Tizen Web Protection Service Daemon

Test Specification

Document version 1.0.0

1. Contents

1.1 Document History

Version	Date	Reason
1.0.0	6/24/2014	First draft from McAfee

1.2 References

Ref	Document	Issue	Title
[1]	Tizen Web Protection API Specification	1.1.1	Tizen Web Protection API Specification

1.3 Glossary and definitions

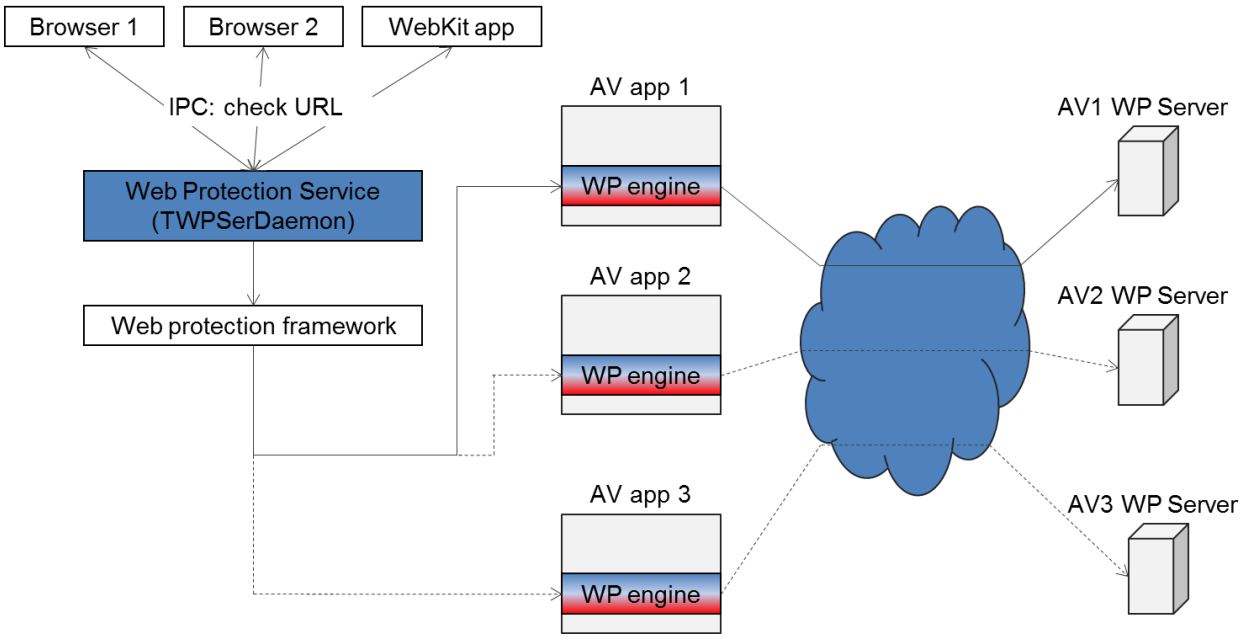
API Application Programming Interface

TWP Tizen Web Protection

2. Purpose and Scope

The overall purpose of this document is to describe the test cases for the Tizen Web Protection Service Daemon. This document includes Test case procedures.

3. Component Description



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 Web Protection 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_WP_TWPSerDaemonGetVerSendMessageN_001

TC_SEC_WP_TWPSerDaemonGetVerSendMessageN_001	TWPSerDaemon interface test.
API Function(s) covered: `int TSCSendMessageN(IpcClientInfo *pInfo, const char *service_name, const char *szMethod, int argc, char **argv, int *argc_reply, char ***argv_reply, int timeout_milliseconds);`	
Test Objectives: This test case verifies that the calling application gets error if TWPSerDaemon is not running.	
Test pre-conditions: No Daemon running.	
Test Procedure: 1. Call `TSCSendMessageN()` with TWPGETVERSIONMETHOD 2. Verify the API return `value` is not equal to 0.	
Test PASS Condition: Step 2 should return a value not equal to 0.	
Test Clean-up procedure: Cleanup Reply.	

