Protection Profile for Web Browsers

31 March 2014

Version 1.0

Table of Contents

Revision History

Version	Date	Description
1.0	31 March 2014	Web Browser Client PP

1 Introduction

Web browsers are client applications that retrieve and render content provided by web servers, primarily using the hypertext transfer protocol (HTTP) or HTTP Secure (HTTPS). Browsers have grown in complexity over the years, starting as tools used to display simple, unchanging web pages and becoming sophisticated execution environments for web content. The use of browsers to administer accounts, servers or embedded systems remotely requires them to handle sensitive information securely. Innovations such as tabs, extensions and HTML5 have not only increased browser functionality, but also introduced new security concerns. Being the principal method for accessing the Internet, and due to their complexity and the information that they process, browsers are a natural target for attackers. As a result, it is paramount that the security of web browsers be improved to reduce the risk to client machines and enterprise networks.

This document provides a baseline set of Security Functional Requirements (SFRs) for a web browser client. It is intended to improve the security of browsers by encouraging the use of operating system security services and requiring the use of sandboxing technologies and environmental mitigations provided by the underlying platform. Additionally, these requirements define security functionality that browsers must provide.

The requirements in this document apply to all web browsers that run on any operating system, regardless of the composition of the underlying platform.

1.1 Overview of the TOE

The Target of Evaluation (TOE) in this document is any web browser client capable of running on any operating system or platform and used primarily to render web content using HTTP and HTTPS.

1.2 Usage of the TOE

Web browsers are used to perform many tasks that can be categorized into three primary use cases.

[USE CASE 1] Surfing the web

Browsers are used to retrieve and display content on the web, such as web pages, streaming media, images and specialized formats (e.g., Java, Flash, Word, PDF). They can also be used to write content to web sites (web 2.0 – e.g., Facebook). Web surfing can be done over the Internet or Intranet.

[USE CASE 2] Remote Administration Client

Browsers are used to provide remote administration interfaces for systems such as servers, network devices and embedded systems, such as SCADA, smart TVs and thermostats. As opposed to surfing the web, where the browser is interacting with unknown servers, the browser, acting as a Remote Administration Client, is connecting to a server that the user trusts.

[USE CASE 3] Content Creation

Browsers are used to create content via an increasing number of Software as a Service (SaaS) offerings, including Microsoft Office 365, Google Drive, and Adobe Creative Cloud, where user data and records are stored online.

2 SECURITY PROBLEM DESCRIPTION

The following describes the problems that compliant TOEs will address.

2.1 Threats

2.1.1 Malicious or Flawed Updates

Since the most common attack vector used involves attacking unpatched versions of software containing well-known flaws, updating the browser is necessary to ensure that changes to threat environment are addressed. Timely application of patches ensures that the client is a “hard target”, thus increasing the likelihood that product will be able to maintain and enforce its security policy. However, the updates to be applied to the product must be trustable in some manner; otherwise, an attacker can write their own “update” that instead contains malicious code of their choosing, such as a rootkit, bot, or other malware. Once this “update” is installed, the attacker then has control of the system and all of its data.

[T.UNAUTHORIZED_UPDATE]

2.1.2 Malicious or Flawed Add-on

Web browser functionality can be extended with integration of third-party utilities and tools. This expanded set of capabilities is made possible via the use of browser plug-ins and extensions. The tight integration between the basic browser code and the new capabilities that plug-ins and extensions provide increases the risk that they could inject serious flaws into the browser application, either maliciously by an attacker, or accidentally by a developer. These flaws enable undesirable behaviors, including, but not limited to, allowing unauthorized access to sensitive information in the browser, unauthorized access to the device’s file system, or even privilege escalation that enables unauthorized access to other applications or the operating system.

[T.UNAUTHORIZED_ADD-ON]

2.1.3 Network Eavesdropping

Network eavesdropping involves an attacker positioning himself on the network in order to monitor transmissions between the system and the intended destination of some potentially sensitive data. With respect to web browsers, this entails monitoring the transactions between the browser and one or more web servers, such as transmissions between a user attempting to pay a utility bill and the utility’s website.

[T.NETWORK_EAVESDROP]

2.1.4 Network Attack

Network attack is similar to network eavesdropping in that it entails an attacker positioning herself on the network. It differs from network eavesdropping in that it involves the attacker initiating communications with the target system, or modifying data between the target system and the data’s legitimate destination. With respect to browsers, network attack might involve

sending malicious data to the browser in order to exploit vulnerabilities that may influence its behavior, or modifying account information en route to a web server.

Browser attacks generally occur on Internet connected browsers, but are not unknown to occur within closed networks. Attackers can use phishing or another social engineering technique to persuade a user to visit a malicious site. Users may also unintentionally or intentionally visit malicious sites in the course of web browsing. The site then presents malicious content to the user's browser to exploit it and perform installation of malware, often with no indication to the user. These attacks depend on vulnerabilities in the browser or browser extensions.

Some examples of network attacks are:

- Insufficient protection of session tokens can lead to session hijacking, where a token is captured and reused in order to gain the privileges of the user who initiated the session.
- Cross-site scripting (XSS) and cross-site request forgery (CSRF) attacks are methods used to compromise user credentials (usually by stealing the user's session token) to a web site. These attacks are more likely a result of server security problems, but some browsers incorporate technologies that try to detect the attacks.
- Inadequate sandboxing of browser tabs/windows or a faulty cross domain communications model can lead to leakage of content from one domain (e.g., cnn.com) in one tab/window to a different domain (e.g., google.com) in a different tab/window. This is an exploit of the ability of browsers to display content from multiple domains simultaneously and can be carried out via scripts.

[T.NETWORK_ATTACK]

2.1.5 Data Access

Access to a web browser while it is in operation may give rise to loss of confidentiality and/or integrity of user data stored by the web browser. Browser data such as cookie cache, history, web form data, etc. could be accessed by these attacks.

[T.DATA_ACCESS]

2.2 Assumptions

The assumptions for the TOE are defined in Annex A.1.1.

3 Security Objectives

Compliant TOEs will provide security functionality to address security objectives as enumerated below, and to implement policies that address additional threats to the TOE. The following sections provide a description of this functionality, given the threats enumerated above.

3.1 Security Objectives of the TOE

3.1.1 O.COMMS Protected Communications

To address the network attack and network eavesdropping threats, the TOE must provide for protected communications between the browser and a given web server in instances where such protected communications are desirable. The data between these two entities in the operational environment are protected via a trusted path, implemented using one or more of these standard protocols: HTTPS, Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS).

FCS_CKM.1(*),FCS_CKM_EXT.4, FCS_COP.1(*), FCS_DTLS_EXT.1, FCS_HTTPS_EXT.1, FCS_TLSC_EXT.1, FCS_STS_EXT.1, FDP_STR_EXT.1, FIA_X509_EXT.1, FIA_X509_EXT.2, FPT_INT_EXT.3, FTP_ITC.1

3.1.2 O.ISOLATION Domain Isolation

To address the network attack associated with content leakage between different web domains rendered by the browser, the TOE must ensure that content originating from different domains (e.g., in a tab or iFrame) is properly isolated to disallow access.

FDP_ACF_EXT.1, FDP_SBX_EXT.1, FDP_SOP_EXT.1

3.1.3 O.CONFIG TOE Configuration

In order to protect sensitive data stored (either temporarily or permanently) or processed by the browser, conformant TOEs will provide the capability to define, configure, and apply security policies defined by the administrator. If enterprise policies are configured by the administrator for the TOE, these must take precedence over any user-defined settings.

FDP_ACC_EXT.1, FDP_TRK_EXT.1, FMT_MOF.1, FMT_SMF.1, FMT_SMR.1

3.1.4 O.INTEGRITY Integrity of TOE

To ensure the integrity of the browser is maintained, conformant TOEs will perform self-tests to insure the integrity of software and data has been maintained.

To address issues associated with malicious or flawed browser software, plug-ins or extensions, conformant TOEs must implement mechanisms to ensure the integrity of browser software, plug-ins and extensions, and to ensure that they come from legitimate sources. The TOE must

provide mechanisms and enforce policies that enable any browser software, plug-ins and extensions, as well as any subsequent updates to them, to be verified upon installation and execution. In addition, the TOE shall also control the downloading and launching of executables.

FAU_GEN.1, FAU_SEL.1, FCS_COP.1(2), FCS_COP.1(3), FPT_DNL_EXT.1, FPT_DNL_EXT.2, FPT_INT_EXT.1, FPT_INT_EXT.2, FPT_MCD_EXT.1, FPT_TUD_EXT.1, FPT_TUD_EXT.2, FPT_TUD_EXT.3

3.1.5 O.STORAGE Secure Storage of Sensitive Information

Browsers handle many types of potentially sensitive user information (e.g., passwords, web form data, cryptographic keys). It is critical that browsers protect this information when it is stored. The browser shall make use of platform encryption and authentication mechanisms and libraries to protect this information rather than mechanisms and libraries that are part of the browser itself.

FCS_COP.1(1), FCS_CKM_EXT.1, FCS_CKM_EXT.4, FCS_COP.1(4), FDP_COO_EXT.1, FDP_DEL_EXT.1, FDP_DEL_EXT.2, FDP_DEL_EXT.3, FDP_PBR_EXT.1, FDP_PST_EXT.1

4 Security Requirements

Some of the Security Functional Requirements included in this section are derived from Part 2 of the *Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4*, with additional extended functional components.

4.1 Conventions

The CC defines operations on Security Functional Requirements: assignments, selections, assignments within selections and refinements. This document uses the following font conventions to identify the operations defined by the CC:

- Assignment: Indicated with *italicized* text;
- Refinement made by PP author: Indicated by the word “Refinement” in **bold text** after the element number with additional text in **bold** text and deletions with strikethroughs, if necessary;
- Selection: Indicated with underlined text;
- Assignment within a Selection: Indicated with *italicized and underlined* text;
- Iteration: Indicated by appending the iteration number in parenthesis, e.g., (1), (2), (3).

Extended SFRs are identified by having a label ‘EXT’ after the requirement name for TOE SFRs.

4.2 Security Functional Requirements

This section addresses those Security Functional Requirements that may be met by the TOE.

4.2.1 Class: User Data Protection (FDP)

Access Control Functions

FDP_ACF_EXT.1 Extended: Local and Session Storage Separation

FDP_ACF_EXT.1 The TOE shall separate local (permanent) and session (ephemeral) storage based on domain, protocol and port:

- Session storage shall be accessible only from the originating tab or window;
- Local storage shall only be accessible from windows and tabs running the same web application.

Application Note:

The separation of local and session storage is described in World Wide Web Consortium (W3C) Proposed Recommendation: “Web Storage.”

In this context, a domain is a realm of administrative autonomy, authority or control on the Internet (e.g., cnn.com, ietf.org). A sub-domain is denoted by a prefix before the top-level domain name (e.g., news.cnn.com). A protocol is a system of digital rules for data exchange within or between computers; in a web environment, the typical protocols are HTTP and HTTPS. A port is an application-specific construct that functions as a communications endpoint in a computer’s host OS; in a web environment, port 80 is the default port for HTTP

communications, although other ports can be used. In a web address, the port follows the domain or sub-domain name (e.g., http://www.cnn.com:80).

Assurance Activity

TSS

The evaluator shall examine the TSS to ensure it describes how the TSF separates local and session storage.

Guidance

The evaluator shall examine the operational guidance to verify that it documents the location on the file system that will be used for local storage and the location used for session storage.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall obtain or create JavaScript-based scripts that store and retrieve information from local and session storage and shall set up a web server with two or more web pages using different protocols and/or ports. The evaluator shall incorporate the scripts into the web pages. The evaluator shall open two or more browser windows and navigate to the same web page. The evaluator shall verify that the script for accessing session storage that is running in one window cannot access session storage associated with a different window.
- Test 2: Using the same web server, the evaluator shall open two or more browser tabs and navigate to the same web page. The evaluator shall verify that the script for accessing session storage that is running in one tab cannot access session storage associated with a different tab.
- Test 3: Using the same web server, the evaluator shall open two or more browser windows and navigate to the same web page. The evaluator shall verify that the script for accessing local storage that is running in one window can access local storage associated with a different window.
- Test 4: Using the same web server, the evaluator shall open two or more browser tabs and navigate to the same web page. The evaluator shall verify that the script for accessing local storage that is running in one tab can access local storage associated with a different tab.
- Test 5: Using the same web server, the evaluator shall open windows and tabs and navigate to different web pages. The evaluator shall verify that a script running in the context of one domain/protocol/port in a window or tab cannot access information associated with a different domain/protocol/port in a different window or tab.

Cookie Handling
FDP_COO_EXT.1 Extended: Cookie Blocking

FDP_COO_EXT.1.1 The TOE shall provide the capability to block the storage of third party cookies by websites.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure it describes how the TSF blocks third party cookies and when the blocking occurs (e.g., automatically, when blocking is enabled).

Guidance

The evaluator shall examine the operational guidance to verify that it provides a description of the configuration option for blocking of third party cookies.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall clear all cookies and then configure the TOE so that storage of third party cookies is allowed. The evaluator shall load a web page that stores a third party cookie. The evaluator shall navigate to the location where cookies are stored and shall verify that the cookie is present.
- Test 2: The evaluator shall clear all cookies and then configure the TOE so that storage of third party cookies is blocked (i.e. not allowed). The evaluator shall load a web page that attempts to store a third party cookie and shall verify that the cookie was not stored.

Information Deletion

FDP_DEL_EXT.1 Extended: Deletion of Browser Data

FDP_DEL_EXT.1.1 The TOE shall provide the capability to delete [selection: browser cache, history, passwords, web form information, cookies, extensions, plug-ins] from the TOE when invoked by the user.

FDP_DEL_EXT.1.2 The TOE shall provide the capability to delete [selection: browser cache, history, passwords, web form information, cookies, extensions, plug-ins] from the TOE when the browser is terminated.

