Requirements from the *Application Software Extended Package for Web Browsers*

Version: 2.0

2015-06-16

National Information Assurance Partnership

Revision History

Version	Date	Comment
v 2.0	2015-06-16	Application Software Extended Package for Web Browsers
v 1.0	2014-03-31	Initial release - Protection Profile for Web Browsers

Introduction

Purpose. This document presents simplified view of the functional and assurance requirements found in the *Application Software Extended Package for Web Browsers*. Common Criteria evaluation, facilitated in the U.S. by the National Information Assurance Partnership (NIAP), is required for IA and IA-enabled products in National Security Systems according to CNSS Policy #11.

Using this document. This representation of the Protection Profile includes:

- *Security Functional Requirements* for use in evaluation. These are featured without the formal Assurance Activities specified in the Protection Profile, to assist the reader who is interested only in the requirements.

  It also includes, in tables shown later, particular types of security functional requirements that are not strictly required in all cases. These are:

  - *Selection-based Security Functional Requirements* which become required when certain selections are made inside the regular Security Functionality Requirements (as indicated by the **[selection:]** construct).
  - *Objective Security Functional Requirements* which are highly desired but not yet widely-available in commercial technology.
  - *Optional Security Functional Requirements* which are available for evaluation and which some customers may insist upon.

- *Security Assurance Requirements* which relate to developer support for the product under evaluation, development processes, and other non-functionality security relevant requirements.

Security Functional Requirements

FDP_ACF_EXT.1.1 The browser shall separate local (permanent) and session (ephemeral) storage based on domain, protocol and port:

- Session storage shall be accessible only from the originating window/tab;
- Local storage shall only be accessible from windows/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".

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

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

- The rendering process can only directly access the area of the file system dedicated to the browser.
- The rendering process can only directly invoke inter-process communication mechanisms with its own browser processes.
- The rendering process has reduced privilege with respect to other browser processes [**selection**: *[**assignment**: other methods by which the principle of least privilege is implemented 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 browser 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-browser processes on the host. Typically, if a rendering process needs to access a file or communicate with a non-browser 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 browser 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 browser.

FDP_SOP_EXT.1.1 The browser shall only permit scripts contained in one web page to access data in a second web page if both pages are from the same origin.

FDP_SOP_EXT.1.2 The browser shall enforce the same origin policy for all 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.

FDP_STR_EXT.1.1 The browser 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".

FDP_TRK_EXT.1.1 The browser shall provide notification to the user when tracking information for [**selection**:

geolocation,

browser history,

browser preferences,

browser statistics

] is requested by a website.

FMT_MOF_EXT.1.1 The browser shall be capable of performing the following management functions, controlled by the administrator or user as shown:

- X = Mandatory
- O = Optional

Management Function	Administrator	User
Enable/disable storage of third party cookies	O	X
Enable/disable use of OCSP for obtaining the revocation status of X.509 certificates	O	O
Configure inclusion of user-agent information in HTTP headers	O	O
Enable/disable ability for websites to collect tracking information about the user through [**selection**: *zombie cookies*, *add-on based tracking (e.g. Flash cookies)*, *browsing history*, *[**assignment**: other tracking mechanisms]*]	O	O
Enable/disable deletion of stored browsing data (cache, web form information)	O	X
Enable/disable storage of sensitive information (e.g., auto-fill, auto-complete) in persistent storage	O	O
Configure size of cookie cache	O	O
Configure size of cache	O	O
Enable/disable interaction with Graphic Processing Units (GPUs)	O	O
Configure the ability to advance to a web site with an invalid or unvalidated X.509 certificate	O	O
Enable/disable establishment of a trusted channel if the browser cannot establish a connection to determine the validity of a certificate	O	O
Configure the use of an application reputation service to detect malicious applications prior to download	O	O
Configure the use of a URL reputation service to detect sites that contain malware or phishing content	O	O
Enable/disable automatic installation of software updates and patches	O	O
Enable/disable ability for websites to register protocol handlers	O	O
Enable/disable display notification when unsigned, untrusted or unverified mobile code is encountered	O	O
Enable/disable user's ability to select default actions upon download of a file (e.g., always open, or always save, a downloaded file)	O	O
Enable/disable launching of downloaded files outside the browser	O	O
Enable/disable JavaScript	O	O
Enable/disable [**selection**: *ActiveX*, *Flash*, *Java*, *[**assignment**: other mobile code types supported by the browser]*] mobile code	O	O
Enable/disable support for add-ons	O	O
Enable/disable individual add-ons	O	O
Enable/disable HSTS mode	O	O

Application Note: For these management functions, the term "Administrator" refers to the administrator of a non-mobile device or the device owner of a mobile device. The intent of this requirement is to allow the user and administrator of the platform to configure the browser with configuration policies. 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 browser itself, or the browser and its platform in coordination with each other.

