5 s	should pass verification.	
Test Clean-up procedure:		

No specific cleanup required.

# 5.54 Test Case TC\_SEC\_CS\_TCSScanData\_0048

TC_SEC_CS_TCSScanData_0048 Compress flag TCS data clean test.	
API Function(s) covered:	
int TCSScanData(TCSSCAN_HANDLE hScan, TCSScanParam *pParam,	
TCSScanResult *pResult);	
Test Objectives:	
This test case verifies that TCSScanData() can correctly scan clean data with compress flag enabled.	
Test pre-conditions:	
For validation plug-in only.	
Repairing functionality is required in validation plug-in.	
Test Procedure:	
1. Call TCSLibraryOpen().	
2. Call TCSScanData() with a buffer filled by clean data, TCS_SA_SCANONLY as the scan action ID, TCS_DTYPE_UNKNOWN as the data type identifier, set compress flag to 1.	
3. Verify that the return value of TCSScanData() is 0.	
4. Verify that the no malware found.	
5. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:	
Step 3 should pass verification.	
Step 4 should pass verification.	

TC\_SEC\_CS\_TCSScanData\_0048 Compress f

Compress flag TCS data clean test.

Test Clean-up procedure:

No specific cleanup required.

## 5.55Test Case TC\_SEC\_CS\_TCSScanData\_0049

TC_SEC_CS_TCSScanData_0049 Compress flag TCS data clean test.
API Function(s) covered:
int TCSScanData(TCSSCAN_HANDLE hScan, TCSScanParam *pParam,
TCSScanResult *pResult);
Test Objectives:
This test case verifies that TCSScanData() can correctly scan clean data with compress flag disabled.
Test pre-conditions:
For validation plug-in only.
Repairing functionality is required in validation plug-in.
<u>Test Procedure:</u>
1. Call TCSLibraryOpen().
2. Call TCSScanData() with a buffer filled by clean data, TCS_SA_SCANONLY as the scan action ID, TCS_DTYPE_UNKNOWN as the data type identifier, set compress flag to 0.
3. Verify that the return value of TCSScanData() is 0.
4. Verify that the no malware found.
5. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().
Test PASS Condition:
Step 3 should pass verification.
Step 4 should pass verification.
Test Clean-up procedure:
No specific cleanup required.

## 5.56 Test Case TC\_SEC\_CS\_TCSScanData\_0050

TC\_SEC\_CS\_TCSScanData\_0050

Compress flag TCS data test.

TC_SEC_CS_TCSScanData_0050 Compress flag TCS data test.		
API Function(s) covered:		
int TCSScanData(TCSSCAN_HANDLE hScan, TCSScanParam *pParam,		
TCSScanResult *pResult);		
Test Objectives:		
This test case verifies that TCSScanData() can correctly detect malware with compress flag	g enabled.	
Test pre-conditions:		
For validation plug-in only.		
Repairing functionality is required in validation plug-in.		
Test Procedure:		
1. Call TCSLibraryOpen().		
2. Call TCSScanData() with a buffer filled by test malware, TCS_SA_SCANONLY a TCS_DTYPE_UNKNOWN as the data type identifier, set compress flag to 1.	s the scan action ID,	
3. Verify that the return value of TCSScanData() is 0.		
4. Verify that the infected data is repaired by comparing with the respective clean buffer supposed to be infected.	data if the input data is	
5. Call pfFreeResult() to release the resource returned by TCS library.		
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibr	aryOpen().	
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Test Clean-up procedure:		
No specific cleanup required.		

# 5.57Test Case TC\_SEC\_CS\_TCSScanData\_0051

TC_SEC_CS_TCSScanData_0051	Compress flag TCS data test.
API Function(s) covered:	
int TCSScanData(TCSSCAN_HANDLE hSc	can, TCSScanParam *pParam,
TCSScanResult *pRe	esult);
<u>Test Objectives:</u>	
This test case verifies that TCSScanData() can	not correctly detect malware without compress flag enabled.
Test pre-conditions:	

TC_SEC_CS_TCSScanData_0051 Compress flag TCS data test.	
For validation plug-in only.	
Repairing functionality is required in validation plug-in.	
Test Procedure:	
1. Call TCSLibraryOpen().	
2. Call TCSScanData() with a buffer filled by test malware, TCS_SA_SCANONLY as the scan action ID, TCS_DTYPE_UNKNOWN as the data type identifier, set compress flag to 0.	
3. Verify that the return value of TCSScanData() is 0.	
4. Verify that no malware found.	
5. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:	
Step 3 should pass verification.	
Step 4 should pass verification.	
Test Clean-up procedure:	
No specific cleanup required.	

# 5.58Test Case TC\_SEC\_CS\_TCSScanData\_0052

TC_SEC_CSSTUB_TCSScanData_0052 Stub TCS function error return.
API Function(s) covered:
int TCSScanData(TCSLIB_HANDLE hLib, TCSScanParam *pParam,
TCSScanResult *pResult);
Test Objectives:
This test case verifies that the calling application can get proper error code from TCS stub functions.
Test pre-conditions:
Stub functions
Test Procedure:
1. Call TCSScanData() with INVALID_TCSLIB_HANDLE.
2. Verify it returns -1.
Test PASS Condition:
Step 2 should passed.
Test Clean-up procedure:
None.