Application Note:

If extensions or plug-ins are selected above, the applicable selection-based requirements from Annex C must also be included in the main body of the ST.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure that it documents where all browser content is stored and what browser content can be deleted. The TSS shall also describe the deletion of browser content when the browser is terminated.

Guidance

The evaluator shall examine the operational guidance to verify that it includes instructions for how the user can delete stored content and select which types of stored content can be deleted. The operational guidance shall also include instructions for ensuring that content is deleted upon browser termination.

Tests

The evaluator shall perform the following tests for each type of stored content:

- Test 1: The evaluator shall set up a web session with a web server that prompts the creation and storage of stored content. The evaluator shall verify that the supported type of content is stored in the documented location. The evaluator shall then delete content via the browser and shall verify that the stored content has been deleted from the documented location.
- Test 2: The evaluator shall verify that browser content is in the specified location. The evaluator shall invoke the TOE, set it to delete content, and shall terminate the TOE. The evaluator shall navigate to the documented location of the content and verify that the stored content has been deleted.

Sandboxing
FDP_SBX_EXT.1 Extended: Sandboxing of Rendering Processes

FDP_SBX_EXT.1.1 The TOE shall ensure that web page rendering is performed in a process that is restricted in the following manner:

- The rendering process cannot directly access the TOE platform's file system;
- The rendering process cannot directly invoke inter-process communication mechanisms with non-TOE processes;
- The rendering process has reduced privilege with respect to other TOE processes [selection: [assignment: *methods by which the principle of least privilege is implement for rendering processes*], in no other ways].

Application Note:

Web browsers implement a variety of methods to ensure that the process that renders HTML and interprets JavaScript operates in a constrained environment in order to reduce the risk that the rendering process can be corrupted by the HTML or JavaScript it is processing. This component requires the TSF to lower the privileges of rendering processes by ensuring that it cannot directly access the file system of the host, and that it cannot use IPC mechanisms provided by the host to communicate with non-TOE processes on the host. Typically, if a rendering process needs to access a file or communicate with a non-TOE process, it must request such access through the TSF (which is allowed by the requirement).

In addition to the two required measures, other measures can be implemented depending on the TOE and the host platform. These may involve such actions as changing the owner of the

rendering process to a low-privileged account or dropping platform-defined privileges in the rendering process. The ST author fills in the additional measures implemented by the TOE.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure it describes how the rendering of HTML and interpretation of JavaScript is performed by the TOE in terms of the platform processes that are involved (with "process" being an active entity that executes code). For the processes that render HTML or interpret JavaScript, the evaluator shall examine the TSS to check that it describes how these processes are prevented from accessing the platform file system. The evaluator shall check the TSS to ensure it describes each platform-provided IPC mechanism, and details for each mechanism how the rendering process is unable to use it to communicate with non-TOE processes. The evaluator shall also confirm that the TSS describes how IPC and file system access is enabled (if this capability is implemented); for instance, through a more privileged TOE process that does not perform web page rendering. The evaluator shall ensure that these descriptions are present for all platforms claimed in the ST.

For each additional mechanism listed in the third bullet of this component by the ST author, the evaluator shall examine the TSS to ensure 1) the mechanisms is described; 2) the description of the mechanism is sufficiently detailed to determine that it contributes to the principle of least privilege being implemented in the rendering process; and 3) appropriate supporting information is provided in the TSS (or pointers to such information are provided) that provides context for understanding the claimed least privilege mechanisms.

Guidance

The evaluator shall examine the operational guidance to determine that it provides a description of the restrictions available on rendering processes. Additionally, if such mechanisms are configurable (for instance, if a user can choose which mechanisms to "turn on"), the evaluator shall examine the operational guidance to ensure that the method for enabling and disabling the mechanisms are provided, and the consequences of such actions are described.

Tests

Note: The following tests require the developer to provide access to a test platform that provides the evaluator with debugging and test tools that are typically not found on consumer platforms.

The evaluator shall perform the following tests on each platform claimed in the ST:

- Test 1: The evaluator shall use debugging or test facilities to introduce code into a rendering process that attempts to directly access the platform's file system, and then direct execution to it. The evaluator shall ensure that execution of this code fails to access the TOE file system.
- Test 2: For each IPC mechanism described in the TSS, the evaluator shall use debugging or test facilities to introduce code into a rendering process that attempts to

directly communicate with another non-TOE process on the platform, and then direct execution to it. The evaluator shall ensure that this attempt fails.

- Test 3: For each additional mechanism claimed in the ST, the evaluator shall devise a test to show that the mechanism functions as described in the TSS. If no such test can be crafted, the evaluator shall provide a justification in the test report for why the mechanism cannot be tested.
- Test 4: For each mechanism that can be configured or turned on or off, the evaluator shall perform tests to ensure that the configuration of the mechanism behaves as specified in the operational guidance.

Same Origin Policy

FDP_SOP_EXT.1 Extended: Same Origin Policy

FDP_SOP_EXT.1.1 The TOE shall ensure that content retrieved from one origin cannot interact with content retrieved from another origin without consent from the origin whose content is being retrieved.

FDP_SOP_EXT.1.2 The TOE shall not allow same origin policy exceptions for different domains.

Application Note:

The Same Origin Policy concept is described in RFC 6454, "The Web Origin Concept."

Origin is defined as the combination of domain, protocol and port. Two URIs sharing the same domain, protocol and port are considered to have the same origin.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure it describes its implementation of a same origin policy and explains how it complies with RFC 6454. If the TSF allows the relaxation of the same origin policy for subdomains in different tabs or windows, the TSS shall describe how these exceptions are implemented.

Guidance

N/A

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall obtain or create scripts that can retrieve content from designated locations and shall set up a web server with two or more web pages representing different domains. The evaluator shall incorporate the scripts into the web pages. The evaluator shall associate each page with a different protocol and/or port. The evaluator shall open two or more browser windows and navigate to a different page on the website in each window. The evaluator shall run the scripts and shall verify that the

script that is running in one window cannot access content that was retrieved in a different window.

- Test 2: Using the same web server, the evaluator shall open two or more browser tabs and navigate to a different page on the website in each tab. The evaluator shall run the scripts and shall verify that the script that is running in one tab cannot access content that was retrieved in a different tab.
- Test 3: If the TSF supports relaxation of the same origin policy for subdomains, the evaluator shall configure the web server with subdomains and shall repeat tests 1 and 2. The evaluator shall verify that the scripts can retrieve content from another window/tab at a different subdomain.

Secure Data Transmission
FDP_STR_EXT.1 Extended: Secure Transmission of Cookie Data

FDP_STR_EXT.1.1 The TOE shall ensure that cookies containing the “secure” attribute in the set-cookie header are sent over HTTPS.

Application Note:

The set-cookie header functionality is described in RFC 6265, “HTTP State Management Mechanism.”

Assurance Activity:

TSS

The evaluator shall examine the TSS to verify it describes the TOE’s support for the “secure” attribute of the set-cookie header in accordance with RFC 6265, including the required sending of cookies containing this attribute over HTTPS.

Guidance

N/A

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall connect the TOE to a cookie-enabled test website implementing HTTPS and have the website present the TOE with a “secure” cookie. The evaluator shall examine the TOE’s cookie cache and verify that that it contains the secure cookie.
- Test 2: The evaluator shall reconnect to the cookie-enabled website over an insecure channel and verify that no “secure” cookie is sent.

User Tracking Information
FDP_TRK_EXT.1 Extended: Tracking Information Collection

FDP_TRK_EXT.1.1 The TOE shall support the ability of websites to collect [selection: web sites accessed, geo-location information, system configuration, system status, error conditions, crash conditions, [assignment: *other tracking information*]] pertaining to the TOE user.

FDP_TRK_EXT.1.2 The TOE shall provide notification to the user when tracking information is requested by a website.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure it describes the TSF's support for tracking information and specifies the tracking information that the TOE allows websites to collect about the TOE user.

Guidance

The evaluator shall examine the operational guidance to ensure it describes any notifications that the user will receive when tracking information is requested by a website and the options that the user has upon receiving the notification.

Tests

The evaluator shall perform the following tests for each type of tracking information listed in the TSS:

- Test 1: The evaluator shall configure a website that requests the tracking information about the user and shall navigate to that website. The evaluator shall verify that the user is notified about the request for tracking information and that, upon consent, the web site retrieves the tracking information.

4.2.2 Class: Protection of the TSF (FPT)

Downloads to the TOE

FPT_DNL_EXT.1 Extended: Launch of Downloaded Executables

FPT_DNL_EXT.1.1 The TOE shall prevent downloaded executables from launching automatically.

FPT_DNL_EXT.1.2 The TOE shall present the user with the option to either download or discard the executable.

Application Note:

In this context, an executable is a file containing compiled code for a software program that, when launched, initiates the installation of the program. It is invoked independent of and outside the context of the TOE. It does not include mobile code, scripts, or plug-ins.

This requirement does not include the invocation of any program that is already installed on the TOE platform.

This requirement ensures that if the user intentionally (via clicking on a link) or unintentionally initiates the download of an executable, the TSF will intervene by, for example, opening a dialog box that presents the user with the option to either save the executable to the file system or not download the executable.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure that it describes the behavior of the TSF when a user initiates the download of an executable.

Guidance

The evaluator shall examine the operational guidance to ensure it describes the dialog box that appears when a download is initiated and the implications of the options presented by the dialog box.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall navigate to a website that hosts executables and shall attempt to download and launch several executables. The evaluator shall verify that the TOE always presents a dialog box with the option to either download the executable to the file system or discard the executable.

FPT_DNL_EXT.2 Extended: Download Location

FPT_DNL_EXT.2.1 The TOE shall have the capability to specify where downloads are saved.

Assurance Activity:

TSS

N/A

Guidance

The evaluator shall examine the operational guidance to ensure it describes the default location for downloads and instructions for modifying the location.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall navigate to a website that serves files and shall to download a file. The evaluator shall inspect the default location specified in the operational guidance and shall verify that the download is present.
- Test 2: The evaluator shall modify the location where downloaded files are saved. The evaluator shall attempt to download a file from a website that serves files and shall verify that the file is saved to the configured location.

External Interactions
FPT_INT_EXT.1 Extended: Interactions with Background Processes

FPT_INT_EXT.1.1 The TOE shall shut down any TOE-spawned background processes when the TOE exits.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure it describes all TOE-spawned processes that run while the TOE is running and describes the ability to terminate these processes when the TOE exits. The TSS should also describe the nature of the background processes, the circumstances under which they are spawned, the process/image names associated with them (if any) and their expected lifetimes.

Guidance

N/A

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall start an instance of the TOE. The evaluator shall inspect the TOE platform's running processes and identify which are associated with the TOE instance. The evaluator shall shut down the TOE instance and verify that the TOE background processes have been terminated.

Mobile Code
FPT_MCD_EXT.1 Extended: Mobile Code

FPT_MCD_EXT.1.1 The TOE shall support the capability to execute signed [selection: ActiveX, Flash, Java, JavaScript, [assignment: *other mobile code types supported by the TOE*]] mobile code.

FPT_MCD_EXT.1.2 The TOE shall support the capability to execute unsigned [selection: ActiveX, Flash, Java, JavaScript, [assignment: *other mobile code types supported by the TOE*]] mobile code.

FPT_MCD_EXT.1.3 The TOE shall support the capability to execute [selection: ActiveX, Flash, Java, JavaScript, [assignment: *other mobile code types supported by the TOE*]] mobile code from an untrusted or unverified source.

FPT_MCD_EXT.1.3 The TOE shall notify the user when unsigned, untrusted or unverified mobile code is encountered.

FPT_MCD_EXT.1.4 The TOE shall provide the user with the option to discard unsigned, untrusted or unverified mobile code without executing it.

Application Note:

The ST writer must specify all mobile code types that are supported by the browser. If they are not listed, the ST writer must fill in the assignment with the mobile code types that are missing.

An authorized signer may directly sign the code itself, or the code may be delivered over an authenticated HTTPS connection with an authorized entity.

The execution of signed mobile code is specified by FIA_X509_EXT.2.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure it lists the types of signed mobile code that the TSF supports. The TSS shall describe how the TSF handles unsigned mobile code, mobile code from an untrusted source, and mobile code from an unverified source.

Guidance

The evaluator shall examine the operational guidance to verify it provides configuration instructions for each of the supported mobile code types. The operational guidance shall also describe the alert that the TOE displays to the user when unsigned, untrusted, or unverified mobile code is encountered and the actions the user can take.

Tests

The evaluator shall perform the following tests for each mobile code type specified in the TSS:

- Test 1: The evaluator shall construct web pages containing unsigned, correctly authenticated, and incorrectly authenticated mobile code and ensure that the TOE alerts the user when it encounters mobile code that fails to authenticate and provides the user with the option to discard the mobile code without executing it, but does execute mobile code that properly authenticates.

4.3 TOE or TOE Platform Security Functional requirements

This section addresses Security Functional Requirements that may be met by either the TOE itself, by the TOE platform, or by a combination of the TOE and the TOE platform.

4.3.1 Class: Cryptographic Support (FCS)

Cryptographic Key Management
FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1(1) Cryptographic Key Generation (for key establishment)

FCS_CKM.1.1(1) Refinement: The [selection: TOE, TOE platform] shall generate asymmetric cryptographic keys **used for key establishment** in accordance with

- NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" for elliptic curve-based key establishment schemes and implementing "NIST curves" P-256, P-384 and [selection: P-521, no other curves] (as defined in FIPS PUB 186-4, "Digital Signature Standard")
- NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography" for RSA-based key establishment schemes

[selection:

- NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" for finite field-based key establishment schemes;
- No other schemes

]

and specified cryptographic key sizes equivalent to, or greater than, a symmetric key strength of 112 bits.