Disabling OCSP shall only be permitted if CRL was selected in FIA_X509_EXT.1.1 ().

FPT_DNL_EXT.1.1 The browser shall prevent downloaded content from launching automatically.

FPT_DNL_EXT.1.2 The browser shall present the user with the option to either save or discard downloaded files.

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

In this context, an executable is a file containing code for a software program that is invoked independent of and outside the context of the browser. It does not include mobile code, scripts, or add-ons.

FPT_MCD_EXT.1.1 The browser shall support the capability to execute signed [**selection**:

ActiveX,

Flash,

Java,

ActionScript,

*[**assignment**: other mobile code types supported by the browser]* ,

no

] mobile code.

FPT_MCD_EXT.1.2 The browser shall provide the user with the option to discard unsigned, untrusted or unverified [**selection**:

ActiveX,

Flash,

Java,

ActionScript,

*[**assignment**: other mobile code types supported by the browser]*

] mobile code without executing it.

Application Note: The ST author must specify all mobile code types for which the browser provides this support.

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

FPT_AON_EXT.1.1 The browser shall include the capability to load [**selection**: *trusted add-ons*, *no add-ons*] .

Application Note: FPT_AON_EXT.2 depends upon the selection made here. If the browser does not include support for installing only trusted add-ons, this requirement can be met by demonstrating the ability to disable all support for add-ons as specified in FMT_MOF_EXT.1. Cryptographic verification (i.e., trust) of add-ons is tested in FPT_AON_EXT.2.1

Security Assurance Requirements

Selection-Based Security Functional Requirements

FPT_AON_EXT.2.1 The browser shall [**selection**: *provide the ability*, *leverage the platform*] to provide a means to cryptographically verify add-ons using a digital signature mechanism and [**selection**: *published hash*, *no other functions*] prior to installation and update.

This is a selection-based requirement. Its inclusion depends upon selection in FPT_AON_EXT.1.1.

FPT_AON_EXT.2.2 The browser shall [**selection**: *provide the ability*, *leverage the platform*] to query the current version of the add-on.

This is a selection-based requirement. Its inclusion depends upon selection in FPT_AON_EXT.1.1.

FPT_AON_EXT.2.3 The browser shall prevent the automatic installation of add-ons.

This is a selection-based requirement. Its inclusion depends upon selection in FPT_AON_EXT.1.1.

Objective Security Functional Requirements

FCS_STS_EXT.1.1 The browser shall implement HTTP Strict-Transport-Security according to RFC 6797.

This is currently an objective requirement.

FCS_STS_EXT.1.2 The browser shall retain persistent data signaling HSTS enablement for the time span declared by the website in a

max-age directive.

This is currently an objective requirement.

FCS_STS_EXT.1.3 The browser shall cache the "freshest" Strict Security policy information.

This is currently an objective requirement.

Application Note: Freshness refers to the length of time between generation by the origin server and the expiration time when the origin server specifies that a stored response can no longer be used by a cache without further validation (RFCs 6797 and 7234). If a browser receives the HSTS header from a website, all future HTTP sessions between the browser and the domain or superdomain of that website must occur over TLS 1.2 (RFC 5246) or greater by utilizing HTTPS (RFC 2818) negotiating the strongest cipher possible.

FPT_INT_EXT.1.1 The browser shall utilize an application reputation service to prevent downloading of malicious applications.

This is currently an objective requirement.

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. Using a reputation service would require configuration of the trusted service to be used. The quality of the reputation service may fall outside of the scope of the evaluation.

FPT_INT_EXT.2.1 The browser shall utilize a URL reputation service to prevent connections with malicious websites.

This is currently an objective requirement.

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 browser 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 browser. For example, some browsers 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 browser startup, updating the list periodically from the URL reputation service(s) until the browser is terminated. Ultimately, the result should be that the browser blocks the connection to the bad URL.

Optional Security Functional Requirements

FDP_PST_EXT.1.1 The browser shall provide the capability to operate without storing persistent data to the file system with the following exceptions: [**selection**: *credential information*, *administrator provided configuration information*, *certificate revocation information*, *no exceptions*] .

Application Note: Any data that persists after the browser closes, including temporary files, is considered to be persistent data.