5.2 Test Case TC_SEC_WP_TWPSerDaemonGetVerSendMessageN_002

TC_SEC_WP_TWPSerDaemonGetVerSendMessageN_002	TWPSerDaemon interface test.
API Function(s) covered: `int TWPSerGetVersion(void *pData, int req_argc, char **req_argv, char ***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);`	

TC_SEC_WP_TWPSerDaemonGetVerSendMessageN_002	TWPSerDaemon interface test.

Test Objectives:

This test case verifies that the calling application gets correct result if TWPSerDaemon is running.

Test pre-conditions:

Daemon running.

Test Procedure:

1. Start the Daemon.
2. Call `TSCSendMessageN()` with TWPGETVERSIONMETHOD
3. Verify the API return value is equal to 0 and number of reply arguments is 4.
4. Verify status code set to 0.
5. Verify framework version equal to TWP_FRAMEWORK_VERSION.
6. Verify plugin version is greater than zero.
7. Verify Daemon version is equal to TWP_DAEMON_VERSION.

Test PASS Condition:

Step 3, 4, 5, 6 and 7 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop the Daemon.

5.3 Test Case TC_SEC_WP_TWPSerDaemonGetVerSendMessageN_003

TC_SEC_WP_TWPSerDaemonGetVerSendMessageN_003	TWPSerDaemon interface test.

API Function(s) covered:

```
int TWPSerGetVersion(void *pData, int req_argc, char **req_argv, char
***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);
```

Test Objectives:

This test case verifies that the calling application gets correct result everytime when TWPSerDaemon is running.

Test pre-conditions:

TC_SEC_WP_TWPSerDaemonGetVerSendMessageN_003	TWPSerDaemon interface test.
Daemon running.	
Test Procedure: 1. Start the Daemon. 2. Do below steps for 500 times. 3. Call `TSCSendMessageN()` with TWPGETVERSIONMETHOD 4. Verify the API return value is equal to 0 and number of reply arguments is 4. 5. Verify status code is set to 0. 6. Verify framework version equal to TWP_FRAMEWORK_VERSION. 7. Verify plugin version is greater than zero. 8. Verify Daemon version is equal to TWP_DAEMON_VERSION.	
Test PASS Condition: Step 2, 3, 4, 5, 6, 7 and 8 should pass.	
Test Clean-up procedure: Cleanup Reply. Stop Daemon.	

5.4 Test Case TC_SEC_WP_TWPSerDaemonGetVerSendMessageN_004

TC_SEC_WP_TWPSerDaemonGetVerSendMessageN_004	TWPSerDaemon interface test.
API Function(s) covered: `int TWPSerGetVersion(void *pData, int req_argc, char **req_argv, char ***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);`	
Test Objectives: This test case verifies that the calling application succeeds but does not get plugin version if TWPSerDaemon is running and there is no plugin.	
Test pre-conditions: Daemon running and Plugin removed.	

TC_SEC_WP_TWPSerDaemonGetVerSendMessageN_004	TWPSerDaemon interface test.

Test Procedure:

1. Start the Daemon.
2. Remove Plugin.
3. Call `TSCSendMessageN()` with TWPGETVERSIONMETHOD
4. Verify the API return value is equal to 0 and number of reply arguments is 1.

Test PASS Condition:

Step 3, 4 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop the Daemon.

5.5 Test Case TC_SEC_WP_TWPSerDaemonGetVerSendMessageAsync_001

TC_SEC_WP_TWPSerDaemonGetVerSendMessageAsync_001	TWPSerDaemon interface test.

API Function(s) covered:

```
int TWPSerGetVersion(void *pData, int req_argc, char **req_argv, char
***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);
```

Test Objectives:

This test case verifies that the calling application gets correct result by using TSCSendMessageN in different threads when TWPSerDaemon is running.

Test pre-conditions:

Daemon running.

Test Procedure:

1. Start the Daemon.
2. Do below steps in 200 threads.
3. Call `TSCSendMessageN()` with TWPGETVERSIONMETHOD.
4. Verify the API return value is equal to 0 and number of reply arguments is 4.

TC_SEC_WP_TWPSerDaemonGetVerSendMessageAsync_001	TWPSerDaemon interface test.