## 5.59Test Case TC\_SEC\_CS\_TCSScanFile\_0001

TC_SEC_CS_TCSScanFile_0001	Call TCS interface to scan a benign file.
API Function(s) covered:	
AP1 Function(s) covered:	

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult );

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

### Test Objectives:

This test case tests the scan request interface and verifies that the TCS interface returns the expected return value in the case of a benign file.

### Test pre-conditions:

For validation plug-in only.

### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with a benign file path, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_UNKNOWN as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is 0.
- 5. Call pfFreeResult() to release the resource returned by TCS library.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

### Test PASS Condition:

Step 3 should pass verification.

Step 4 should pass verification.

### Test Clean-up procedure:

## 5.60 Test Case TC\_SEC\_CS\_TCSScanFile\_0002

# TC\_SEC\_CS\_TCSScanFile\_0002 Call TCS interface to scan an infected file.

### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult );

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

### Test Objectives:

This test case verifies that the expected return value is returned when the TCS interface is called to scan an infected file.

#### **Test pre-conditions:**

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with an infected file, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_UNKNOWN as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call pfFreeResult () to release the resource returned by TCS library.

6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

### Test Clean-up procedure:

## 5.61 Test Case TC\_SEC\_CS\_TCSScanFile\_0003

## TC\_SEC\_CS\_TCSScanFile\_0003 Call TCS interface to scan a benign HTML file.

#### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult );

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

### Test Objectives:

This test case verifies that the TCS interface returns the expected return value when it is called to scan a benign HTML file.

### Test pre-conditions:

For validation plug-in only.

### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with a benign HTML file path, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_HTML as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is 0.
- 5. Call pfFreeResult () to release the resource returned by TCS library.
- $6. \quad Call ~ {\tt TCSLibraryClose()} ~ with the ~ {\tt TCS library handle returned by the ~ {\tt TCSLibraryOpen()}.$

### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

## 5.62Test Case TC\_SEC\_CS\_TCSScanFile\_0004

# TC\_SEC\_CS\_TCSScanData\_0004 Call TCS interface to scan an infected HTML file.

#### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult );

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

#### Test Objectives:

This test case verifies that the expected return value is returned when the TCS interface is called to scan an infected HTML file.

### Test pre-conditions:

For validation plug-in only.

### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with an infected HTML file path, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_HTML as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call pfFreeResult () to release the resource returned by TCS library.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

### Test Clean-up procedure:

# 5.63Test Case TC\_SEC\_CS\_TCSScanFile\_0005

TC_SEC_CS_TCSScanFile_0005 Call TCS interface to scan a benign URL within a file.		
API Function(s) covered:		
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>		
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,		
int iDataType, int iAction, int iCompressFlag,		
TCSScanResult *pResult );		
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>		
Test Objectives:		
This test case verifies that the expected value is returned from the interface when it is called to scan a benign URL within a file.		
Test pre-conditions:		
For validation plug-in only.		
Test Procedure:		
1. Call TCSLibraryOpen().		
2. Call TCSScanFile() with a benign URL file path, TCS_SA_SCANONLY as the scan action ID and TCS_DTYPE_URL as the data type identifier.		
3. Verify that the return value of TCSScanFile() is 0.		
4. Verify that the number of the detected malware is 0.		
5. Call pfFreeResult() to release the resource returned by TCS library.		
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().		
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Test Clean-up procedure:		

# 5.64Test Case TC\_SEC\_CS\_TCSScanFile\_0006

TC_SEC_CS_TCSScanFile_0006 Call TCS interface to scan an infected URL within a file.
API Function(s) covered:
TCSLIB_HANDLE TCSLibraryOpen(void);
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,
int iDataType, int iAction, int iCompressFlag,
TCSScanResult *pResult );
int TCSLibraryClose(TCSLIB_HANDLE hLib);
This test case verifies that the expected return value is returned when the interface is called to scan an infected URL within a file.           Test pre-conditions:           For validation plug-in only.
Test Procedure:
1. Call TCSLibraryOpen().
2. Call TCSScanFile() with an infected URL file path, TCS_SA_SCANONLY as the scan action ID and TCS_DTYPE_URL as the data type identifier.
3. Verify that the return value of TCSScanFile() is 0.
4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
5. Call pfFreeResult() to release the resource returned by TCS library.
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().
Test PASS Condition:
Step 3 should pass verification.
Step 4 should pass verification.
<u>Test Clean-up procedure:</u>

# 5.65Test Case TC\_SEC\_CS\_TCSScanFile\_0007

TC_SEC_CS_TCSScanFile_0007	Call TCS interface to scan a benign Email file.
API Function(s) covered:	
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>	
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,	
int iDataType, int iAction, int iCompressFlag,	
TCSScanResult *pResult );	
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>	
Test Objectives: This test case verifies that the expected return value is returned when the interface is called to scan a benign Email file.	
Test pre-conditions: For validation plug-in only.	
Test Procedure:	
1. Call TCSLibraryOpen().	
2. Call TCSScanFile() with a benign Email file path, TCS_SA_SCANONLY as the scan action ID and TCS_DTYPE_EMAIL as the data type identifier.	
3. Verify that the return value of TCSScanFile() is 0.	
4. Verify that the number of the detected malware is 0.	
5. Call pfFreeResult() to release the resource returned by TCS library.	
6. Call TCSLibraryClose() with the TC	CS library handle returned by the TCSLibraryOpen().
Test PASS Condition:	
Step 3 should pass verification.	
Step 4 should pass verification.	
Test Clean-up procedure:	

## 5.66Test Case TC\_SEC\_CS\_TCSScanFile\_0008

## TC\_SEC\_CS\_TCSScanFile\_0008 Call TCS interface to scan an infected Email file.

#### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult );

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

## Test Objectives:

This test case verifies that the expected return value is returned when the interface is called to scan an infected Email file.

