Managing Access to Security Hardware in PC Browsers

Darmawan Suwirya, H. Karen Lu, and Laurent Castillo

Gemalto, Technology and Innovation, Austin, TX, USA
{darmawan.suwirya, karen.lu, laurent.castillo}@gemalto.com

Abstract. Web applications that require higher levels of security have various security options that can be deployed on the server. However providing security on the client consuming the service remains a challenge especially when the application runs in a web browser. While various types of security hardware can work with the user's computer, there is no convenient way for web applications to access such hardware from the web browser. To fill this gap, we develop a software tool called SEAM, which is a SEcure Add-on Management framework built on top of the web browser native extension framework. Through SEAM, web applications can access secure hardware in a controlled manner. SEAM was devised with usability and security at the core. It is flexible and convenient to use, easy to deploy, and works across major PC platforms and web browsers. This paper describes the rationale and design of SEAM, and illustrates its applications.

Keywords: SEAM, web application, secure hardware, web browser, browser extension, SConnect

1 Introduction

Web applications' ubiquity, ease of use, ease of deployment, and cross platform availability make them a preferred way for providers to deliver services, and for users to consume these services. Many such applications, especially those involving personal identifiable information and/or high value transactions, require higher levels of security and privacy protections. Service providers can deploy various security measures, such as firewall and intrusion detection, on the server side, but they still have difficulties to deploy any on the client side because their clients mostly run the application in web browsers, in which they have little control. Adversaries take this opportunity and utilize vulnerabilities of web applications to attack the client side for their own financial gains.

One way to enhance the security is to adopt multi-factor authentication that protect both web applications and users. This requires at least two factors among "what you know", "what you have", "who you are", "where you are", and so on. Beside the what-you-know factor (e.g. username/password), other factors typically require special hardware in acquiring the information. Although web browsers have been continuously improving in term of usability and security, there is still no convenient way for web applications to access such hardware from the web browser. To fill this gap, we develop a software tool called SEAM, which is a SEcure Add-ons Management framework built on top of the web browser native extension framework. Through SEAM, web applications can access security hardware in a controlled manner.

SEAM is based on our previous work SConnect [1], which addresses the need of web applications to access smartcards in PC browsers. Most of the SConnect concepts and values are still valid and remain as the state-of-the art, e.g. architecture, design, implementation choices, and security, application and deployment models. As the use cases grew and learning from years of usage and deployment experiences, it became necessary to extend SConnect to support new usage and deployment scenarios.

The remaining content of this paper will explain about the new requirements in section 2. We'll describe an application example in section 3 and present the high level SEAM architecture in section 4. We'll explain the security mechanisms in section 5 and finally outline the SEAM implementation in section 6. Ensuring the integrity of web applications, web browsers, and user computers is out of the scope of this paper.

2 Requirements

SEAM intends to address the following new requirements while still satisfying the original SConnect requirements in term of *security*, *ease of use* (for both software developers and end users), *simplicity of deployment*, and *availability* (across major PC platforms and web browsers):

Multi Hardware Support. Some use cases need hardware connectivity beyond the smartcard. Some may even require more than one type of hardware connectivity concurrently. For example, a Match-on-Card [3] application would require connectivity to both biometric and smartcard reader devices in order to perform the user authentication.

Hardware Driver Deployment. The smartcard reader API has been well standardized through PCSC [2] and its driver usually comes by default with most of OS installations. Unfortunately, that is rarely the case for other types of security hardware. They usually come with their own set of drivers that users have to install manually. For a web application to get the best end user experience, it is critical for it to be able to easily deploy and install device drivers.

Multi Versioning Support. Earlier SConnect's design and implementation revealed issues with sharing the use of an add-on among multiple web applications. In practice, multiple versions of the add-on need to be installed and co-exist at the same time in the user's browser. This means that we need to have a way to allow sharing usage of SEAM among multiple web applications in a safe manner, without the risk of breaking web applications when upgrading another.

Easy Customization. Web applications often require customization of the add-ons, for instance for branding, consistency of the UI, or addition of new features. SEAM should provide a way to support and manage this kind of customization efficiently.

Easy Deployment. As we anticipate that there will be a large number of SEAM package variants as new features are added, automating add-ons deployment becomes a very crucial point to address, for both the web applications developers and end user experience.

3 Application

Let's take a real use case example of a web application offering a strong multi factors authentication service that utilizes both smartcard and biometric devices. In this application, the protocol is based on a simple PKI based challenge response authentication scheme: the server issues a challenge, sends it to the client for signature computation, and finally verifies the signature received.

On the client side, signature computation is performed by the PKI application on the smartcard. For added security and convenience, it's also utilizing the Match-on-Card application on the smartcard, replacing the traditional PIN verification. In this implementation, the application requires two connectivity add-ons and one functional add-on: PCSC smartcard reader connectivity, Futronic FS80 [4] fingerprint reader connectivity and FingerjetFX [5] fingerprint extractor function. Futronic FS80 also requires installation of its own driver.