5. Verify status code is set to 0.
6. Verify framework version equal to TWP_FRAMEWORK_VERSION.
7. Verify plugin version is greater than zero.
8. Verify Daemon version is equal to TWP_DAEMON_VERSION.

Test PASS Condition:

Step 2, 3, 4, 5, 6, 7 and 8 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop Daemon.

5.6 Test Case TC_SEC_WP_TWPSerDaemonGetVerSendMessageAsync_002

TC_SEC_WP_TWPSerDaemonGetVerSendMessageAsync_002	TWPSerDaemon interface test.

API Function(s) covered:

```
int TWPSerGetVersion(void *pData, int req_argc, char **req_argv, char
***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);
```

Test Objectives:

This test case verifies that the calling application gets correct result by using TSCSendMessageAsync in different threads when TWPSerDaemon is running.

Test pre-conditions:

Daemon running.

Test Procedure:

1. Start the Daemon.
2. Do below steps in 100 threads.
3. Call `TSCSendMessageAsync()` with TWPGETVERSIONMETHOD.
4. Verify the API return value is equal to 0 and number of reply arguments is 4.
5. Verify status code is set to 0.
6. Verify framework version equal to TWP_FRAMEWORK_VERSION.

TC_SEC_WP_TWPSerDaemonGetVerSendMessageAsync_002	TWPSerDaemon interface test.
7. Verify plugin version is greater than zero. 8. Verify Daemon version is equal to TWP_DAEMON_VERSION.	
Test PASS Condition: Step 2, 3, 4, 5, 6, 7 and 8 should pass.	
Test Clean-up procedure: Cleanup Reply. Stop Daemon.	

5.7 Test Case TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_001

TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_001	TWPSerDaemon interface test.
API Function(s) covered: `int TWPSerGetURLReputation(void *pData, int req_argc, char **req_argv, char ***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);`	
Test Objectives: This test case verifies that the calling application gets error if TWPSerDaemon is not running.	
Test pre-conditions: No Daemon running.	
Test Procedure: 1. Call `TSCSendMessageN()` with TWPGETURLREPUTATIONMETHOD. 2. Verify the API return `value` is not equal to 0.	
Test PASS Condition: Step 2 should return a value not equal to 0.	
Test Clean-up procedure: Cleanup Reply.	

5.8 Test Case TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_002

TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_002	TWPSerDaemon interface test.

API Function(s) covered:

```
int TWPSerGetURLReputation(void *pData, int req_argc, char **req_argv, char
***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);
```

Test Objectives:

This test case verifies that the calling application gets correct result for high risk URL if TWPSerDaemon is running.

Test pre-conditions:

Daemon running.

Test Procedure:

1. Start the Daemon.
2. Call `TSCSendMessageN()` with TWPGETURLREPUTATIONMETHOD.
3. Verify the API return value is equal to 0 and number of reply arguments is 3.
4. Verify status code is set to 0.
5. Verify risk level is set to TWP_High.
6. Verify redirect URL length is greater than zero.

Test PASS Condition:

Step 3, 4, 5, and 6 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop the Daemon.

5.9 Test Case TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_003

TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_003	TWPSerDaemon interface test.

API Function(s) covered:

```
int TWPSerGetURLReputation(void *pData, int req_argc, char **req_argv, char
***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);
```

Test Objectives:

This test case verifies that the calling application gets correct result for Medium Risk URL if TWPSerDaemon is running.

Test pre-conditions:

Daemon running.

Test Procedure:

1. Start the Daemon.
2. Call `TSCSendMessageN()` with TWPGETURLREPUTATIONMETHOD.
3. Verify the API return value is equal to 0 and number of reply arguments is 3.
4. Verify status code is set to 0.
5. Verify risk level is set to TWP_Medium.
6. Verify redirect URL length is greater than zero.

Test PASS Condition:

Step 3, 4, 5, and 6 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop the Daemon.

5.10 Test Case TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_004

TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_004	TWPSerDaemon interface test.

API Function(s) covered:

```
int TWPSerGetURLReputation(void *pData, int req_argc, char **req_argv, char
***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);
```

TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_004	TWPSerDaemon interface test.