## Test pre-conditions:

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with an infected Email file path, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_EMAIL as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call pfFreeResult () to release the resource returned by TCS library.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

## Test Clean-up procedure:

# 5.67Test Case TC\_SEC\_CS\_TCSScanFile\_0009

TC_SEC_CS_TCSScanFile_0009	Call TCS interface to scan a benign phone number within a file.
API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryOpen(void)	;
int TCSScanFile(TCSLIB_HANDLE hLib	o, char const *pszFileName,
int iDataType, int	: iAction, int iCompressFlag,
TCSScanResult *pRe	esult );
int TCSLibraryClose(TCSLIB_HANDLE	hLib);
Test Objectives:	
This test case verifies that the expected return valu number within a file.	e is returned when the interface is called to scan a benign phone
Test pre-conditions:	
For validation plug-in only.	
Test Procedure:	
1. Call TCSLibraryOpen().	
<ol> <li>Call TCSScanFile() with a benign ph and TCS_DTYPE_PHONE as the data typ</li> </ol>	one number file path, TCS_SA_SCANONLY as the scan action ID be identifier.
3. Verify that the return value of TCSScan	File() is 0.
4. Verify that the number of the detected ma	llware is 0.
5. Call pfFreeResult() to release the re-	esource returned by TCS library.
6. Call TCSLibraryClose() with the Te	CS library handle returned by the TCSLibraryOpen().
Test PASS Condition:	
Step 3 should pass verification.	
Step 4 should pass verification.	

## Test Clean-up procedure:

## 5.68 Test Case TC\_SEC\_CS\_TCSScanFile\_0010

number within a file.
API Function(s) covered:
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,
int iDataType, int iAction, int iCompressFlag,
TCSScanResult *pResult );
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>
Test Objectives:

Call TCS interface to scan an infected phone

## Test Objectives:

TC\_SEC\_CS\_TCSScanFile\_0010

This test case verifies that the expected value is returned when the interface is called to scan an infected phone number within a file.

### **Test pre-conditions:**

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with an infected phone number file path, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_PHONE as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call pfFreeResult () to release the resource returned by TCS library.

6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

### Test Clean-up procedure:

## 5.69Test Case TC\_SEC\_CS\_TCSScanFile\_0011

TC_SEC_CS_TCSScanFile_0011	Call TCS interface to scan a benign Java file.

### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult);

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

## Test Objectives:

This test case verifies that the expected return value is returned when the interface is called to scan a benign Java file.

### Test pre-conditions:

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with a benign Java file path, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_JAVA as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is 0.
- 5. Call pfFreeResult () to release the resource returned by TCS library.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

## Test Clean-up procedure:

## 5.70 Test Case TC\_SEC\_CS\_TCSScanFile\_0012

TC_SEC_CS_TCSScanFile_0012	Call TCS interface to scan an infected Java file.

### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult );

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

## **Test Objectives:**

This test case verifies that the expected value is returned when the interface is called to scan an infected Java file.

### **Test pre-conditions:**

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with an infected Java file path, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_JAVA as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call pfFreeResult () to release the resource returned by TCS library.

6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

## **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

## Test Clean-up procedure:

## 5.71 Test Case TC\_SEC\_CS\_TCSScanFile\_0013

TC_SEC_CS_TCSScanFile_0013	Call TCS interface to scan a benign text file.
API Function(s) covered:	

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult );

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

## **Test Objectives:**

This test case verifies that the expected return value is returned when interface is called to scan a benign text file.

## Test pre-conditions:

For validation plug-in only.

## **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with a benign text file path, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_TEXT as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is 0.
- 5. Call pfFreeResult() to release the resource returned by TCS library.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

## **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

## Test Clean-up procedure:

## 5.72Test Case TC\_SEC\_CS\_TCSScanFile\_0014

## TC\_SEC\_CS\_TCSScanFile\_0014Call TCS interface to scan an infected text file.

### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult );

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

## Test Objectives:

This test case verifies that the expected return value is returned when the interface is called to scan an infected text file.

## **Test pre-conditions:**

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with an infected text file path, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_TEXT as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call pfFreeResult () to release the resource returned by TCS library.

6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

## Test Clean-up procedure:

## 5.73 Test Case TC\_SEC\_CS\_TCSScanFile\_0015

	multiple malware.
API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryOpen(void)	;
int TCSScanFile(TCSLIB_HANDLE hLib	o, char const *pszFileName,
int iDataType, int	: iAction, int iCompressFlag,
TCSScanResult *pRe	esult );
int TCSLibraryClose(TCSLIB_HANDLE	hLib);
Test Objectives:	
This test case verifies that the expected return valu multiple malware.	e is returned when the interface is called to scan a file infected by

Call TCS interface to scan a file infected by

## **Test pre-conditions:**

For validation plug-in only.