To use the application, the user will be first requested to insert her smartcard. After detecting the insertion of a valid smartcard, containing PKI and Match-on-Card applications, and a valid fingerprint reader device, the application asks the user to scan her fingerprint. In the case where any fingerprint related component is missing, the application will provide a fallback choice on traditional PIN verification.

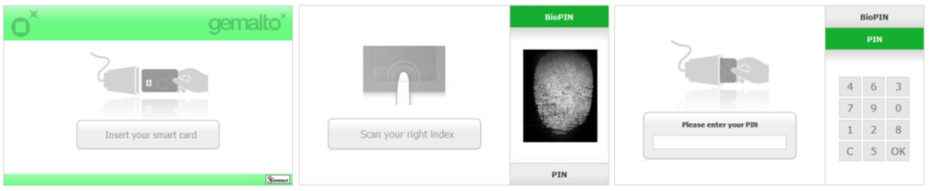

Figure 1. Match-on-Card application example

4 Architecture

Figure 2 below describes the high level, end-to-end architecture of the SEAM solution. SEAM consists of client and server side components. The client side component, called SEAM Extension, is in the form of a browser extension that enables JavaScript code running in the browser to access native resources, e.g. security hardware. The server side component, called SEAM Library, is a JavaScript library that web applications download on-demand. It encapsulates the SEAM Extension functionalities in an easy-to-use application interface.

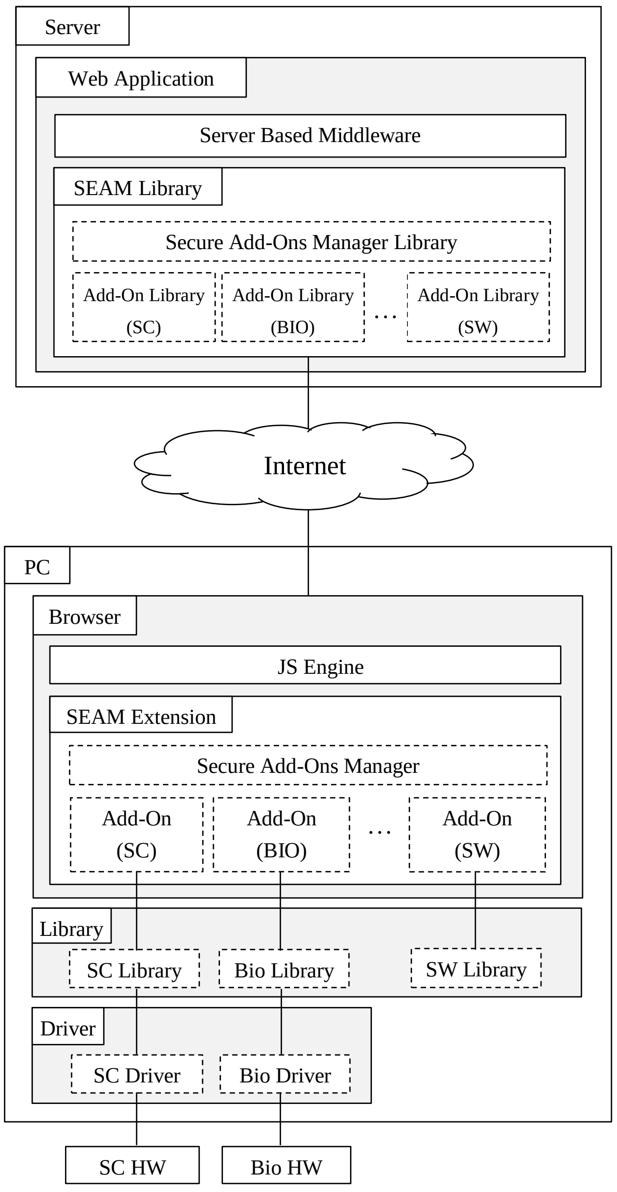

Figure 2. SEAM architecture

The SEAM Extension itself is made of two parts: the core and the add-ons. The core works as a secure add-ons manager on top of the browser's native extension framework: for instance, it offers services to install, list, and remove add-ons. The add-ons provide the actual implementation for accessing native resources. Delving a bit deeper, the core consists of the following components:

Security Gateway. This module is needed to prevent malicious web applications from accessing the security hardware through the add-ons. Without passing all the security checks and requirements imposed by this layer, the web application cannot use the SEAM Extension. The security mechanisms will be further detailed in the next chapter.

Add-Ons Package Manager. This module is responsible for managing the lifecycle of add-on packages, e.g. installation and un-installation, and provides the add-ons management user interface.

Add-ons Runtime Manager. This module is responsible for hot plugging add-ons on install and removal. It makes sure each add-on is correctly instantiated, and corresponding API calls are correctly mapped and routed at runtime.

5 Security Mechanism

Enabling web applications to access security hardware will enhance security. At the same time, this will also enlarge the attack surface. To protect the user, the security hardware, and the web application, SEAM employs several security mechanisms described below. When any of these security checks fails, SEAM prevents the web application from accessing the security hardware.