Test Objectives:

This test case verifies that the calling application gets correct result for Unverified Risk URL.if TWPSerDaemon is running.

Test pre-conditions:

Daemon running.

Test Procedure:

1. Start the Daemon.
2. Call `TSCSendMessageN()` with TWPGETURLREPUTATIONMETHOD.
3. Verify the API return value is equal to 0 and number of reply arguments is 2.
4. Verify status code is set to 0.
5. Verify risk level is set to TWP_Unverified.

Test PASS Condition:

Step 3, 4, and 5 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop the Daemon.

5.11 Test Case TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_005

TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_005	TWPSerDaemon interface test.

API Function(s) covered:

```
int TWPSerGetURLReputation(void *pData, int req_argc, char **req_argv, char
***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);
```

Test Objectives:

This test case verifies that the calling application gets correct result for Minimal Risk URL if TWPSerDaemon is running.

Test pre-conditions:

TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_005	TWPSerDaemon interface test.
Daemon running.	
Test Procedure: 1. Start the Daemon. 2. Call `TSCSendMessageN()` with TWPGETURLREPUTATIONMETHOD. 3. Verify the API return value is equal to 0 and number of reply arguments is 2. 4. Verify status code is set to 0. 5. Verify risk level is set to TWP_Minimal.	
Test PASS Condition: Step 3, 4, and 5 should pass.	
Test Clean-up procedure: Cleanup Reply. Stop the Daemon.	

5.12 Test Case TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_006

TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_006	TWPSerDaemon interface test.
API Function(s) covered: `int TWPSerGetURLReputation(void *pData, int req_argc, char **req_argv, char ***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);`	
Test Objectives: This test case verifies that the calling application gets correct error result if TWPSerDaemon is running and there is no plugin.	
Test pre-conditions: Daemon running. No Plugin.	

TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_006	TWPSerDaemon interface test.

Test Procedure:

1. Start the Daemon.
2. Remove Plugin.
3. Call `TSCSendMessageN()` with TWPGETURLREPUTATIONMETHOD.
4. Verify the API return value is equal to 0 and number of reply arguments is 1.
5. Verify status code is set to 500.

Test PASS Condition:

Step 3, 4, and 5 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop the Daemon.

Restore Plugin.

5.13 Test Case TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_007

TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_007	TWPSerDaemon interface test.

API Function(s) covered:

```
int TWPSerGetURLReputation(void *pData, int req_argc, char **req_argv, char
***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);
```

Test Objectives:

This test case verifies that the calling application gets correct result everytime if TWPSerDaemon is running.

Test pre-conditions:

Daemon running.

Test Procedure:

1. Start the Daemon.
2. Do below steps for 200 times.
3. Call `TSCSendMessageN()` with TWPGETURLREPUTATIONMETHOD and High Risk URL.

TC_SEC_WP_TWPSerDaemonGetRepSendMessageN_007	TWPSerDaemon interface test.
4. Verify the API return value is equal to 0 and number of reply arguments is 3. 5. Verify status code is set to 0. 6. Verify risk level is set to TWP_High. 7. Verify redirect URL length is greater than zero.	
Test PASS Condition: Step 2, 3, 4, 5, 6 and 7 should pass.	
Test Clean-up procedure: Cleanup Reply. Stop the Daemon.	

5.13 Test Case TC_SEC_WP_TWPSerDaemonGetRepSendMessageAsync_001

TC_SEC_WP_TWPSerDaemonGetRepSendMessageAsync_001	TWPSerDaemon interface test.
API Function(s) covered: `int TWPSerGetURLReputation(void *pData, int req_argc, char **req_argv, char ***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);`	
Test Objectives: This test case verifies that the calling application gets correct result by using TSCSendMessageN in parallel threads if TWPSerDaemon is running.	
Test pre-conditions: Daemon running.	
Test Procedure: 1. Start the Daemon. 2. Do below steps for 200 different threads. 3. Call `TSCSendMessageN()` with TWPGETURLREPUTATIONMETHOD and High Risk URL. 4. Verify the API return value is equal to 0 and number of reply arguments is 3. 5. Verify status code is set to 0.	