Application Note:

This component requires that the TSF or the TOE platform be able to generate the public/private key pairs that are used for key establishment purposes for the various cryptographic protocols used by the TOE (e.g., trusted channel). If multiple schemes are supported, then the ST author should iterate this requirement to capture this capability. The scheme used will be chosen by the ST author from the selection. Since the domain parameters to be used are specified by the requirements of the protocol in this PP, it is not expected that the TOE will generate domain parameters, and therefore there is no additional domain parameter validation needed when the TOE complies with the protocols specified in this PP.

The generated key strength of 2048-bit DSA and RSA keys need to be equivalent to, or greater than, a symmetric key strength of 112 bits. See NIST Special Publication 800-57, "Recommendation for Key Management" for information about equivalent key strengths.

RSA and elliptic curve-based schemes are required in order to comply with the required ciphersuites in FCS_TLSC_EXT.1.

Assurance Activity:

TSS

Requirement met by the TOE platform: The evaluator shall examine the ST of the platform to ensure that the key establishment claimed in that platform's ST contains the key establishment requirement in the Web Browser's ST. The evaluator shall also examine the TSS of the Web Browser's ST to verify that it describes (for each supported platform) how the key establishment functionality is invoked (it should be noted that this may be through a mechanism that is not implemented by the Web Browser; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Guidance

N/A

Tests

Requirement met by the TOE:

The evaluator shall verify the implementation of the key generation and key establishment schemes used on the TOE:

Key Generation:

The evaluator shall verify the implementation of the key generation routines of the supported schemes using the applicable tests below.

Key Generation for RSA-Based Key Establishment Schemes:

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d.

Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

- Random Primes:
    - Provable primes
    - Probable primes
- Primes with Conditions:
    - Primes p1, p2, q1,q2, p and q shall all be provable primes
    - Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes
    - Primes p1, p2, q1,q2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator shall seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the

correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

Key Generation for Finite-Field Cryptography (FFC) – Based 56A Schemes

FFC Domain Parameter and Key Generation Tests

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

- Cryptographic and Field Primes:
    - Primes q and p shall both be provable primes
    - Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

- Cryptographic Group Generator:
    - Generator g constructed through a verifiable process
    - Generator g constructed through an unverifiable process.

The Key generation specifies 2 ways to generate the private key x:

- Private Key:
    - len(q) bit output of RBG where 1 <=x <= q-1
    - len(q) + 64 bit output of RBG, followed by a mod q-1 operation where 1<= x<=q-1.

The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- g != 0,1
- q divides p-1
- g^q mod p = 1
- g^x mod p = y

for each FFC parameter set and key pair.

Key Generation for Elliptic Curve Cryptography (ECC) - Based 56A Schemes

ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-284 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

ECC Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-284 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Establishment Schemes

The evaluator shall verify the implementation of the key establishment schemes of the supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role-key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE’s public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and TOE id fields.

If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

SP800-56 Key Establishment Schemes

At this time, detailed test procedures for RSA-based key establishment schemes are not available. In order to show that the TSF complies with 800-56A and/or 800-56B, depending on the selections made, the evaluator shall ensure that the TSS lists all sections of the appropriate 800-56 standard(s) to which the TOE complies.

FCS_CKM_EXT.1 Extended: Cryptographic Storage

FCS_CKM_EXT.1.1 The [selection: TOE, TOE platform] shall store persistent secrets, private keys, [assignment: *sensitive web-form data*], and secure cookies when not in use in platform-provided key storage.

Application Note:

This requirement ensures that persistent secrets (passwords, other credentials, secret keys), private keys, sensitive web-form data, and secure cookies are stored securely when not in use.

Sensitive web-form data may include personally identifiable information (e.g. social security numbers, addresses, date of birth) and financial information (e.g. credit card numbers, bank account numbers). The ST author will specify in the TSS what web-form data the TOE supports and how it is protected.

Secure cookies are cookies which have the "secure" attribute in the "set-cookie" header.

If any of the above are manipulated by the TOE and others are manipulated by the platform, then both of the selections can be specified by the ST author and the ST author must identify in the TSS those keys which are manipulated by the TOE and those by the platform.

This requirement mandates persistent secrets, private keys, sensitive web-form data, and secure cookies used by the Web Browser will be stored by the platform.

Assurance Activity:

TSS

Regardless of whether this requirement is met by the TOE or the TOE platform, the evaluator will examine the TSS to ensure that it lists each persistent secret (password, credential, or secret key) and private key needed to meet the requirements in the ST. If any of the above are manipulated by the TOE and others are manipulated by the platform, the evaluator shall verify that the TSS identifies which keys are manipulated by the TOE and which by the platform. The evaluator shall verify that the TSS identifies which web-form data is stored, which is treated as sensitive, and which is protected. The evaluator shall verify that the TSS identifies how cookies are identified as secure. For each of these items, the evaluator will confirm that the TSS lists how the item is identified, for what purpose it is used, and how it is stored. The evaluator then performs the following actions.

Persistent secrets and private keys manipulated by the TOE platform: For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the persistent secrets, private keys, sensitive web-form data, and secure cookies listed as being stored by the platform in the Web Browser ST are identified as being protected in that platform's ST.

Persistent secrets and private keys manipulated by the TOE: The evaluator shall examine the TSS for the TOE to determine that it makes a case that, for each item listed as being manipulated by the TOE, it is not written unencrypted to persistent memory, and that the item is stored by the platform.

Guidance

N/A

Tests

N/A

FCS_CKM_EXT.4 Extended: Cryptographic Key Zeroization

FCS_CKM_EXT.4.1 The [selection: TOE, TOE platform] shall zeroize all plaintext secret and private cryptographic keys and CSPs when no longer required.

Application Note:

The ST author should select the platform if the Web Browser performs no operations using plaintext secret, private cryptographic keys, and CSPs.

Any security related information (such as keys, authentication data, and passwords) must be zeroized when no longer in use to prevent the disclosure or modification of security critical data.

The zeroization indicated above applies to each intermediate storage area for plaintext key and Cryptographic Service Provider (CSP) (i.e., any storage, such as memory buffers, that is included in the path of such data) upon the transfer of the key/CSP to another location.

Assurance Activity:

TSS

Requirement met by TOE platform: The evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption), private keys, and CSPs used to generate key that are not otherwise covered by the FCS_CKM_EXT.4 requirement levied on the TOE. For each platform listed in the ST, the evaluator shall examine the TSS of the ST of the platform to ensure that each of the secret keys, private keys, and CSPs used to generate key listed above are covered.

Requirement met by TOE: The evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption) and private keys, as well as the CSPs used to generate key; when the keys are zeroized (for example, immediately after use, on system shutdown, etc.); and the type of zeroization procedure that is performed (overwrite with zeros, overwrite three times with random pattern, etc.). If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the zeroization procedure in terms of the memory in which the data are stored (for example, "secret keys stored in volatile memory are zeroized by overwriting once with zeros, while secret keys stored on the internal hard drive are zeroized by overwriting three times with a random pattern that is changed before each write"). If a read-back is done to verify the zeroization, this shall be described as well.

Guidance

N/A

Tests

Assurance Activity Note*: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer platforms.*

Requirement met by TOE: For each key clearing situation described in the TSS the evaluator shall repeat the following test.

- Test 1: The evaluator shall utilize appropriate combinations of specialized operational environment and development tools (debuggers, simulators, etc.) for the TOE and instrumented TOE builds to test that keys are cleared correctly, including all intermediate copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.

  Cryptographic TOE implementations in software shall be loaded and exercised under a debugger to perform such tests. The evaluator shall perform the following test for each key subject to clearing, including intermediate copies of keys that are persisted encrypted by the TOE:

  - Load the instrumented TOE build in a debugger.
  - Record the value of the key in the TOE subject to clearing.
  - Cause the TOE to perform a normal cryptographic processing with the key from #1.
  - Cause the TOE to clear the key.
  - Cause the TOE to stop the execution but not exit.
  - Cause the TOE to dump the entire memory footprint of the TOE into a binary file.
  - Search the content of the binary file created in #4 for instances of the known key value from #1.

  The test succeeds if no copies of the key from #1 are found in step #7 above and fails otherwise.

  The evaluator shall perform this test on all keys, including those that persist in encrypted form, to ensure intermediate copies are cleared.

- Test 2: In cases where the TOE is implemented in firmware and operates in a limited operating environment that does not allow the use of debuggers, the evaluator shall utilize a simulator for the TOE on a general purpose operating system. The evaluator shall provide a rationale explaining the instrumentation of the simulated test environment and justifying the obtained test results.

Cryptographic Operation

FCS_COP.1(1) Cryptographic Operation (for encryption/decryption)

FCS_COP.1.1(1) The [selection: TOE, TOE platform] shall perform *[encryption/decryption]* in accordance with a specified cryptographic algorithm

- AES-CBC (as defined in NIST SP 800-38A) mode;

[selection:

- AES-GCM (as defined in NIST SP 800-38D)
- no other modes]

and cryptographic key sizes 128-bit, 256-bit.

Application Note:

128 and 256 bit key sizes are mandated in order to comply with FCS_TLSC_EXT.1

Assurance Activity:

TSS

Requirement met by the TOE platform: For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the encryption/decryption function(s) claimed in that platform's ST contains the encryption/decryption function(s) in the Web Browser's ST. The evaluator shall also examine the TSS of the Web Browser's ST to verify that it describes (for each supported platform) how the encryption/decryption functionality is invoked for each mode and key size selected in the Web Browser's ST (it should be noted that this may be through a mechanism that is not implemented by the Web Browser; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Guidance

N/A

Tests

Requirement met by the TOE:

AES-CBC Tests

AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall

have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N].