#### **Test Procedure:**

1. Call TCSLibraryOpen().

TC SEC CS TCSScanFile 0015

- 2. Call TCSScanFile() with a file path of a file infected by multiple malware, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_UNKNOWN as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 5. Call pfFreeResult () to release the resource returned by TCS library.

6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

### Test Clean-up procedure:

# 5.74Test Case TC\_SEC\_CS\_TCSScanFile\_0016

TC_SEC_CS_TCSScanFile_0016 Call TCS interface to repair an infected file.	
API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryOpen(void);	
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,	
int iDataType, int iAction, int iCompressFlag,	
TCSScanResult *pResult );	
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>	
<u>Test Objectives:</u> This test case verifies that the expected return value is returned when the TCS interface is called to repair an infected file.	
Test pre-conditions:	
For validation plug-in only.	
Repairing functionality is required in validation plug-in.	
Test Procedure:	
1. Call TCSLibraryOpen().	
2. Call TCSScanFile() with an infected file path, TCS_SA_SCANREPAIR as the scan action ID and TCS_DTYPE_UNKNOWN as the data type identifier.	
3. Verify that the return value of TCSScanFile() is 0.	
4. Verify that the content file is repaired by comparing with prepared clean file.	
5. Call pfFreeResult() to release the resource returned by TCS library.	
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:	
Step 3 should pass verification.	
Step 4 should pass verification.	
Test Clean-up procedure:	
No specific cleanup required.	

# 5.75Test Case TC\_SEC\_CS\_TCSScanFile\_0017

TC\_SEC\_CS\_TCSScanFile\_0017

<u>API Fu</u>	nction(s) covered:	
TCSLI	TCSLIB_HANDLE TCSLibraryOpen(void);	
int T	CSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,	
	int iDataType, int iAction, int iCompressFlag,	
	<pre>TCSScanResult *pResult );</pre>	
int T	CSLibraryClose(TCSLIB_HANDLE hLib);	
Test Ob	jectives:	
This tes	t case verifies that the expected return value is returned when the TCS interface is called to repair an infected file.	
Test pr	e-conditions:	
For vali	dation plug-in only.	
Repairii	g functionality is required in validation plug-in.	
<u>Test Pr</u>	ocedure:	
1.	Call TCSLibraryOpen().	
2.	Call TCSScanFile() with an infected HTML file path, TCS_SA_SCANREPAIR as the scan action ID and TCS_DTYPE_HTML as the data type identifier.	
3.	Verify that the return value of TCSScanFile() is 0.	
4.	Verify that the content file is repaired by comparing with prepared clean file.	
5.	Call pfFreeResult () to release the resource returned by TCS library.	
6.	Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().	
Test PA	SS Condition:	
Step 3 s	hould pass verification.	
Step 4 s	hould pass verification.	
•	^ 	

Call TCS interface to repair an infected HTML file.

## Test Clean-up procedure:

# 5.76 Test Case TC\_SEC\_CS\_TCSScanFile\_0018

	Call TCS interface to repair an infected URL within file.	
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void	);	
int TCSScanFile(TCSLIB_HANDLE hLi	b, char const *pszFileName,	
int iDataType, in	t iAction, int iCompressFlag,	
TCSScanResult *pR	esult );	
int TCSLibraryClose(TCSLIB_HANDLE	hLib);	
<b>Test Objectives:</b> This test case verifies that the expected return value is returned when the interface is called to repair an infected URL within a file.		
Test pre-conditions:	Test pre-conditions:	
For validation plug-in only.		
Repairing functionality is required in validation plug-in.		
Test Procedure:		
1. Call TCSLibraryOpen().		
<ol> <li>Call TCSScanFile() with an infected TCS_DTYPE_URL as the data type iden</li> </ol>	URL file path, TCS_SA_SCANREPAIR as the scan action ID and iffier.	
3. Verify that the return value of TCSScan	File() is 0.	
4. Verify that the content file is repaired by	comparing with prepared clean file.	
5. Call pfFreeResult() to release the r	resource returned by TCS library.	
6. Call TCSLibraryClose() with the T	CS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Test Clean-up procedure:		
No specific cleanup required.		

## 5.77Test Case TC\_SEC\_CS\_TCSScanFile\_0019

API Function(s) covered:	
TCSLIB_HANDLE TCSLibraryOpen(void);	
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,	
int iDataType, int iAction, int iCompressFlag,	
TCSScanResult *pResult );	
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>	

Call TCS interface to repair an infected Email file.

## Test Objectives:

This test case verifies that the expected return value is returned when the interface is called to repair an infected Email file.

## **Test pre-conditions:**

For validation plug-in only.

Repairing functionality is required in validation plug-in.

## **Test Procedure:**

1. Call TCSLibraryOpen().

TC SEC CS TCSScanFile 0019

- 2. Call TCSScanFile() with an infected Email file path, TCS\_SA\_SCANREPAIR as the scan action ID and TCS\_DTYPE\_EMAIL as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the content file is repaired by comparing with prepared clean file.
- 5. Call pfFreeResult () to release the resource returned by TCS library.