Digital Signature. SEAM Extension is digitally signed by using a code signing certificate issued by a trusted and well-known certificate authority. A signed extension instills confidence in users by validating the source of the extension.

Enforce HTTPS. In order to ensure secure communication with the remote web server and to prevent man-in-the-middle attacks (MITM), SEAM Extension mandates HTTPS between browser and remote web server.

Server Validation. During SSL handshaking, the browser receives and examines the website's SSL certificate. When the browser determines that certificate is invalid, it presents a warning to the user. Many users ignore the warning and continue [6], which may land them at a malicious website or make them fall victim to MITM attacks. To mitigate this risk, the SEAM Extension does an additional SSL certificate validation.

Connection Key. The Connection Key uniquely binds the SEAM-based web application with the SEAM Extension. The SEAM Extension issuer (e.g. Gemalto) signs the Connection Key with an asymmetric key whose public part is embedded in the Extension. The Connection Key contains additional information about the web application server, like its add-ons usage permissions and SSL fingerprint. The owner of a web application must go through a strict vetting process defined by the issuer in order to obtain the Connection Key, which ensures controlled distribution.

Revocation List. SEAM Extension also implements CRL (Certificate Revocation List) mechanism to check the revocation status of the Connection Key.

User Consent. As the last step, after passing all the transparent security checks mentioned earlier, the SEAM Extension will ask for user permission at the beginning of each use. For convenience, it also provides a way to remember a user's decision for that particular domain name.

Signed Add-Ons Package. This mechanism is added in SEAM to ensure the integrity and security of the add-on package installation. The mechanism is similar to the one used for the Connection Key, utilizing a different set of key-pairs specific for this purpose. Signature is applied to a cryptographic hash of the add-on package payload, and embedded inside the add-on's manifest file.

6 Implementation

On the server side, SEAM and each of the add-ons come with their associated JavaScript libraries that will allow an easier integration in web applications. These JavaScript libraries abstract all the direct calls to the SEAM Extension and add-ons functionalities, and hide differences that might exist from one browser specific implementation to another.

On the client side, SEAM is implemented as a browser extension, native to each browser extensibility technology, targeting three major OS, i.e. Windows, Linux and OSX, and four major browsers, i.e. Internet Explorer, Firefox, Chrome and Safari. For Internet Explorer, it uses ActiveX and Browser Helper Object, and is packaged as an .exe installer. For Firefox, it uses XPCOM, and is packaged as .xpi bundle. For Chrome, it uses a combination of Chrome Extensions and NPAPI, and is packaged as .crx bundle. For Safari, it uses a combination of Safari Extensions and NPAPI, and is packaged as .pkg installer.

The resulting SEAM Extension installer packages are kept around a few hundreds KB each. It ensures fast and smooth download times. Install experience and access to add-ons management page are maintained as close as possible to each browser's native experience. This familiarity ensures a better installation and use experience.

SEAM Add-ons are packaged in a proprietary format, which is basically a zip file containing the following components:

- **addon.zip.** This file contains all the necessary components for the add-on implementation itself, e.g. security hardware connectivity.
- **driver.zip.** This file contains all the necessary components for driver installation. This file is optional and only present if a hardware driver deployment and installation is required.

- **icon.png.** This file is the icon file that will be used and displayed in the install dialog and add-ons management page.
- **manifest.json.** This file contains the declaration and description of the add-on package itself, encoded in JSON format. Most importantly, it also declares signature values for checking the integrity of driver.zip and addon.zip files contained inside the package.

7 Conclusion

SEAM is a secure add-ons management framework built on top of native browser extension mechanisms. It allows web applications to access native resources in a secure and flexible manner. When the web application needs to interface with a new hardware, SEAM also enables it to easily deploy the associated device driver. SEAM employs several security mechanisms in the implementation that will protect its use from various malicious attack scenarios. At the same time, the SEAM design puts the main focus on user experience, overall usability, and maintainability of the solution. SEAM is a work in progress. We continue improving its functionality and performance.

8 Acknowledgement

The authors gratefully acknowledge the feedback and support from Dr. Ksheerabdhi Krishna, Dr. Olivier Potonniee and Michael Hutchinson.

References

1. Kapil Sachdeva, H. Karen Lu, Ksheerabdhi Krishna, *“A Browser-Based Approach to Smart Card Connectivity”*, IEEE Workshop on Web 2.0 Security & Privacy, Oakland, CA, 2009.
2. PC/SC Workgroup, *Specifications*, http://www.pcscworkgroup.com, V 2.01.11, June 2012.
3. Match-on-Card, http://www.matchoncard.com/what-is-moc/
4. Futronic, FS80 USB 2.0 Fingerprint Scanner, http://www.futronic-tech.com/product_fs80.html
5. DigitalPersona, FingerJetFXose, http://www.digitalpersona.com/fingerjetfx/
6. Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, Lorrie Faith Cranor, *“Crying Wolf: An Empirical Study of SSL Warning Effectiveness”*, The 18th Usenix Security Symposium, Montreal, Canada, Aug. 10-14, 2009.