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
    if i == 1:
        CT[1] = AES-CBC-Encrypt(Key, IV, PT)
        PT = IV
```

else:

CT[i] = AES-CBC-Encrypt(Key, PT)

PT = CT[i-1]

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Monte Carlo Tests

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

- 128 bit and 256 bit keys
- Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.
- Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.
- Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

FCS_COP.1(2) Cryptographic Operation (for cryptographic hashing)

FCS_COP.1.1(2) The [selection: TOE, TOE platform] shall perform cryptographic hashing in accordance with a specified cryptographic algorithm:

- SHA-1;
- SHA-256;

- SHA-384;
- [selection: SHA-512, no other algorithms]

and message digest sizes 160, 256, 384 and [selection: 512, no other sizes] that meet the following: FIPS PUB 180-4.

Application Note:

In future versions of this document, SHA-1 may be removed as an option. SHA-1 for generating digital signatures is no longer allowed, and SHA-1 for verification of digital signatures is strongly discouraged as there may be risk in accepting these signatures. SHA-1 is currently required in order to comply with FCS_TLSC_EXT.1.SHA-256 and SHA-384 are mandated in order to comply with FCS_TLSC_EXT.1

The intent of this requirement is to specify the hashing function used for digital signature generation and verification associated with trusted updates and trusted channel. The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(1).

Assurance Activity:

TSS

Requirement met by the platform: The intent of this requirement is to specify the hashing function used for digital signature generation and verification associated with trusted updates and trusted channel. The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(1).

Requirement met by the TOE: The evaluator shall check the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required hash sizes is present. The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

Guidance

N/A

Tests

Requirement met by the TOE:

The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs.

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

Short Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where 1 ≤ i ≤ m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 ≤ i ≤ m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Test

This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

FCS_COP.1(3) Cryptographic Operation (for digital signature)

FCS_COP.1.1(3) The [selection: TOE, TOE platform] shall perform [cryptographic signature services in accordance with the following specified cryptographic algorithms

- RSA Digital Signature Algorithm (rDSA) with a key size (modulus) of 2048 bits or greater that meets FIPS PUB 180-2 or FIPS-PUB 186-4, “Digital Signature Standard”,
- Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater that meets FIPS PUB 186-4, “Digital Signature Standard” with “NIST curves” P-256, P-

384 and [selection: P-521, no other curves] (as defined in FIPS PUB 186-4, “Digital Signature Standard”),

[selection:

- Digital Signature Algorithm (DSA) with a key size (modulus) of 2048 bits or greater, that meets FIPS-PUB 186-4, “Digital Signature Standard”;
- No other cryptographic signature service

].

Application Note:

The TOE must perform RSA and ECDSA digital signatures in accordance with FCS_TLSC_EXT.1. The TOE may also verify signatures on plug-ins and extensions.

If multiple schemes are supported, then the ST author should iterate this requirement to capture this capability. The scheme used will be chosen by the ST author from the selection.

Assurance Activity:

TSS

Requirement met by the TOE platform: For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the digital signature functions claimed in that platform's ST contains the digital signature functions in the Web Browser's ST. The evaluator shall also examine the TSS of the Web Browser’s ST to verify that it describes (for each supported platform) how the digital signature functionality is invoked for each operation they are used for in the Web Browser (it should be noted that this may be through a mechanism that is not implemented by the Web Browser; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Guidance

N/A

Tests

Requirement met by the TOE:

Key Generation:

Key Generation for RSA Signature Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d.

Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

- Random Primes:
    - Provable primes

    - Probable primes
- Primes with Conditions:
    - Primes p1, p2, q1,q2, p and q shall all be provable primes
    - Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes
    - Primes p1, p2, q1,q2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

ECDSA Key Generation Tests

FIPS 186-4 ECDSA Key Generation Test

For each supported NIST curve, i.e., P-256, P-284 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-284 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-284 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

ECDSA FIPS 186-4 Signature Verification Test

For each supported NIST curve (i.e., P-256, P-284 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages.

The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Signature Verification Test

The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

FCS_COP.1(4) Cryptographic operation (Keyed-Hash Message Authentication)

FCS_COP.1.1(4) The [selection: TOE, TOE platform] shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm

- HMAC- SHA-256

[selection:

- SHA-1
- SHA-384
- SHA-512
- no other algorithms

],

key sizes [assignment: *key size (in bits) used in HMAC*], and message digest sizes 256 and [selection: 160, 384, 512, no other size] bits that meet the following: FIPS Pub 198-1, "The Keyed-Hash Message Authentication Code, and FIPS Pub 180-4, "Secure Hash Standard."

Application Note:

The intent of this requirement is to specify the keyed-hash message authentication function used when used for key establishment purposes for the various cryptographic protocols used by the TOE (e.g., trusted channel). The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(1). HMAC-SHA256 is required in order to comply with the required ciphersuites in FCS_TLSC_EXT.1.

Assurance Activity:

Requirement met by the platform: For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the keyed-hash function(s) claimed in that platform's ST contains the keyed-hash function(s) in the Web Browser's ST. The evaluator shall also examine the TSS of the Web Browser's ST to verify that it describes (for each supported platform) how the keyed-hash functionality is invoked for each mode and key size selected in the Web Browser's ST (it should be noted that this may be through a mechanism that is not implemented by the Web Browser; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the TOE: The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

Guidance

N/A

Tests

Requirement met by the TOE:

For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and IV using a known good implementation.

Hypertext Transport Protocol Secure (HTTPS)
FCS_HTTPS_EXT.1 Extended: HTTPS Implementation

FCS_HTTPS_EXT.1.1 The [selection: TOE, TOE platform] shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2 The [selection: TOE, TOE platform] shall implement HTTPS using TLS as specified in FCS_TLSC_EXT.1.

Assurance Activity:

The evaluator shall attempt to establish an HTTPS connection with a webserver, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS. All other tests are performed in conjunction with FCS_TLSC_EXT.1.

Random Bit Generation

FCS_RBG_EXT.1 Extended: Random Bit Generation

FCS_RBG_EXT.1.1 The [selection: TOE, TOE platform] shall perform all deterministic random bit generation services in accordance with [selection: choose one of: NIST Special Publication

800-90A using [selection: Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES), Dual_EC_DRBG (any)]; FIPS Pub 140-2 Annex C: X9.31 Appendix 2.4 using AES].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [selection: a software-based noise source, a platform-based RBG] with a minimum of [selection: 128 bits, 256 bits] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

Application Note:

For the first selection in FCS_RBG_EXT.1.1, the ST author should select whether the TOE or the platform on which the TOE is installed provides the RBG services.

NIST Special Pub 800-90B, Appendix C describes the minimum entropy measurement that will probably be required future versions of FIPS-140. If possible this should be used immediately and will be required in future versions of this PP.

For the second selection in FCS_RBG_EXT.1.1, the ST author should select the standard to which the RBG services comply (either 800-90 or 140-2 Annex C).

SP 800-90A contains four different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used (if 800-90A is selected), and include the specific underlying cryptographic primitives used in the requirement or in the TSS. While any of the identified hash functions are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed. While any of the curves defined in 800-90A are allowed for Dual_EC_DRBG, the ST author not only must include the curve chosen, but also the hash algorithm used.

For the first selection in FCS_RBG_EXT.1.2, the ST author indicates whether the sources of entropy are software-based or platform-based, or both. If there are multiple sources of entropy, the ST will describe each entropy source and whether it is software or platform-based. Platform-based noise sources are preferred.

The platform-based RBG source is the output of a validated RBG provided by the platform, which is used as an entropy source for a TSF-provided DRBG according to FCS_RBG_EXT.1.1. In this way, the developer has chained RBGs as described in NIST SP800-90C.

Note that for FIPS Pub 140-2 Annex C, currently only the method described in NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section 3 is valid. If the key length for the AES implementation used here is different than that used to encrypt the user data, then FCS_COP.1 may have to be adjusted or iterated to reflect the different key length. For the selection in FCS_RBG_EXT.1.2(1), the ST author selects the minimum number of bits of entropy that is used to seed the RBG.

The ST author also ensures that any underlying functions are included in the baseline requirements for the TOE.

Assurance Activity:

TSS

Requirement met by the platform: For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the RBG functions claimed in that platform's ST contains the RBG functions in the Web Browser's ST. The evaluator shall also examine the TSS of the Web Browser's ST to verify that it describes (for each supported platform) how the RBG functionality is invoked for each operation they are used for in the Web Browser (it should be noted that this may be through a mechanism that is not implemented by the Web Browser; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the TOE: Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Annex E.

If the ST author has selected a platform-based noise source, the evaluator shall verify that the platform's RBG has been validated by examining the platform's ST. The evaluator shall verify that the platform's RBG is seeded with at least the amount of entropy selected by the ST author for this profile. In this case, the ST author is not responsible for Annex E documentation of the platform's RBG.

Guidance

N/A

Tests

Requirement met by the TOE:

The evaluator shall perform the following tests, depending on the standard to which the RBG conforms.

Implementations Conforming to FIPS 140-2, Annex C

The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluator shall conduct the following two tests. Note that the "expected values" are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

The evaluator shall perform a Variable Seed Test. The evaluator shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluator shall also provide a key (of the length appropriate to the AES algorithm) that is constant for all 128 (Seed, DT) pairs. The DT value is incremented by 1 for each set. The seed values shall have no repeats within the set. The evaluator ensures that the values returned by the TSF match the expected values.

The evaluator shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluator shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluator then invokes the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-

Key Triple DES and AES Algorithms, Section 3. The evaluator ensures that the 10,000th value produced matches the expected value.

Implementations Conforming to NIST Special Publication 800-90A

The evaluator shall perform 15 trials for the RBG implementation. If the RBG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RBG functionality.

If the RBG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. “generate one block of random bits” means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90A).

If the RBG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

Entropy input: the length of the entropy input value must equal the seed length.

Nonce: If a nonce is supported (CTR_DRBG with no df does not use a nonce), the nonce bit length is one-half the seed length.

Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

Transport Layer Security

FCS_TLSC_EXT.1 Extended: TLS

FCS_TLSC_EXT.1.1 The [selection: TOE, TOE platform] shall implement TLS 1.2 (RFC 5246) supporting the following ciphersuites:

Mandatory Ciphersuites:

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289
-

Optional Ciphersuites:
[selection:

- TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
- TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
- TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- no other ciphersuite

].

Application Note:

The ciphersuites to be used in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported; if there are no ciphersuites supported other than the mandatory suites, then "None" should be selected. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS_RSA_WITH_AES_128_CBC_SHA is required in order to ensure compliance with RFC 5246.

FCS_TLSC_EXT.1.2 The TOE shall not establish a trusted channel if the distinguished name (DN) contained in a certificate does not match the expected DN for the peer.

Application Note:

The DN may be in the Subject Name field or the Subject Alternative Name extension of the certificate. The expected DN may either be configured or may be compared to the Domain Name or IP address used by the peer.

Trusted communication channels include any of TLS, HTTPS, or DTLS performed by the TSF or TOE platform. Validity checking to establish the trusted channel is performed in conjunction with FIA_X509_EXT.1.

FCS_TLSC_EXT.1.3 The TOE shall present the signature_algorithm extension in the Client Hello with the following hash algorithms: [selection: SHA256, SHA384, SHA512] and no other hash algorithms.

Application Note:

This requirement limits the hashing algorithms supported for the purpose of digital signature verification by the client and limits the server to the supported hashes for the purpose of digital signature generation by the server. The signature_algorithm extension is only supported by TLS 1.2.

FCS_TLSC_EXT.1.4 The TOE shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [selection: secp256r1, secp384r1, secp521r1] and no other curves.

Application Note:

This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS_COP.1(3) and FCS_CKM.1(1). This extension is required for clients supporting Elliptic Curve ciphersuites.

Assurance Activity:

TSS

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component. The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

The evaluator shall verify that the TSS describes how the DN in the certificate is compared to the expected DN. The evaluator shall verify that, for HTTPS connections, the expected DN is the Domain Name or IP address of the peer, that the comparison is performed automatically, and that the TSS describes how wildcards in the DN are used.

The evaluator shall verify that the TSS describes the signature_algorithm extension and whether the required behavior is performed by default or may be configured.

The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behavior is performed by default or may be configured.

Guidance

If the DN is not compared automatically to the Domain Name or IP address, the evaluator shall ensure that the AGD guidance includes configuration of the expected DN for the connection.

If the TSS indicates that the signature_algorithm extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the signature_algorithm extension.

If the TSS indicates that the Supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves Extension.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).
- Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.
- Test 3: The evaluator shall attempt a connection with a certificate where the DN matches either the configured expected DN or the Domain Name/IP address of the peer. The evaluator shall verify that the TSF is able to successfully connect. The evaluator shall attempt a connection with a certificate where the DN does not match either the configured expected DN or the Domain Name/IP address of the peer. The evaluator shall verify that the TSF is not able to successfully connect. A user notification indicating the failure of the connection is acceptable in accordance with FIA_X509_EXT.2.3.
- Test 4: The evaluator shall configure the server to send a certificate in the TLS connection that is not supported according to the Client's signature_algorithm extension (for example, send a certificate with a SHA-1 signature). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.
- Test 5: The evaluator shall configure the server to perform an ECDHE key exchange in the TLS connection using a non-supported curve (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.
- Test 6: The evaluator shall configure the server to send a certificate in the TLS connection that the does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA certificate while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.
- Test 7: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection.

- Test 8: The evaluator shall setup a man-in-the-middle tool between the TOE and the server and shall perform the following modifications to the traffic:
    - Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
    - Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.
    - Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
    - Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange.
    - Configure the server to require mutual authentication and then modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field must not be the CA used to sign the client's certificate. The evaluator shall verify that the server rejects the connection after receiving the Client Finished handshake message.
    - Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