TC_SEC_WP_TWPSerDaemonGetRepSendMessageAsync_001	TWPSerDaemon interface test.
6. Verify risk level is set to TWP_High. 7. Verify redirect URL length is greater than zero.	
Test PASS Condition: Step 2, 3, 4, 5, 6 and 7 should pass.	
Test Clean-up procedure: Cleanup Reply. Stop the Daemon.	

5.14 Test Case TC_SEC_WP_TWPSerDaemonGetRepSendMessageAsync_002

TC_SEC_WP_TWPSerDaemonGetRepSendMessageAsync_002	TWPSerDaemon interface test.
API Function(s) covered: `int TWPSerGetURLReputation(void *pData, int req_argc, char **req_argv, char ***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);`	
Test Objectives: This test case verifies that the calling application gets correct result by using TSCSendMessageAsync in parallel threads if TWPSerDaemon is running.	
Test pre-conditions: Daemon running.	
Test Procedure: 1. Start the Daemon. 2. Do below steps for 200 different threads. 3. Call `TSCSendMessageAsync()` with TWPGETURLREPUTATIONMETHOD and High Risk URL. 4. Verify the API return value is equal to 0 and number of reply arguments is 3. 5. Verify status code is set to 0. 6. Verify risk level is set to TWP_High. 7. Verify redirect URL length is greater than zero.	

TC_SEC_WP_TWPSerDaemonGetRepSendMessageAsync_002	TWPSerDaemon interface test.
Test PASS Condition: Step 2, 3, 4, 5, 6 and 7 should pass.	
Test Clean-up procedure: Cleanup Reply. Stop the Daemon.	

5.15 Test Case TC_SEC_WP_TWPSerDaemonGetRepGetVerSendMessageAsync_001

TC_SEC_WP_TWPSerDaemonGetRepGetVerSendMessageAsync_001	TWPSerDaemon interface test.
API Function(s) covered: `int TWPSerGetURLReputation(void *pData, int req_argc, char **req_argv, char ***rep_argv, int *rep_argc, CALLBACKFUNC callback, TSC_METHOD_HANDLE *handle);`	
Test Objectives: This test case verifies that the calling application gets correct result by calling both methods "TWPGETURLREPUTATIONMETHOD" and "TWPGETVERSIONMETHOD" in parallel threads while TWPSerDaemon is running.	
Test pre-conditions: Daemon running.	
Test Procedure: 1. Start the Daemon. 2. Do below steps for 50 different threads. 3. Call `TSCSendMessageAsync()` for method TWPGETURLREPUTATIONMETHOD with argument as High Risk URL. 4. Call `TSCSendMessageAsync()` for method TWPGETVERSIONMETHOD with no arguments. 5. If TWPGETURLREPUTATIONMETHOD's call back is invoked, verify that API return value is equal to 0 and number of reply arguments is set to 3 and steps 6, 7 and 8. 6. Verify status code is set to 0. 7. Verify risk level is set to TWP_High.	

TC_SEC_WP_TWPSerDaemonGetRepGetVerSendMessageAsync_001	TWPSerDaemon interface test.

8. Verify redirect URL length is greater than zero.
9. If TWPGETVERSIONMETHOD's call back is invoked, verify that API return value is equal to 0 and number of reply arguments is set to 4 and steps 10, 11, 12 and 14.
10. Verify status code set to 0.
11. Verify framework version equal to TWP_FRAMEWORK_VERSION.
12. Verify plugin version is greater than zero.
13. Verify Daemon version is equal to TWP_DAEMON_VERSION.

Test PASS Condition:

Steps 2 to 13 should pass.

Test Clean-up procedure:

Cleanup Reply.

Stop the Daemon.

6. Test Guide

To run test cases, we need to have:

- TWP plug-in for test purpose
- Test contents
- Test cases
- TWP security framework
- TWP Service Daemon

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

7. Test Contents

URL	Risk Level
http://www.zcrack.com	TWP_High
http://www.thepiratebay.se	TWP_Medium
http://www.targetingnow.com/delivery	TWP_Unverified
http://www.google.com	TWP_Minimal