6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

## 5.78 Test Case TC\_SEC\_CS\_TCSScanFile\_0020

number within a file.
API Function(s) covered:
TCSLIB_HANDLE TCSLibraryOpen(void);
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,
int iDataType, int iAction, int iCompressFlag,
TCSScanResult *pResult );
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>
Test Objectives:
This test case verifies that the expected value is returned when the interface is called to repair an infected phone number within a file.

Call TCS interface to repair an infected phone

## **Test pre-conditions:**

For validation plug-in only.

Repairing functionality is required in validation plug-in.

#### **Test Procedure:**

1. Call TCSLibraryOpen().

TC\_SEC\_CS\_TCSScanFile\_0020

- 2. Call TCSScanFile() with an infected phone number file path, TCS\_SA\_SCANREPAIR as the scan action ID and TCS\_DTYPE\_PHONE as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the content file is repaired by comparing with prepared clean file.
- 5. Call pfFreeResult () to release the resource returned by TCS library.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

# 5.79Test Case TC\_SEC\_CS\_TCSScanFile\_0021

TC_SEC_CS_TCSScanFile_0021 Call TCS interface to repair an infected Java file	
API Function(s) covered:	
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>	
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,	
int iDataType, int iAction, int iCompressFlag,	
TCSScanResult *pResult );	
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>	
Test Objectives:	
This test case verifies that the expected value is returned when the interface is called to repair an infected Java file.	
Test pre-conditions:	
For validation plug-in only.	
Repairing functionality is required in validation plug-in.	
Test Procedure:	
1. Call TCSLibraryOpen().	
2. Call TCSScanFile() with an infected Java file path, TCS_SA_SCANREPAIR as the scan action ID and TCS_DTYPE_JAVA as the data type identifier.	
3. Verify that the return value of TCSScanFile() is 0.	
4. Verify that the content file is repaired by comparing with prepared clean file.	
5. Call pfFreeResult() to release the resource returned by TCS library.	
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:	
Step 3 should pass verification.	
Step 4 should pass verification.	
Test Clean-up procedure:	
No specific cleanup required.	

# 5.80 Test Case TC\_SEC\_CS\_TCSScanFile\_0022

TC_SEC_CS_TCSScanFile_0022 Call TCS interface to repair an infected text file.
API Function(s) covered:
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,
int iDataType, int iAction, int iCompressFlag,
<pre>TCSScanResult *pResult );</pre>
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>
Test Objectives:
This test case verifies that the expected return value is returned when the interface is called to repair an infected text file.
Test pre-conditions:
For validation plug-in only.
Repairing functionality is required in validation plug-in.
<u>Test Procedure:</u>
1. Call TCSLibraryOpen().
2. Call TCSScanFile() with an infected text file path, TCS_SA_SCANREPAIR as the scan action ID and TCS_DTYPE_TEXT as the data type identifier.
3. Verify that the return value of TCSScanFile() is 0.
4. Verify that the content file is repaired by comparing with prepared clean file.
5. Call pfFreeResult() to release the resource returned by TCS library.
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().
Test PASS Condition:
Step 3 should pass verification.
Step 4 should pass verification.
Test Clean-up procedure:
No specific cleanup required.

# 5.81 Test Case TC\_SEC\_CS\_TCSScanFile\_0023

TC_SEC_CS_TCSScanFile_0023	Call TCS interface to repair a file infected by multiple malware.	
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void)	);	
int TCSScanFile(TCSLIB_HANDLE hLik	o, char const *pszFileName,	
int iDataType, int	: iAction, int iCompressFlag,	
TCSScanResult *pRe	esult );	
int TCSLibraryClose(TCSLIB_HANDLE	hLib);	
Test Objectives:		
This test case verifies that the expected return valu multiple malware.	he is returned when the interface is called to repair a file infected by	
Test pre-conditions:		
For validation plug-in only.		
Repairing functionality is required in validation pl	ug-in.	
Test Procedure:		
1. Call TCSLibraryOpen().		
2. Call TCSScanFile() with an infected file path of the file infected by multiple malware, TCS_SA_SCANREPAIR as the scan action ID and TCS_DTYPE_UNKNOWN as the data type identifier.		
3. Verify that the return value of TCSScan	File() is 0.	
4. Verify that the content file is repaired by	comparing with prepared clean file.	
5. Call pfFreeResult() to release the r	esource returned by TCS library.	
6. Call TCSLibraryClose() with the T	CS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Test Clean-up procedure:		

## 5.82Test Case TC\_SEC\_CS\_TCSScanFile\_0024

## TC\_SEC\_CS\_TCSScanFile\_0024

Call TCS interface to repair an infected file where the repair functionality is not implemented in the TCS library.

## API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult );

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

## Test Objectives:

This test case verifies that the expected return value is returned when calling the TCS interface to repair an infected file where the repair functionality is not implemented in the TCS library.

## **Test pre-conditions:**

For validation plug-in only.

Repairing functionality is required to be not implemented in validation plug-in for this test case.

### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with an infected file path and TCS\_DTYPE\_TEXT as the data type identifier, and TCS\_SA\_SCANREPAIR as the scan action ID.
- 3. Verify that the return value of TCSScanFile() is -1.
- 4. Call TCSGetLastError() to get error code.
- 5. Verify that the error code returned by TCSGetLastError() is TCS\_ERROR\_NOT\_IMPLEMENTED.