    - Send an unencrypted packet from the Server after the client has issued the ChangeCipherSpec message and verify that the client denies the connection.

4.3.2 Class: Identification and Authentication (FIA)

X509 Certificates
FIA_X509_EXT.1 Extended: X509 Validation

FIA_X509_EXT.1.1 The [selection: TOE, TOE platform] shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the cA flag is set to TRUE for all CA certificates.
- The TSF shall validate the revocation status of the certificate using [selection: the Online Certificate Status Protocol (OCSP) as specified in RFC 2560, a Certificate Revocation List (CRL) as specified in RFC 5759].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
    - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3).

- Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.

Application Note:

FIA_X509_EXT.1.1 lists the rules for validating certificates. The ST author shall select whether revocation status is verified using OCSP or CRLs. Certificates may optionally be used for trusted updates of the TOE (FPT_TUD_EXT.1.3) and installation of plug-ins and extensions (FPT_TUD_EXT.2.3 and FPT_TUD_EXT.3.3) and, if implemented by the TOE, must be validated to contain the Code Signing purpose extendedKeyUsage. Certificates must be used to perform authentication with Web Servers using FCS_TLSC_EXT.1 and must be validated to contain the Server Authentication purpose extendedKeyUsage.

Regardless of the selection of TSF or TOE platform, the validation is expected to end in a trusted root certificate in a root store managed by the platform.

While FIA_X509_EXT.1.1 requires that the TOE perform certain checks on the certificate presented by a TLS server, there are corresponding checks that the server will have to perform on a certificate presented by the client; namely that the extendedKeyUsage field of the client certificate includes "Client Authentication" and that the key agreement bit (for the Diffie-Hellman ciphersuites) or the key encipherment bit (for RSA ciphersuites) be set. Certificates obtained for use by the TOE will have to conform to these requirements in order to be used in the enterprise.

FIA_X509_EXT.1.2 The [selection: TOE, TOE platform] shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

Application Note:

This requirement applies to certificates that are used and processed by the TOE or TOE platform.

Assurance Activity:

TSS

The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

Guidance

N/A

Tests

The tests described must be performed in conjunction with the other Certificate Services Assurance Activity, including the use cases in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules.

- Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function (application validation, trusted channel setup, or trusted software update) failing. The evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.
- Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.
- Test 3: The evaluator shall test that the TOE can properly handle revoked certificates – conditional on whether CRL or OCSP is selected; if both are selected, and then a test is performed for each method. The evaluator has to only test one up in the trust chain (future revisions may require to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that will be revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.
- Test 4: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.
- Test 5: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension not set. The validation of the certificate path fails.
- Test 6: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.
- Test 7: The evaluator shall modify a single byte in the middle of the certificate and demonstrate that the certificate fails to validate.

FIA_X509_EXT.2 Extended: X509 Authentication

FIA_X509_EXT.2.1 The [selection: TOE, TOE platform] shall use X.509v3 certificates as defined by RFC 5280 to support authentication for HTTPS, TLS, and [selection: DTLS, no other protocol], and code signing for TOE updates, code signing for mobile code installation, and [selection: code signing for extension installation, code signing for plug-in installation, no additional uses].

Application Note:

DTLS shall be selected if FCS_DTLS_EXT.1 is included in the main body. Certificates must be used for trusted updates of TOE software and may optionally be used for installation of plug-ins and extensions.

Code signing for TOE updates and mobile code must adhere to the rules specified in FPT_TUD_EXT.1.1 and FPT_MCD_EXT.1, respectively. If selected, code signing for extensions and plug-ins must adhere to the rules specified in FPT_TUD_EXT.2 and FPT_TUD_EXT.3, respectively.

FIA_X509_EXT.2.2 When the [selection: TOE, TOE platform] cannot establish a connection to determine the validity of a certificate, the [selection: TOE, TOE platform] shall [selection: allow the administrator to choose whether to accept the certificate in these cases, accept the certificate, not accept the certificate].

Application Note:

Often a connection must be established to perform a verification of the revocation status of a certificate - either to download a CRL or to perform OCSP. The selection is used to describe the behavior in the event that such a connection cannot be established (for example, due to a network error). If the TOE has determined the certificate valid according to all other rules in FIA_X509_EXT.1, the behavior indicated in the second selection shall determine the validity. The TOE must not accept the certificate if it fails any of the other validation rules in FIA_X509_EXT.1.

FIA_X509_EXT.2.3 The [selection: TOE, TOE platform] shall notify the user if the peer certificate is deemed invalid during trusted communication channel establishment.

Application Note:

Trusted communication channels include any of TLS, HTTPS, or DTLS performed by the TSF or TOE platform. Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.

FIA_X509_EXT.2.4 The [selection: TOE, TOE platform] shall not install code if the code signing certificate is deemed invalid.

Application Note:

Certificates may optionally be used for trusted updates of system software (FPT_TUD_EXT.1.3).

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure that it describes how the TOE chooses which certificates to use.

The evaluator shall examine the TSS to ensure that it describes the behavior of the TOE or TOE platform when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel.

Guidance

The evaluator shall verify that any necessary instructions are contained in the operational guidance for configuring the operating environment so that the TOE can use the certificates.

If the requirement is that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

Tests

The evaluator shall perform Test 1 for each function listed in FIA_X509_EXT.2.1 that requires the use of certificates:

- Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing or in the receipt of a user notification (as required by elements 3 and 4). The evaluator shall then load into the platform's root store any certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds.
- Test 2: The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the operational guidance to determine that all supported administrator-configurable options behave in their documented manner.

4.3.3 Class: Security Management (FMT)

Management of Functions in TSF
FMT_MOF.1 Management of Functions Behavior

FMT_MOF.1.1 The [selection: TOE, TOE and TOE platform] shall restrict the ability to *perform* the functions:

1. enable/disable storage of sensitive web form information;
2. configuration of mobile code:
    a. ability to install mobile code;
    b. ability to uninstall mobile code;
    c. ability to update mobile code;
    d. ability to execute unsigned mobile code;
    e. ability to execute mobile code from an untrusted or unverified publisher

[selection:

3. enable/disable storage of third party cookies;
4. configuration of extensions:
    a. ability to install extensions;
    b. ability to uninstall extensions;
    c. ability to update extensions;
    d. ability to disable extensions
5. configuration of plug-ins:
    a. ability to install plug-ins;
    b. ability to uninstall plug-ins;
    c. ability to update plug-ins;
    d. ability to disable plug-ins

6. enable/disable use of OCSP for obtaining the revocation status of an X.509 certificate;
7. configure inclusion of user-agent information in HTTP headers;
8. enable/disable ability for websites to collect tracking information about the user
9. deletion of stored browsing data (cache, web form information):
    a. ability to specify which content should be deleted;
    b. ability to delete all stored content;
    c. automatically delete content upon termination of the browser session;
10. enable/disable storage of sensitive information in persistent storage;
11. ability to specify location where downloaded files are saved to disk;
12. configure size of cookie cache;
13. enable/disable interaction with Graphic Processing Units (GPUs)
14. configure the ability to advance to a web site with an invalid or unvalidated X.509 certificate;
15. enable/disable use of private browsing sessions;
16. configure the use of an application reputation service to detect malicious applications prior to downloading them;
17. configure the use of a URL reputation service to detect sites that contain malware or phishing content;
18. enable/disable automatic installation of software updates and patches;
19. enable/disable establishment of a trusted channel if the TSF cannot establish a connection to determine the validity of a certificate;
20. enable/disable ability for websites to register protocol handlers.

]

to an administrator according to the administrator policy.

Application Note:

The intent of this requirement is to allow the administrator of the TOE platform to configure the TOE with a policy that may not be over-ridden by the user. If the administrator has not set a policy for a particular function, the user may still perform that function. Enforcement of the policy is done by the TOE itself, or the TOE and the TOE platform in coordination with each other

Assurance Activity:

TSS

The evaluator shall verify that the TSS describes those management functions which may only be configured by the TOE platform administrator and cannot be over-ridden by the user when set according to policy.

Guidance

The evaluator shall examine the operational guidance to verify that it includes instructions for a TOE platform administrator to configure the functions listed in FMT_MOF.1.1.

Tests

The evaluator shall perform the following test:

- Test 1: The evaluator shall create policies that collectively include all management functions controlled by the TOE platform administrator and cannot be over-ridden by the user as defined in FMT_MOF.1.1. The evaluator shall apply these policies to the TOE, attempt to override each setting as the user, and verify that the TSF does not permit it.

Specification of Management Functions
FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1 The [selection: TOE, TOE and TOE platform] shall be capable of performing the following management functions:

1. enable/disable storage of sensitive web form information;
2. enable/disable storage of third party cookies;
3. configure inclusion of user-agent information in HTTP headers;
4. enable/disable ability for websites to collect tracking information about the user
5. deletion of browsing data (cache, web form information):
    a. ability to specify which content should be deleted;
    b. ability to delete all stored content;
    c. automatically delete content upon termination of the browser session;
6. configuration of mobile code:
    a. ability to install mobile code;
    b. ability to uninstall mobile code;
    c. ability to update mobile code;
    d. ability to execute unsigned mobile code;
    e. ability to execute mobile code from an untrusted or unverified publisher
7. ability to specify location where downloaded files are saved to disk;
8. configure size of cookie cache;
9. enable/disable the ability to advance to a web site with an invalid or unvalidated X.509 certificate;
10. enable/disable automatic installation of software updates and patches;

[selection:

11. configuration of extensions:
    a. ability to install extensions;
    b. ability to uninstall extensions;
    c. ability to update extensions;
    d. ability to disable extensions
12. configuration of plug-ins:
    e. ability to install plug-ins;
    f. ability to uninstall plug-ins;
    g. ability to update plug-ins;
    h. ability to disable plug-ins
13. enable/disable storage of sensitive information in persistent storage;

14. enable/disable use of OCSP for obtaining the revocation status of an X.509 certificate;
15. enable/disable interaction with Graphic Processing Units (GPUs)
16. enable/disable use of private browsing sessions;
17. configure the use of an application reputation service to detect malicious applications prior to downloading them;
18. configure the use of a URL reputation service to detect sites that contain malware or phishing content;
19. enable/disable establishment of a trusted channel if the TSF cannot establish a connection to determine the validity of a certificate;
20. enable/disable ability for websites to register protocol handlers

].

Application Note

There may be some instances where an administrator configures security management functions and "pushes" configuration information down to the TOE. This is an acceptable form of management; the ST author simply must make clear in the ST what management functions are configured at the TOE, and which are configured by the administrator. It may be the case that the functions overlap (i.e., can be done by an end-user on the platform or by the administrator) and this is fine as long as the ST is clear and the guidance documentation describes how to perform the functions.

Function 1 is specified in FCS_CKM_EXT.1.

Function 2 is specified in FDP_COO_EXT.1.

Function 3 addresses configuring the information in user-agent strings as well as enabling/disabling the sending of any information in the user-agent string.

Function 4 is specified in FDP_TRK_EXT.1.

Function 5 is specified in FDP_DEL_EXT.1.

Function 6 is specified in FIA_X509_EXT.2.1 and FPT_MCD_EXT.1.

Function 7 is specified in FPT_DNL_EXT.2.1.

Function 8 addresses the ability to configure the size of the cookie cache to control the number of cookies stored by each user.

Function 9 is specified in FIA_X509_EXT.1 and FIA_X509_EXT.2.

Function 10 is specified in FPT_TUD_EXT.1.

Function 11 should be selected if extensions are selected in FDP_DEL_EXT.1 and FPT_TUD_EXT.1. Uninstallation is specified in FDP_DEL_EXT.2; installation and updating is specified in FPT_TUD_EXT.2.

Function 12 should be selected if plug-ins are selected in FDP_DEL_EXT.1 and FPT_TUD_EXT.1. Uninstallation is specified in FDP_DEL_EXT.3; installation and updating is specified in FPT_TUD_EXT.3.

Function 13 is specified in FDP_PST_EXT.1 and should be selected if the TOE supports it.

Function 14 is specified in FIA_X509_EXT.1.1 and should be selected if the TOE supports it.

For function 15, a GPU is a specialized electronic circuit that is very efficient at manipulating and rendering computer graphics. Browsers that have this capability should include this function. In the browser's UI, it may appear as an option for accelerated graphics. It should be selected if the TOE supports it.

Function 16 is specified in FDP_PBR_EXT.1 and should be selected if the TOE supports it.

Function 17 is specified in FPT_INT_EXT.2 and should be selected if the TOE supports it.

Function 18 is specified in FPT_INT_EXT.3 and should be selected if the TOE supports it.

Function 19 is specified in FIA_X509_EXT.2.2 and should be selected if the TOE supports it.

For function 20, a protocol handler is a web application that can be associated with a particular web protocol (e.g., mail, calendar). A web application can attempt to register itself with a browser as a potential handler for a given protocol. This function should be selected if the TOE supports it.

Assurance Activity

TSS

As stated in the application note, a TOE may be configured either locally or remotely. The evaluator shall examine the TSS to ensure it clearly states which functions can be performed locally and remotely.

Guidance

The evaluator shall verify that every management function mandated by the PP is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

The operational guidance documentation will describe how configuration is performed locally and/or remotely in accordance with the TSS description of each function.

Tests

The evaluator shall test the ability of the TOE, or the ability of the TOE in conjunction with the TOE platform, to provide the management functions by configuring the TOE and testing each option listed in the requirement above. The evaluator shall consult the AGD guidance to perform each of the following tests, iterating each test as necessary if both the user and administrator may perform the function.

Function 1: The test of this function is performed in conjunction with FCS_CKM_EXT.1.

Function 2: The test of this function is performed in conjunction with FDP_COO_EXT.1.

Function 3:

- Test 1: The evaluator shall configure the option to disable the user-agent string per the instructions in the operational guidance. The evaluator shall initiate a connection to a server and sniff the traffic between the TOE and the server by using a network protocol analyzer. The evaluator shall inspect the captured network traffic and verify that the user-agent string is not present in the HTTP header.
- Test 2: The evaluator shall configure the contents of the user-agent string per the instructions in the operational guidance. The evaluator shall initiate a connection to a server and sniff the traffic between the TOE and the server by using a network protocol analyzer. The evaluator shall inspect the captured network traffic and verify that the user-agent string matches the configured value.

Function 4: The evaluator shall enable the collection of tracking information and shall perform the tests associated with FDP_TRK_EXT.1. The evaluator shall disable the collection of tracking information, repeat the tests, and shall verify that no tracking information is collected.

Function 5: The test of this function is performed in conjunction with FDP_DEL_EXT.1.

Function 6: The test of this function is performed in conjunction with FIA_X509_EXT.2.1 and FPT_MCD_EXT.1.

Function 7: The test of this function is performed in conjunction with FPT_DNL_EXT.2.1.

Function 8:

- Test 1: The evaluator shall navigate to cookie-enabled websites and verify that once the default cookie cache size threshold has been exceeded, no other cookies are written to the cookie cache.
- Test 2: The evaluator shall configure the size of the cookie cache to the maximum size permissible per the operational guidance. The evaluator shall verify that, upon accessing a website that attempts to store a cookie that will exceed the maximum cookie cache size threshold, the cookie is not written to the cookie cache.