6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

#### **Test PASS Condition:**

Step 3 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

# 5.83Test Case TC\_SEC\_CS\_TCSScanFile\_0025

TC_SEC_CS_TCSScanFile_0025	Call TCS file scan interface with an invalid library instance handle.	
API Function(s) covered:		
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,		
int iDataType, int	iAction, int iCompressFlag,	
TCSScanResult *pRes	sult );	
Test Objectives:		
This test case verifies that $-1$ is returned when an invalid scanner instance handle is passed to the TCS file scan interface.		
Test pre-conditions:		
For validation plug-in only.		
Test Procedure:		
1. Call TCSScanFile() with an invalid TCS scanner instance handle INVALID_TCSLIB_HANDLE.		
2. Verify that the return value of TCSScanFile() is -1.		
Test PASS Condition:		
Step 2 should pass verification.		
Test Clean-up procedure:		
No specific cleanup required.		

## 5.84Test Case TC\_SEC\_CS\_TCSScanFile\_0026

## TC\_SEC\_CS\_TCSScanFile\_0026 Concurrency TCS file scan test.

#### API Function(s) covered:

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult );

#### **Test Objectives:**

This test case verifies that TCSScanFile() can be correctly handled by multiple scanner instance handles in multiple threads.

#### Test pre-conditions:

For validation plug-in only.

#### **Test Procedure:**

- 1. Create multiple threads to execute from 2 to 10.
- 2. Call TCSLibraryOpen().
- 3. Call TCSScanFile() with an infected file, TCS\_SA\_SCANONLY as the scan action ID, TCS\_DTYPE\_UNKNOWN as the data type identifier.
- 4. Verify that the return value of TCSScanFile() is 0.
- 5. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
- 6. Call pfFreeResult() to release the resource returned by TCS library.
- 7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().
- 8. Repeat 2 ~ 9 with different parameter for TCSScanFile(), other test samples: (html, url, email, phone number, Java code, text) and respective data type identifier.

## **Test PASS Condition:**

Step 4 should pass verification.

Step 5 should pass verification.

#### Test Clean-up procedure:

# 5.85 Test Case TC\_SEC\_CS\_TCSScanFile\_0027

TC_SEC_CS_TCSScanFile_0027 Concurrency TCS file clean test.			
API Function(s) covered:			
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,			
int iDataType, int iAction, int iCompressFlag,			
TCSScanResult *pResult );			
Test Objectives:			
This test case verifies that TCSScanFile() can be correctly handled by multiple scanner instance handles in multiple threads.			
Test pre-conditions:			
For validation plug-in only.			
Repairing functionality is required in validation plug-in.			
Test Procedure:			
1. Create multiple threads to execute from 2 to 10.			
2. Call TCSLibraryOpen().			
3. Call TCSScanFile() with an infected file, TCS_SA_SCANREPAIR as the scan action ID, TCS_DTYPE_UNKNOWN as the data type identifier.			
4. Verify that the return value of TCSScanFile() is 0.			
5. Verify that the file is repaired by comparing with the respective clean file if the input file is supposed to be infected.			
6. Call pfFreeResult() to release the resource returned by TCS library.			
7. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().			
8. Repeat 2 ~ 9 with different parameter for TCSScanFile(), other test samples: (html, url, email, phone number, java code, text) and respective data type identifier.			
Test PASS Condition:			
Step 4 should pass verification.			
Step 5 should pass verification.			
Test Clean-up procedure:			
No specific cleanup required.			

## 5.86 Test Case TC\_SEC\_CS\_TCSScanFile\_0028

file.		Call TCS interface to scan a benign JavaScript file.
-------	--	--

TC_	_SEC_	_CS_	_TCSScanFile_0028	
-----	-------	------	-------------------	--

#### API Function(s) covered:

TCSLIB\_HANDLE TCSLibraryOpen(void);

int TCSScanFile(TCSLIB\_HANDLE hLib, char const \*pszFileName,

int iDataType, int iAction, int iCompressFlag,

TCSScanResult \*pResult);

int TCSLibraryClose(TCSLIB\_HANDLE hLib);

#### **Test Objectives:**

This test case verifies that the expected return value is returned when the interface is called to scan a benign JavaScript file.

### Test pre-conditions:

For validation plug-in only.

#### **Test Procedure:**

- 1. Call TCSLibraryOpen().
- 2. Call TCSScanFile() with a benign Java file path, TCS\_SA\_SCANONLY as the scan action ID and TCS\_DTYPE\_JAVAS as the data type identifier.
- 3. Verify that the return value of TCSScanFile() is 0.
- 4. Verify that the number of the detected malware is 0.
- 5. Call pfFreeResult () to release the resource returned by TCS library.
- 6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().

### **Test PASS Condition:**

Step 3 should pass verification.

Step 4 should pass verification.