Function 9: The test of this function is performed in conjunction with FIA_X509_EXT.1 and FIA_X509_EXT.2.

Function 10: The evaluator shall enable automatic installation of updates and patches and shall perform the tests associated with FPT_TUD_EXT.1.4. The evaluator shall then disable automatic installation, rerun the tests, and shall verify that no updates or patches are installed.

Function 11: (Conditional)

- Test 1. For installation, updating and deletion of extensions, the tests shall be performed in conjunction with FPT_TUD_EXT.2 and FDP_DEL_EXT.2.
- Test 2: The evaluator shall load a TOE with a number of extensions. The evaluator shall attempt to disable a number of extensions. The evaluator shall then attempt to use the functionality of the disabled extensions and shall verify that the functionality does not

work. The evaluator shall also inspect the TOE's extension interface and shall verify that the disabled extensions appear as disabled.

Function 12: (Conditional)

- Test 1. For installation, updating and deletion of plug-ins, the tests shall be performed in conjunction with FPT_TUD_EXT.3 and FDP_DEL_EXT.3.
- Test 2: The evaluator shall load a TOE with a number of plug-ins. The evaluator shall attempt to disable a number of plug-ins. The evaluator shall then attempt to use the functionality of the disabled plug-ins and shall verify that the functionality does not work. The evaluator shall also inspect the TOE's plug-in interface and shall verify that the disabled plug-ins appear as disabled.

Function 13: Conditional) The test of this function is performed in conjunction with FDP_PST_EXT.1.

Function 14: (Conditional) The test of this function is performed in conjunction with FIA_X509_EXT.1.

Function 15: (Conditional) The evaluator shall access the TOE's UI and shall verify that the ability to enable /disable interaction with the GPU is available in the UI.

Function 16: (Conditional) The test of this function is performed in conjunction with FDP_PBR_EXT.1.

Function 17: (Conditional) The test of this function is performed in conjunction with FPT_INT_EXT.2.

Function 18: (Conditional) The test of this function is performed in conjunction with FPT_INT_EXT.3.

Function 19: (Conditional) The test of this function is performed in conjunction with FIA_X509_EXT.2.2.

Function 20: (Conditional)

- Test 1: The evaluator shall allow registration of protocol handlers. The evaluator shall navigate to a test or actual website running an application that can register itself as a protocol handler and shall verify that the application can register itself.
- Test 2: The evaluator shall remove the registration of the protocol handler in test 1 and disallow protocol handler registration. The evaluator shall navigate to the same site and shall verify that the website is not allowed to register itself.

FMT_SMR.1 Security Management Roles

FMT_SMR.1.1 The [selection: TOE, TOE platform] shall maintain the roles: *Administrator*.

FMT_SMR.1.2 The [selection: TOE, TOE platform] shall be able to associate users with roles.

Assurance Activity:

TSS

The evaluator shall examine the TSS and user documents to verify that they describe the administrator role and the powers granted to and limitations of the role.

Guidance

The evaluator shall examine the operational guidance to ensure that it contains instructions for administering the TOE and which interfaces are supported.

Tests

In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this PP be tested; for instance, if the TOE can be administered through a local hardware interface or TLS/HTTPS then both methods of administration must be exercised during the evaluation team's test activities.

4.3.4 Class: Protection of the TSF (FPT)

TSF Self-Test

FPT_TST_EXT.1 Extended: TSF Self-Test

FPT_TST_EXT.1.1 The [selection: TOE, TOE platform] shall run a suite of self-tests during initial start-up (on power on) to ensure the integrity of its executable and data.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure that it specifies the self-tests that are performed at start-up; this description should include an outline of what the tests are actually doing (e.g., verifying the integrity of the TOE executable). The TSS must include any error states that the TSF or TOE platform may enter when self-tests fail, and the conditions and actions necessary to exit the error states and resume normal operation. The evaluator shall verify that the TSS indicates these self-tests are run at start-up automatically, and do not involve any inputs from or actions by the user or operator.

Guidance

N/A

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall perform the integrity check on a known good TSF executable and verify that the check is successful.
- Test 2: The evaluator shall modify the TOE executable, performs the integrity check on the modified TSF executable and verify that the check fails.
- Test 3: The evaluator shall perform the integrity check on known good TOE data and verify that the check is successful.
- Test 4: the evaluator shall modify the TOE configuration data, perform the integrity check on the modified TOE data, and verify that the check fails.

Trusted Update
FPT_TUD_EXT.1 Extended: Trusted Software Updates and Patches

FPT_TUD_EXT.1.1 The [selection: TOE, TOE platform] shall provide the ability to query the current version of the TOE software, [selection: extension, plug-in, no other add-on].

FPT_TUD_EXT.1.2 The [selection: TOE, TOE platform] shall provide the ability to initiate updates and patches to the TOE software, [selection: extension, plug-in, no other add-on].

FPT_TUD_EXT.1.3 The [selection: TOE, TOE platform] shall provide a means to verify software updates and patches to the TOE, [selection: extension updates, plug-in updates, no other updates] using a digital signature mechanism and [selection: published hash, no other functions] prior to installing those updates and patches.

FTP_TUD_EXT.1.4 The [selection: TOE, TOE platform] shall provide the ability to install TOE updates and patches, [selection: extensions, plug-ins, no other add-on] automatically after verification.

Application Note:

The digital signature mechanism referenced in the third element is the one specified in FCS_COP.1(3). The published hash referenced is generated by one of the functions specified in FCS_COP.1(2).

If extensions or plug-ins are selected above, the applicable selection-based requirements from Annex C must also be included in the main body of the ST.

Assurance Activity:

TSS

Updates to the TOE are signed by an authorized source and may also have a hash associated with them. The definition of an authorized source must be contained in the TSS, along with a description of how the certificates used by the update verification mechanism are contained on the system. The evaluator shall ensure this information is contained in the TSS.

The evaluator shall also ensure that the TSS (or the operational guidance) describes how the candidate updates are obtained; the processing associated with verifying the digital signature or calculating the hash of the updates; and the actions that take place for successful (hash or

signature was verified) and unsuccessful (hash or signature could not be verified) cases. If these activities are performed entirely by the underlying platform, a reference to the ST of each platform indicating that the required functionality is included for each platform shall be verified by the evaluator.

Guidance

The evaluator shall examine the operational guidance to verify it documents the steps for verifying the current version of the TOE software, initiating updates and patches to the TOE software, and configuring verification of software updates and patches.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall perform the version verification activity to determine the current version of the product. The evaluator shall obtain a legitimate update using procedures described in the operational guidance and verify that it is successfully installed on the TOE. Then, the evaluator shall perform a subset of other assurance activity tests to demonstrate that the update functions as expected. After the update, the evaluator shall perform the version verification activity again to verify the version correctly corresponds to that of the update.
- Test 2: The evaluator shall perform the version verification activity to determine the current version of the product. The evaluator shall obtain or produce an illegitimate update, and attempt to install it on the TOE or TOE platform. The evaluator shall verify that the TOE or TOE platform rejects the update.
- Test 3: The evaluator shall attempt to install an unsigned patch or update and shall verify that installation fails.
- Test 4: the evaluator shall sign a patch or update with an invalid certificate. The evaluator shall attempt to install the patch or update and shall verify that installation fails.
- Test 5: The evaluator shall sign a patch or update with a certificate containing a missing or invalid code signing entendedKeyUsage extension. The evaluator shall attempt to install the patch or update and shall verify that installation fails.
- Test 6: The evaluator shall attempt to install a signed patch or update and shall verify that installation is successful and happens automatically after the signature is verified.

4.3.5 Class: Trusted Path/Channel (FTP)

Trusted Channel
FTP_ITC.1 Inter-TSF Trusted Channel

FTP_ITC.1 **Refinement**: The [selection: TOE, TOE platform] shall use **[selection: HTTPS, TLS, DTLS]** to provide a **trusted** communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured

identification of its end points and protection of the channel data **from disclosure and detection of modification of the channel data**.

FPT_ITC.1.2 The [selection: TOE, TOE platform] shall permit *the TSF* to initiate communication via the trusted channel.

FPT_ITC.1.3 The [selection: TOE, TOE platform] shall initiate communication via the trusted channel for *all traffic traversing that connection*.

Application Note:

The intent of the above requirement is to use the cryptographic protocols identified in the requirement to establish and maintain a trusted channel between the TOE and a trusted server.

Assurance Activity:

TSS

The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to a web server in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specification. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.

Guidance

The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to the access point, and that it contains recovery instructions should a connection be unintentionally broken.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall ensure that the TOE or TOE platform is able to initiate communications with a web server using the protocols specified in the requirement, setting up the connections as described in the operational guidance and ensuring that communication is successful.
- Test 2: The evaluator shall ensure, for each communication channel with a web server, the channel data is not sent in plaintext.

5 Security Assurance Requirements

The Security Objectives for the TOE in Section 3 were constructed to address threats identified in Section 2. The Security Functional Requirements (SFRs) in Section 4 are a formal instantiation of the Security Objectives. The PP draws from the CC Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

While this section contains the complete set of SARs from the CC, the Assurance Activity to be performed by an evaluator are detailed both in Section 4 as well as in this section.

The general model for evaluation of TOEs against STs written to conform to this PP is as follows:

After the ST has been approved for evaluation, the Common Criteria Testing Laboratory (CCTL) will obtain access to a TOE and its supporting environmental IT, as well as its administrative guidance. The Assurance Activity listed in the ST (which will be refined by the CCTL to be TOE-specific, either within the ST or in a separate document) will then be performed by the CCTL. The results of these activities will be documented and presented (along with the administrative guidance used) for validation.

For each family, “Developer Notes” are provided on the developer action elements to clarify what, if any, additional documentation/activity needs to be provided by the developer. For the content/presentation and evaluator activity elements, additional Assurance Activity are described as a whole for the family, rather than for each element. Additionally, the Assurance Activity described in this section are complementary to those specified in Section 4.

The TOE security assurance requirements identify the management and evaluative activities required to address the threats identified in Section 4 of this PP.

Class ADV: Development

The information about the TOE is contained in the guidance documentation available to the end user as well as the TOE Summary Specification (TSS) portion of the ST. The TOE developer must concur with the description of the product that is contained in the TSS as it relates to the functional requirements. The Assurance Activity contained in Section 4 should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

ADV_FSP.1 Basic functional specification

Developer action elements:

ADV_FSP.1.1D The developer shall provide a functional specification.

ADV_FSP.1.2D The developer shall provide a tracing from the functional specification to the SFRs.

Developer Note: As indicated in the introduction to this section, the functional specification is comprised of the information contained in the AGD_OPR and AGD_PRE documentation, coupled with the

information provided in the TSS of the ST. The Assurance Activity in the functional requirements point to evidence that should exist in the documentation and TSS section; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

Content and presentation elements:

ADV_FSP.1.1C *The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.*

ADV_FSP.1.2C *The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.*

ADV_FSP.1.3C *The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.*

ADV_FSP.1.4C *The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.*

Evaluator action elements:

ADV_ FSP.1.1E *The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.*

ADV_ FSP.1.2E *The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.*

Assurance Activity:

There are no specific Assurance Activity associated with these SARs. The functional specification documentation is provided to support the evaluation activities described in Section 4.2, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other Assurance Activity being performed; if the evaluator is unable to perform an activity because the there is insufficient interface information, then an adequate functional specification has not been provided.

Class AGD: Guidance Documents

The guidance documents will be provided with the developer's security target. Guidance must include a description of how the authorized user verifies that the Operational Environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by an authorized user.

Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes

- instructions to successfully install the TOE in that environment; and

- instructions to manage the security of the TOE as a product and as a component of the larger operational environment.

Guidance pertaining to particular security functionality is also provided; specific requirements on such guidance are contained in the Assurance Activity specified in Section 4.2.

AGD_OPE.1 Operational user guidance

Developer action elements:

AGD_OPE.1.1D The developer shall provide operational user guidance.

Developer Note: Rather than repeat information here, the developer should review the Assurance Activity for this component to ascertain the specifics of the guidance that the evaluators will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

Content and presentation elements:

AGD_OPE.1.1C The operational user guidance shall describe what the authorized user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2C The operational user guidance shall describe, for the authorized user, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3C The operational user guidance shall describe, for the authorized user, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.4C The operational user guidance shall, for the authorized user, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5C The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AGD_OPE.1.6C The operational user guidance shall, for the authorized user, describe the security measures to be followed in order to fulfill the security objectives for the operational environment as described in the ST.

AGD_OPE.1.7C The operational user guidance shall be clear and reasonable.

AGD_OPE.1.8C The operational user guidance shall be expressed in the eXtensible Configuration Checklist Description Format (XCCDF) to support security automation. [Appendix for US only] The operational user guidance shall express each configuration guidance item that could be used in a compliance checking regime as an XCCDF Rule element, and provide references to the NIST 800-53 controls which the item satisfies.

Evaluator action elements:

AGD_OPE.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Assurance Activity:

Some of the contents of the operational guidance will be verified by the Assurance Activity in Section 4.2 and evaluation of the TOE according to the CEM.

The documentation must describe the process for verifying updates to the TOE, either by checking the hash or by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

1. *Instructions for querying the current version of the TOE software.*
2. *For hashes, a description of where the hash for a given update can be obtained. For digital signatures, instructions for obtaining the certificate that will be used by the FCS_COP.1(2) mechanism to ensure that a signed update has been received from the certificate owner. This may be supplied with the product initially, or may be obtained by some other means.*
3. *Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).*
4. *Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.*

AGD_PRE.1 Preparative procedures

Developer action elements:

AGD_PRE.1.1D The developer shall provide the TOE including its preparative procedures.

Developer Note: As with the operational guidance, the developer should look to the Assurance Activity to determine the required content with respect to preparative procedures.

Content and presentation elements:

AGD_PRE.1.1C The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.2C The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

Evaluator action elements:

AGD_PRE.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_PRE.1.2E The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

Assurance Activity:

As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms (that is, combination of hardware and operating system) claimed for the TOE in the ST.

Class ALC: Life-cycle Support