#### Test Clean-up procedure:

# 5.87Test Case TC\_SEC\_CS\_TCSScanFile\_0029

TC_SEC_CS_TCSScanFile_0029       Call TCS interface to scan an infected JavaScript file.
API Function(s) covered:
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,
int iDataType, int iAction, int iCompressFlag,
TCSScanResult *pResult );
int TCSLibraryClose(TCSLIB_HANDLE hLib);
Test Objectives:         This test case verifies that the expected value is returned when the interface is called to scan an infected JavaScript file.
Test pre-conditions:         For validation plug-in only.         Test Procedure:
1. Call TCSLibraryOpen().
2. Call TCSScanFile() with an infected Java file path, TCS_SA_SCANONLY as the scan action ID and TCS_DTYPE_JAVAS as the data type identifier.
3. Verify that the return value of TCSScanFile() is 0.
4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
5. Call pfFreeResult() to release the resource returned by TCS library.
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().
Test PASS Condition:
Step 3 should pass verification.
Step 4 should pass verification.
Test Clean-up procedure:
No specific cleanup required.

## 5.88Test Case TC\_SEC\_CS\_TCSScanFile\_0030

TC_SEC_CS_TCSScanFile_0030	30
----------------------------	----

Call TCS interface to scan a benign file with compress flag.

	II TCS interface to scan a benign file with	
API Function(s) covered:	npress flag.	
TCSLIB_HANDLE TCSLibraryOpen(void);		
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,		
int iDataType, int iA	ction, int iCompressFlag,	
TCSScanResult *pResul	t);	
int TCSLibraryClose(TCSLIB_HANDLE hLi		
Test Objectives:		
	eturned when the interface is called to scan a benign file.	
Test pre-conditions:		
For validation plug-in only.		
Test Procedure:		
1. Call TCSLibraryOpen().		
<ol> <li>Call TCSScanFile() with a benign Java file TCS_DTYPE_UNKOWN as the data type identity</li> </ol>	e path, TCS_SA_SCANONLY as the scan action ID and ier, and compress flag to 1.	
3. Verify that the return value of TCSScanFile	() is 0.	
4. Verify that the number of the detected malware	e is 0.	
5. Call pfFreeResult() to release the resource	e returned by TCS library.	
6. Call TCSLibraryClose() with the TCS lik	prary handle returned by the TCSLibraryOpen().	
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
Test Clean-up procedure:		
No specific cleanup required.		

# 5.89Test Case TC\_SEC\_CS\_TCSScanFile\_0031

TC_SEC_CS_TCSScanFile_0031       Call TCS interface to scan an infected file with compress flag.
API Function(s) covered:
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,
int iDataType, int iAction, int iCompressFlag,
TCSScanResult *pResult );
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>
Test Objectives:
This test case verifies that the expected value is returned when the interface is called to scan an infected file.
For validation plug-in only.          Test Procedure:         1. Call TCSLibraryOpen().
2. Call TCSScanFile() with an infected Java file path, TCS_SA_SCANONLY as the scan action ID and TCS_DTYPE_UNKNOWN as the data type identifier, and compress flag to 1.
3. Verify that the return value of TCSScanFile() is 0.
4. Verify that the number of the detected malware is as expected, the malware name or variant name is as expected and the severity/behaviour is as expected.
5. Call pfFreeResult() to release the resource returned by TCS library.
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().
Test PASS Condition:
Step 3 should pass verification.
Step 4 should pass verification.
Test Clean-up procedure:
No specific cleanup required.

## 5.90 Test Case TC\_SEC\_CS\_TCSScanFile\_0032

TC_SEC_CS_TCSScanFile_0032	
----------------------------	--

Call TCS interface to scan a benign file with compress flag.

TC_SEC_CS_TCSScanFile_0032	Call TCS interface to scan a benign file with compress flag.	
API Function(s) covered:		
TCSLIB_HANDLE TCSLibraryOpen(void)	7	
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,		
int iDataType, int	: iAction, int iCompressFlag,	
TCSScanResult *pRe	esult);	
int TCSLibraryClose(TCSLIB_HANDLE	hLib);	
Test Objectives:		
This test case verifies that the expected return valu	e is returned when the interface is called to scan a benign file.	
Test pre-conditions:		
For validation plug-in only.		
Test Procedure:		
1. Call TCSLibraryOpen().		
<ol> <li>Call TCSScanFile() with a benign Ja TCS_DTYPE_UNKOWN as the data type i</li> </ol>	va file path, TCS_SA_SCANONLY as the scan action ID and dentifier, and compress flag to 0.	
3. Verify that the return value of TCSScan	File() is 0.	
4. Verify that the number of the detected ma	alware is 0.	
5. Call pfFreeResult() to release the re	esource returned by TCS library.	
6. Call TCSLibraryClose() with the Te	CS library handle returned by the TCSLibraryOpen().	
Test PASS Condition:		
Step 3 should pass verification.		
Step 4 should pass verification.		
<u>Test Clean-up procedure:</u>		

No specific cleanup required.

# 5.91 Test Case TC\_SEC\_CS\_TCSScanFile\_0033

TC_SEC_CS_TCSScanFile_0033	Call TCS interface to scan an infected file with compress flag.				
API Function(s) covered:					
<pre>TCSLIB_HANDLE TCSLibraryOpen(void);</pre>					
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,					
int iDataType, int iAction, int iCompressFlag,					

TC_SEC_CS_TCSScanFile_0033	Call TCS interface to scan an infected file with compress flag.					
TCSScanResult *pResult);						
<pre>int TCSLibraryClose(TCSLIB_HANDLE hLib);</pre>						
Test Objectives:						
This test case verifies that the expected return value is returned when the interface is called to scan a infected file.						
Test pre-conditions:	Test pre-conditions:					
For validation plug-in only.						
<u>Test Procedure:</u>						
1. Call TCSLibraryOpen().	1. Call TCSLibraryOpen().					
2. Call TCSScanFile() with a benign Java file path, TCS_SA_SCANONLY as the scan action ID and TCS_DTYPE_UNKOWN as the data type identifier, and compress flag to 0.						
3. Verify that the return value of TCSScan	3. Verify that the return value of TCSScanFile() is 0.					
4. Verify that the number of the detected malware is 0.						
5. Call pfFreeResult() to release the resource returned by TCS library.						
6. Call TCSLibraryClose() with the TCS library handle returned by the TCSLibraryOpen().						
Test PASS Condition:						
Step 3 should pass verification.						
Step 4 should pass verification.						
Test Clean-up procedure:						
No specific cleanup required.						

# 5.92Test Case TC\_SEC\_CS\_TCSScanFile\_0034

TC_SEC_CS_TCSScanFile_0034	Stub TCS function error return.				
API Function(s) covered:					
int TCSScanFile(TCSLIB_HANDLE hLib, char const *pszFileName,					
int iDataType, int iAction, int iCompressFlag,					
TCSScanResult *pResult);					
Test Objectives:					
This test case verifies that the calling application can get proper error code from TCS stub functions.					
Test pre-conditions:					

TC_SEC_CS_TCSScanFile_0034	Stub TCS function error return.					
Test Procedure:						
1. Call TCSScanFile() with INVALID_T	CSLIB_HANDLE.					
2. Verify it returns -1.	2. Verify it returns -1.					
Test PASS Condition:						
Step 2 should passed.						
Test Clean-up procedure:						
None.						

# 6 Test Guide

To run test cases, we need to have:

- TCS plug-in for test purpose
- Test contents
- Test cases
- TCS security framework

Test cases need to be compiled with TCS security framework. A TCS plug-in need to be created which can detect the test contents as expected. All test contents, test cases and test TCS plug-in will be provided as a test suite along with accordinate script file which will automate the test process.

# 7 Test Contents

Sample Name	Status	Content Type	Malware Name	Variant Name	Severity Class	Behavior Class
tcs-testfile- 0.buf	clean	Unknown	n/a	n/a	n/a	n/a
tcs-testfile- 0.class	clean	Java	n/a	n/a	n/a	n/a
tcs-testfile- 0.email	clean	Email	n/a	n/a	n/a	n/a
tcs-testfile- 0.html	clean	HTML	n/a	n/a	n/a	n/a
tcs-testfile- 0.js	clean	JavaScript	n/a	n/a	n/a	n/a
tcs-testfile- 0.phone	clean	Phone Number	n/a	n/a	n/a	n/a
tcs-testfile- 0.txt	clean	Text	n/a	n/a	n/a	n/a
tcs-testfile- 0.url	clean	URL	n/a	n/a	n/a	n/a
tcs-testfile- 0.z	clean	Archived	n/a	n/a	n/a	n/a
tcs-testfile- 0.multiple	clean	Unknown	n/a	n/a	n/a	n/a
tcs-testfile- 1.buf	infected	unknown	Malware- fortest- 1.6.0	Variant- fortest- 1.6.0	TCS_SC_USER	TCS_BC_LEVEL1
tcs-testfile- 1.class	infected	Java	Malware- fortest- 1.7.0	Variant- fortest- 1.7.0	TCS_SC_USER	TCS_BC_LEVEL0
tcs-testfile- 1.email	infected	Email	Malware- fortest- 1.2.0	Variant- fortest- 1.2.0	TCS_SC_TERMINAL	TCS_BC_LEVEL2
tcs-testfile- 1.html	infected	HTML	Malware- fortest- 1.0.0	Variant- fortest- 1.0.0	TCS_SC_USER	TCS_BC_LEVEL0
tcs-testfile- 1.js	infected	JavaScript	Malware- fortest- 1.8.0	Variant- fortest- 1.8.0	TCS_SC_USER	TCS_BC_LEVEL2
tcs-testfile- 1.phone	infected	Phone Number	Malware- fortest- 1.3.0	Variant- fortest- 1.3.0	TCS_SC_TERMINAL	TCS_BC_LEVEL3
tcs-testfile- 1.txt	infected	Text	Malware- fortest- 1.4.0	Variant- fortest- 1.4.0	TCS_SC_TERMINAL	TCS_BC_LEVEL4
tcs-testfile- 1.url	infected	URL	Malware- fortest-	Variant- fortest-	TCS_SC_USER	TCS_BC_LEVEL1

			1.1.0	1.1.0		
tcs-testfile- 1.z	infected	Archived	Malware- fortest- 1.9.0	Variant- fortest- 1.9.0	TCS_SC_USER	TCS_BC_LEVEL2
tcs-testfile- 1.multiple	infected	Unknown	Malware- fortest- 1.6.0	Variant- fortest- 1.6.0	TCS_SC_USER	TCS_BC_LEVEL1
			Malware- fortest- 1.5.0	Variant- fortest- 1.5.0	TCS_SC_USER	TCS_BC_LEVEL0