At the assurance level provided for TOEs conformant to this PP, life-cycle support is limited to an examination of the TOE vendor's development and configuration management process. This is a result of the critical role that a developer's practices play in contributing to the overall trustworthiness of a product.

ALC_CMC.1 Labeling of the TOE

Developer action elements:

ALC_CMC.1.1D The developer shall provide the TOE and a reference for the TOE.

Content and presentation elements:

ALC_CMC.1.1C The TOE shall be labeled with its unique reference.

Evaluator action elements:

ALC_CMC.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Assurance Activity:

The evaluator shall verify that the TOE has been provided with its unique reference labeled. The evaluator shall verify that the CM documentation has been provided and that it describes the method used to uniquely identify each configuration item. The evaluator shall verify that the developer has used a CM system and that this system uniquely identifies each configuration item.

ALC_CMS.1 TOE CM coverage

Developer action elements:

ALC_CMS.1.1D The developer shall provide a configuration list for the TOE.

Content and presentation elements:

ALC_CMS.1.1C The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.1.2C The configuration list shall uniquely identify the configuration items.

Evaluator action elements:

ALC_CMS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Assurance Activity:

The evaluator shall verify that the developer has provided a configuration list for the TOE that contains each item highlighted above. The evaluator shall verify that each item in the configuration list is uniquely identified and its developer is indicated.

Class ATE: Tests

Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through ATE_IND family, while the latter is through the AVA_VAN family. At the assurance level specified in this PP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

ATE_IND.1 Independent testing – conformance

Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operation) documentation provided. The focus of the testing is to confirm that the requirements specified in Section 4.2 are being met, although some additional testing is specified for SARs in Section 4.3. The Assurance Activity identify the additional testing activities associated with these components. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this PP.

Developer action elements:

ATE_IND.1.1D The developer shall provide the TOE for testing.

Content and presentation elements:

ATE_IND.1.1C The TOE shall be suitable for testing.

Evaluator action elements:

ATE_IND.1.1E *The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.*

ATE_IND.1.2E *The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.*

Assurance Activity:

The evaluator shall prepare a test plan and report documenting the testing aspects of the system. The test plan covers all of the testing actions contained in the CEM and the body of this PP's Assurance Activity. While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluators must document in the test plan that each applicable testing requirement in the ST is covered.

The Test Plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluators are expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) is provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform. This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (DTLS, TLS/HTTPS).

The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results. The test report (which could just be an annotated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a "fail" and "pass" result (and the supporting details), and not just the "pass" result.

Class AVA: Vulnerability Assessment

168. For the first generation of this protection profile, the evaluation lab is expected to survey open sources to learn what vulnerabilities have been discovered in these types of products. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. Until penetration tools are created and uniformly distributed to the evaluation labs, evaluators will not be expected to test for these vulnerabilities in the TOE. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor. This information will be used in the development of penetration testing tools and for the development of future protection profiles.

AVA_VAN.1 Vulnerability survey

Developer action elements:

AVA_VAN.1.1D The developer shall provide the TOE for testing.

Content and presentation elements:

AVA_VAN.1.1C The TOE shall be suitable for testing.

Evaluator action elements:

AVA_VAN.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN.1.2E The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN.1.3E The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

Assurance Activity:

As with ATE_IND the evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to determine the vulnerabilities that have been found in MDMs in general, as well as those that pertain to the particular TOE. The evaluator documents the sources consulted and the vulnerabilities found in the report. For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. For example, if the vulnerability can be detected by pressing a key combination on boot-up, a test would be suitable at the assurance level of this PP. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

RATIONALE

The rationale tracing the threats to the objectives and the objectives to the requirements is contained in the prose in Sections 2.0 and 3.0. The only outstanding mappings are those for the Assumptions and Organizational Security Policies; those are contained in Annex A below.

Annex A: Supporting Tables

In this Protection Profile, the focus in the initial sections of the document is to use a narrative presentation in an attempt to increase the overall understandability of the threats to network devices; the methods used to mitigate those threats; and the extent of the mitigation achieved by compliant TOEs. This presentation style does not readily lend itself to a formalized evaluation activity, so this Annex contains the tabular artifacts that can be used for the evaluation activities associated with this document.

Assumptions

The specific conditions listed in the following subsections are assumed to exist in the TOE's Operational Environment. These assumptions include both practical realities in the development of the TOE security requirements and the essential environmental conditions on the use of the TOE.

ST authors should ensure that the assumptions still hold for their particular technology; the table should be modified as appropriate.

Table 1: TOE Assumptions

Assumption Name	Assumption Definition
A.PLATFORMS	The web browsers described in this document could run on any operating system, regardless of the underlying platform.
A.TRUSTED_ADMIN	TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.
A.TRUSTED_USER	The Web Browser User is not malicious, and exercises appropriate precautions.
A.PLATFORM_FUNCTIONS	The platform supporting the client shall offer cryptography, file system and other operating system capabilities.

Threats

The following threats should be integrated into the threats that are specific to the technology by the PP authors when including the requirements described in this document. Modifications, omissions, and additions to the requirements may impact this list, so the PP author should modify or delete these threats as appropriate.

Table 2: Threats

Threat Name	Threat Definition
T.UNAUTHORIZED_ADD-ON	Malicious or exploitable extensions or plug-ins could be used knowingly or unknowingly by a developer, possibly resulting in the capability of attacks against the platform‘s system software.
T.UNAUTHORIZED_UPDATE	Malicious or exploitable software could be used knowingly or unknowingly by a developer, possibly resulting in the capability of attacks against the platform‘s system software.
T.NETWORK_EAVESDROP	If positioned on a wireless communications channel or elsewhere on the network, attackers may monitor and gain access to data exchanged between the browser and other endpoints
T.NETWORK_ATTACK	An attacker may initiate communications with the browser or alter communications between the browser and other endpoints.
T.DATA_ACCESS	Loss of confidentiality of user data and credentials may be a result of an attacker gaining access to a browser while it is in operation.

Security Objectives for the TOE

Table 3: Security Objectives for the TOE

TOE Security Objective	TOE Objective Definition
O.COMMS	The TOE will provide the capability to communicate using one (or more) standard protocols as a means to maintain the confidentiality of data that are transmitted outside of the TOE.
O.CONFIG	The TOE will provide the capability to configure and apply security policies. This ensures the browser can protect user and enterprise data that it may store or process.
O.INTEGRITY	The TOE will provide the capability to perform self-tests to ensure the integrity of critical functionality, software/firmware and data has been maintained. The TOE will also provide a means to verify the integrity of

TOE Security Objective	TOE Objective Definition
	downloaded updates and control the download and launch of executables.
O.ISOLATION	The TOE will provide the capability to isolate content from different domains to prevent any unauthorized access.
O.STORAGE	The TOE will provide the capability to encrypt all user and enterprise data and authentication keys to ensure the confidentiality of data that it stores.

Security Threats to Security Objectives

The following table contains a mapping of Security Threats to Objectives for the TOE.

Table 4: Security Threats to Objectives Mapping

Threat	Objective
T.UNAUTHORIZED_ADD-ON	O.INTEGRITY
T.UNAUTHORIZED_UPDATE	O.INTEGRITY
T.NETWORK_EAVESDROP	O.COMMS
T.NETWORK_ATTACK	O.COMMS; O.ISOLATION
T.DATA_ACCESS	O.STORAGE; O.CONFIG

Annex B: Optional Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this PP. Additionally, there are three other types of requirements specified in Appendices B, C, and D.

The first type (in this Appendix) are requirements that can be included in the ST, but do not have to be in order for a TOE to claim conformance to this PP. The second type (in Appendix C) are requirements based on selections in the body of the PP: if certain selections are made, then additional requirements in that appendix will need to be included. The third type (in Appendix D) are components that are not required in order to conform to this PP, but will be included in the baseline requirements in future versions of this PP. Note that the ST author is responsible for ensuring that requirements that may be associated with those in Appendix B, Appendix C, and/or Appendix D but are not listed (e.g., FMT-type requirements) are also included in the ST.

At any time these may be included in the ST such that the TOE is still conformant to this PP.

B.1 Class: User Data Protection (FDP)

Private Browsing Sessions
FDP_PBR_EXT.1 Extended: Private Browsing Sessions

FDP_PBR_EXT.1.1 The TOE shall provide the capability to support private browsing sessions.

Application Note:

Private browsing is a mode that allows a user to browse the web without storing data such as browsing history, images, and cookies. Browsers that support this capability should include this requirement.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure that it describes how the TOE supports private browsing. The TSS shall specify the data that is not stored locally when in private browsing mode and describe the difference in interactions that the TOE has with websites when operating and not operating in this mode.

Guidance

The evaluator shall examine the operational guidance to ensure it includes the steps for allowing or disallowing private browsing

Tests

The evaluator shall perform the following test:

- Test 1: The evaluator shall clear the browsing history and cookie cache, and shall and enable private browsing. The evaluator shall navigate to test and/or actual websites and shall verify that the browsing history and cookie cache remain unchanged.
- Test 2: The evaluator shall disable private browsing and shall revisit the same websites. The evaluator shall verify that the browsing history and the cookie cache have been updated and reflect the evaluator’s browsing activities.

Annex C: Selection-based Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this PP. There are additional requirements based on selections in the body of the PP: if certain selections are made, then additional requirements below will need to be included.

C.1 Class: Cryptographic Support (FCS)

Datagram Transport Layer Security
FCS_DTLS_EXT.1 Extended: Datagram Transport Layer Security

FCS_DTLS_EXT.1.1(2) The [selection: TOE, TOE platform] shall implement the DTLS protocol in accordance with DTLS 1.2 (RFC 6347).

FCS_DTLS_EXT.1.2(2) The [selection: TOE, TOE platform] shall implement the requirements in FCS_TLS_EXT.1 for the DTLS implementation, except where variations are allowed according to RFC 6347.

Application Note:

Differences between DTLS and TLS are outlined in RFC 6347; otherwise the protocols are the same. In particular, for the applicable security characteristics defined for the TOE, the two protocols do not differ. Therefore, all application notes and Assurance Activity that are listed for FCS_TLSC_EXT.1 apply to the DTLS implementation.

Assurance Activity:

TSS

N/A

Guidance

N/A

Tests

The evaluator shall attempt to establish a connection with a DTLS server, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as DTLS. All other tests are performed in conjunction with the Assurance Activity listed for FCS_TLSC_EXT.1.

C.2 Class: User Data Protection (FDP)

Information Deletion

FDP_DEL_EXT.2 Extended: Deletion of Extension Information

FDP_DEL_EXT.2.1 The TOE shall provide the capability to delete extensions so that all information is removed, including extensions, configuration elements, and stored information.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure that it documents where extensions are stored, where extensions are allowed to store information.

Guidance

The evaluator shall examine the operational guidance to verify that it includes instructions for how the user can delete extensions.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall install a TOE extension, then examine the TOE's file system to verify that the extension and extension data are stored as documented. The evaluator shall then uninstall the TOE extension and examine the TOE's file system to verify that the extension and extension data are removed from the documented locations.

FDP_DEL_EXT.3 Extended: Deletion of Plug-in Information

FDP_DEL_EXT.3.1 The TOE shall provide the capability to delete plug-ins so that all information is removed, including plug-ins, configuration elements, and stored information.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure that it documents where plug-ins are stored, where plug-ins are allowed to store information, and whether the option exists to delete all plug-in information.

Guidance

The evaluator shall examine the operational guidance to verify that it includes instructions for how the user can delete plug-ins and associated content.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall install a TOE plug-in, then examine the TOE's file system to verify that the plug-in and plug-in data are stored as documented. The evaluator shall then uninstall the TOE plug-in and examine the TOE's file system to verify that the plug-in and plug-in data are removed from the documented locations.

C.3 Class: Protection of the TSF (FPT)

Trusted Update
FPT_TUD_EXT.2 Extended: Trusted Extension Update

FPT_TUD_EXT.2.1 The TOE shall provide the capability to query the current version of the extension.

FPT_TUD_EXT.2.2 The TOE shall provide the capability to initiate extension updates.

FTP_TUD_EXT.2.3 The TOE shall provide a means to verify extensions and extension updates using a digital signature mechanism prior to installing the extension or extension updates.

FTP_TUD_EXT.2.4 The TOE shall prevent websites from automatically installing extensions.

Application Note:

Extensions are bundles of code that are added to the browser to add specific functionality that the browser does not provide by default. Extensions could be developed by the browser vendor or by third parties and are allowed full access to view and interact with the web content that the browser sees. Extensions are common on non-mobile platforms, but may not be supported on mobile platforms, where HTML5 tends to be used for websites that serve rich content.

Assurance Activity:

TSS

The evaluator shall examine the TSS to verify that it describes the ability of the TSF to verify that extensions and extension updates come from a trusted source. The evaluator shall examine the TSS to verify that it states that the TSF will reject extensions from unapproved sources.

Guidance

The evaluator shall examine the operational guidance to verify that it includes instructions on how to configure the TOE with trusted extension sources.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall configure the TOE with trusted sources. The evaluator shall create or obtain an extension signed by a trusted source and attempt to install it. The evaluator shall verify that the signature on the extension is valid.
- Test 2: The evaluator shall create or obtain an extension signed by an untrusted source and attempt to install it. The evaluator shall verify that the signed extension is rejected.
- Test 3: The evaluator shall create or obtain an extension signed with an invalid certificate and attempt to install it. The evaluator shall verify that the signed extension is rejected.

- Test 4: The evaluator shall create or obtain an extension signed by a trusted source, modify the extension without re-signing it, and attempt to install it. The evaluator shall verify that the signed extension is rejected.

FPT_TUD_EXT.3 Extended: Trusted Plug-in Update

FPT_TUD_EXT.3.1 The TOE shall provide the ability to query the current version of the plug-in.

FPT_TUD_EXT.3.2 The TOE shall provide the ability to initiate the downloading of plug-ins and plug-in updates.

FTP_TUD_EXT.3.3 The TOE shall provide a means to verify plug-ins using a digital signature mechanism and [selection: published hash, no other functions] prior to installing the plug-in.

FTP_TUD_EXT.3.4 The TOE shall prevent websites from automatically installing plug-ins.

Assurance Activity:

TSS

The evaluator shall examine the TSS to verify that it states that the TSF will reject plug-ins from unapproved sources.

Guidance

The evaluator shall examine the operational guidance to verify that it includes instructions on how to configure the TOE with trusted plug-in sources.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall configure the TOE with trusted plug-in sources. The evaluator shall create or obtain a plug-in signed by a trusted source and attempt to install it. The evaluator shall verify that the signature on the plug-in is valid.
- Test 2: The evaluator shall create or obtain a plug-in signed by an untrusted source and attempt to install it. The evaluator shall verify that the signed plug-in is rejected.
- Test 3: The evaluator shall create or obtain a plug-in signed with an invalid certificate and attempt to install it. The evaluator shall verify that the signed plug-in is valid.
- Test 4: The evaluator shall create or obtain a plug-in signed by a trusted source, modify the plug-in without re-signing it, and attempt to install it. The evaluator shall verify that the signed plug-in is rejected.

Annex D: Objective Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this PP. There are additional requirements that specify security functionality that is desirable and these requirements are contained in this Annex. It is expected that these requirements will transition from objective requirements to baseline requirements in future versions of this PP.

D.1 Class: Security Audit (FAU)

Security Audit Data Generation

FAU_GEN.1 Audit Data Generation

FAU_GEN.1.1 The [selection: TOE, TOE platform] shall be able to generate an audit record of the following auditable events:

- Start-up and shutdown of the audit functions;
- All auditable events for the not specified level of audit;
- *Specifically defined auditable events listed in* Table 5].

FAU_GEN.1.2 The [selection: TOE, TOE platform] shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event.

Application Note:

The ST author can include other auditable events directly in the table; they are not limited to the list presented.

Assurance Activity:

TSS

N/A

Guidance

The evaluator shall examine the operational guidance to ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall ensure that the TSS describes every audit event type mandated by the PP and that the description of the fields contains the information required in FAU_GEN.1.2.

Tests

The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in FAU_GEN.1.1. This should include all instances of an event. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. For administrative actions, the evaluator shall test that each action determined by the evaluator above to be security relevant in the context of this PP is auditable. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the operational guidance, and that the fields in each audit record have the proper entries and that the audit records are provided in a manner suitable for interpretation. The evaluator shall also ensure the ability to apply searches of audit data based on the type of event, the user responsible for causing the event, and identity of the applicable certificate.

Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly. For example, testing performed to ensure that the operational guidance provided is correct verifies that AGD_OPE.1 is satisfied and should address the invocation of the administrative actions that are needed to verify the audit records are generated as expected.

Table 5. Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	None.	None.
FAU_SEL.1	All modifications to the audit configuration that occur while the audit collection functions are operating.	None.
FCS_CKM.1(*)	Failure of the key generation activity.	None.
FCS_CKM_EXT.1	None.	None.
FCS_CKM_EXT.4	Failure of the key zeroization process.	Identity of object or entity being cleared.
FCS_COP.1(1)	Failure of encryption or decryption.	Cryptographic mode of operation, name/identifier of object being encrypted/decrypted.
FCS_COP.1(2)	Failure of hashing function.	Cryptographic mode of operation, name/identifier of object being hashed.
FCS_COP.1(3)	Failure of cryptographic signature.	Cryptographic mode of operation, name/identifier of object being signed/verified.
FCS_COP.1(4)	Failure in cryptographic hashing for non-data integrity.	Cryptographic mode of operation, name/identifier of object being hashed.
FCS_DTLS_EXT.1	Failure to establish a DTLS session.	None.
FCS_HTTPS_EXT.1	Failure to establish a HTTPS session	
FCS_RBG_EXT.1	Failure of the randomization process.	None.
FCS_STS_EXT.1	None.	None.

FCS_TLSC_EXT.1	Failure to establish a TLS session.	None.
FDP_ACC_EXT.1	None.	None.
FDP_ACF_EXT.1	None.	
FDP_COO_EXT.1	Failure to block third party cookies.	None
FDP_CSP_EXT.1	None.	None.
FDP_DEL_EXT.1	None.	None.
FDP_DEL_EXT.2	None.	None.
FDP_DEL_EXT.3	None.	None.
FDP_PBR_EXT.1	None.	None.
FDP_PST_EXT.1	None.	None.
FDP_SBX_EXT.1	All violations of process restrictions.	None.
FDP_SOP_EXT.1	All exceptions to Same Origin Policy.	None.
FDP_STR_EXT.1	None.	None.
FDP_TRK_EXT.1	None.	None.
FIA_X509_EXT.1	Failure of the X.509 certificate validation.	Reason for failure of validation.
FIA_X509_EXT.2	None.	None.
FMT_MOF.1	Any security-relevant changes to the configuration of the TOE.	None.
FMT_SMF.1	None.	None.
FMT_SMR.1	None.	None.
FPT_DNL_EXT.1	None.	None.
FPT_DNL_EXT.2	None.	None.
FPT_INT_EXT.1	None.	None.
FPT_INT_EXT.2	None.	None.
FPT_INT_EXT.3	None.	None.
FPT_MCD_EXT.1	All instances of running unsigned mobile code. All instances of running mobile code from an untrusted or unverified source.	Certificate subject
FPT_TST_EXT.1	Execution of this set of TSF self-tests. Detected integrity violations.	For integrity violations, the TSF code file that caused the integrity violation.
FPT_TUD_EXT.1	Initiation of the update. Any failure to verify the integrity of the update.	None.
FPT_TUD_EXT.2	Initiation of the update. Any failure to verify the integrity of the update.	None.
FPT_TUD_EXT.3	Initiation of the update. Any failure to verify the integrity of the update.	None.
FTP_ITC.1	All attempts to establish a trusted channel. Detection of modification of channel data.	

Security Audit Event Selection
FAU_SEL.1 Selective Audit

FAU_SEL.1.1 The [selection: TOE, TOE platform] shall be able to select the set of events to be audited from the set of all auditable events based on the following attributes:

a) event type;
b) success of auditable security events;
c) failure of auditable security events; and
d) [assignment: *other attributes*].

Application Note:
The intent of this requirement is to identify all criteria that can be selected to trigger an audit event. This can be configured through an interface on the client for a user/administrator to invoke. For the ST author, the assignment is used to list any additional criteria or “none”. The auditable event types are listed in Table 5.

Assurance Activity:
TSS
N/A

Guidance
The evaluator shall examine the operational guidance to ensure that the guidance itemizes all event types, as well as describes all attributes that are to be selectable in accordance with the requirement, to include those attributes listed in the assignment. The operational guidance shall also identify those audit records that are always recorded, regardless of the selection criteria currently being enforced.

Tests
The testing should be accomplished in conjunction with the testing of FAU_GEN.1.1.

D.2 Class: Cryptographic Support (FCS)

Strict Transport Security

FCS_STS_EXT.1 Extended: Strict Transport Security

FCS_STS_EXT.1.1 The [selection: TOE, TOE platform] shall implement HTTP Strict-Transport-Security according to RFC 6797

FCS_STS_EXT.1.2 The [selection: TOE, TOE platform] shall retain persistent data signaling HSTS enablement for the time span declared by the website in a max-age directive.

FCS_STS_EXT.1.3 The [selection: TOE, TOE platform] shall cache the freshest Strict Security policy information.

Application Note:
If a browser receives the HSTS header from a website, all future HTTP sessions between the TOE and the domain or superdomain of that website must occur over TLS 1.2 (RFC 5246) or greater by utilizing HTTPS (RFC 2818).

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure that it documents how the TOE supports HSTS.

Guidance

The evaluator shall examine the operational guidance to ensure it contains instructions on how to use HSTS.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall connect to a HSTS-compliant website while running a network protocol analyzer to monitor the traffic. The evaluator shall examine the captured network traffic and verify that a Strict Transport Security header is received and that there is a directive for the max-age of the HSTS relationship.
- Test 2: The evaluator shall reconnect to the HSTS website again over HTTP and shall verify that the session is redirected to HTTPS.
- Test 3: The evaluator shall reconnect to the HSTS website after the max-age has expired, and verify that the website and TOE reestablish an HSTS relationship.
- Test 4: The evaluator shall update the website HSTS information, and verify that when the TOE reconnects to the website, that information is updated by the TOE.

D.3 Class: User Data Protection (FDP)

Access Control Policy
FDP_ACC_EXT.1 Extended: Access Control on Domains

FDP_ACC_EXT.1.1 The TOE shall provide the capability to group domains and apply access restrictions to specific sets of domains.

Application Note:

An example of this functionality is the concept of zones in Internet Explorer.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure it describes how the TSF supports the grouping of domains and applying access restrictions to them.

Guidance

The evaluator shall examine the operational guidance to ensure it documents the steps for configuring the ability to group domains and apply access restrictions.

Tests

The evaluator shall perform the following tests:

- The evaluator shall create a group of domains and apply a specific restriction to the group. The evaluator shall navigate to websites within the domains in the restricted group and shall verify that the restriction is functional. The evaluator shall navigate to websites within domains not in the restricted group and shall verify that the restriction does not apply.

Storage of Persistent Information
FDP_PST_EXT.1 Extended: Storage of Persistent Information

FDP_PST_EXT.1.1 The TOE shall provide the capability to operate without storing persistent data to the file system.

Application Note:

Exceptions from this requirement are credentials and configuration information.

Assurance Activity:

TSS

The evaluator shall examine the TSS to verify it describes how the TOE operates without storing persistent user data to the file systems.

Guidance

N/A

Tests

The evaluator shall perform the following test:

- Test 1: The evaluator shall operate the TOE for a period of time, ensuring that a wide variety of TOE functionality has been exercised. The evaluator shall then examine the TOE and the underlying platform to ensure that no files have been written to the file system other than credentials or configuration information.

D.4 Class: Protection of the TSF (FPT)

TOE Interaction with External Entities
FPT_INT_EXT.2 Extended: Interactions with Application Reputation Services

FPT_INT_EXT.2.1 The TOE shall utilize an application reputation service to prevent downloading of malicious applications.

Application Note:

An application reputation service is an online service that identifies malicious applications; it is used to detect such applications prior to downloading them.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure it describes the TOE's use of application reputation services in detecting malicious applications.

Guidance

The evaluator shall examine the operational guidance to ensure it describes the TOE's support for use of an application reputation service, including which services the TOE supports by default (if any) and whether additional services can be configured. The operational guidance shall include steps for how to configure the application reputation service.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall configure the TOE to enable the use of one or more application reputation services per the operational guidance. The evaluator shall initiate a connection with a website that attempts to download an application to the TOE while sniffing the network traffic using a network protocol analyzer. The evaluator shall inspect the captured network traffic and shall verify that the TOE initiates a connection to the configured application reputation service(s) before initiating the download.

FPT_INT_EXT.3 Extended: Interactions with URL Reputation Services

FPT_INT_EXT.3.1 The TOE shall utilize a URL reputation service to prevent connections with malicious websites.

Application Note:

A URL reputation service is an online service that identifies websites with malicious or phishing content applications; it is used to detect such websites prior to allowing users to access them.

The goal of this requirement is to ensure that the TOE is prevented from establishing connections with known-bad sources of malware on the Internet. The specifics of the sequence of actions taken before a block decision is made may depend upon the specific implementation of the TOE. For example, some TOEs might implement the check for malicious content by checking against the list of bad URLs provided by the URL reputation service in real time; others may download updated lists of bad URLs at TOE startup, updating the list periodically from the URL reputation service(s) until the TOE is terminated. Ultimately, the result should be that the TOE blocks the connection to the bad URL.

Assurance Activity:

TSS

The evaluator shall examine the TSS to ensure it describes the TSF's use of a URL reputation service in detecting malicious websites.

Guidance

The evaluator shall examine the operational guidance to ensure it describes the TOE's support for use of URL reputation services, including which services the TOE supports by default (if any) and whether additional services can be configured. The operational guidance shall include steps for how to configure the URL reputation service.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall configure the TOE to enable the use of one or more URL reputation services per the operational guidance. The evaluator shall initiate a connection with a known good website while sniffing the network traffic using a network protocol analyzer. The evaluator shall inspect the captured network traffic and shall verify that the TOE initiates a connection to the configured URL reputation service(s).
- Test 2: The evaluator shall configure the TOE to enable the use of one or more URL reputation services per the operational guidance. The evaluator shall initiate a connection with a known malicious website that is identified by one or more of the URL reputation services while sniffing the network traffic using a network protocol analyzer. The evaluator shall verify that a warning appears alerting that the website is known to be malicious and the TOE is not allowed to connect. The evaluator shall inspect the captured network traffic and shall verify that the TOE initiates a connection to the configured URL reputation service(s) and retrieved an updated list of malicious URLs with the tested website being on the list.

Annex E: Glossary and Acronyms

E.1 Technical Definitions

Active Content	code that runs in a browser, either natively or via a plug-in, without user interaction
Administrator	The Administrator is responsible for management activities, including setting the policy that is applied by the enterprise on the Browser. This administrator is likely to be acting remotely. If the device is unattached to an enterprise, the user is the administrator.
Application	software that runs on an operating system and performs tasks on behalf of the user or owner of the platform
CSRF	Cross Site Request Forgery - Vulnerability where an attacker gets a target user to execute a script with that user's privileges
DTLS	Datagram Transport Layer Security
Extension	bundle of code that are added to the browser to add specific functionality that the browser does not provide by default
GPU	Graphics Processing Unit
HTML	HyperText Markup Language - Language used by web servers to present content to browsers
HTML5	HyperText Markup Language version 5, a new version of HTML that incorporates many new features that enrich the browsing experience
HTTP	HyperText Transfer Protocol - Protocol for communicating on the web
HTTPS	HyperText Transfer Protocol Secure; secure version of HTTP that runs over an encrypted channel (SSL/TLS)
JavaScript	programming language commonly used as part of a web browser to enable programmatic access to computational objects
Plug-in	browser add-on to handle specific types of web content
Pop-up	piece of web code that causes a browser window to open outside of the browser instance that is currently in focus
Sandbox	Security mechanism for separating running programs, most often used to test or run unverified programs that may contain malicious code without allowing the programs to harm the host system

Tabs	allow the browsers to display content from multiple web sites without starting another instance of the browser
Web Browser User	A user operating the TOE in unprivileged mode
Web Application	software application utilizing web technologies (e.g., Flash, Java, HTML) that runs in a web browser or other client application
Web Browser	application that retrieves and renders content provided by a web server
Whitelist	mechanism for specifying a list of domains that are allowed to do something
XSS	Cross Site Scripting - Vulnerability resulting from insufficient filtering of special characters that allows an attacker to inject a script into a web page, which is later displayed by a browser; allows the attack to deface web sites or steal session information

E.2 Common Criteria Definitions

Assurance	Grounds for confidence that a TOE meets the SFRs [CC1].
CC	Common Criteria
PP	Protection Profile
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
Security Target (ST)	Implementation dependent statement of security needs for a specific identified TOE.
Target of Evaluation (TOE)	Set of software, firmware and hardware under evaluation, possibly accompanied by guidance.
TOE Security Functionality (TSF)	Combined functionality of all hardware, software, and firmware of a TOE that must be relied upon for the correct enforcement of the SFR
TOE Summary Specification (TSS)	Documentation which provides evaluators with a description of the implementation of SFRs in the TOE.

E.3 Acronyms

AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CRL	Certificate Revocation List
CSP	Cryptographic Service Provider
CSRF	Cross Site Request Forgery
DHE	Diffie-Hellman Key Exchange
DN	Distinguished Name
DSA	Digital Signature Algorithm
DTLS	Datagram Transport Layer Security
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
FFC	Finite-Field Cryptography
FIPS	Federal Information Processing Standards
GCM	Galois/Counter Mode
GPU	Graphics Processing Unit
HMAC	Keyed Hash Message Authentication Code
HTML	HyperText Markup Language
HTML5	HyperText Markup Language version 5
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IETF	Internet Engineering Task Force
IV	Initialization Vector
KAT	Known Answer Test
KDF	Key Derivation Function
NIST	National Institute of Standards and Technology

OCSP	Online Certificate Status Protocol
OID	Object Identifier
PDF	Portable Document Format
rDSA	RSA Digital Signature Algorithm
RFC	Request for Comment (IETF)
RGB	Random Bit Generator
RSA	Rivest Shamir Adelman
SaaS	Software as a Service
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
TLS	Transport Layer Security
W3C	World Wide Web Consortium
XSS	Cross Site Scripting