Towards Enhancing Web Application Security Using Trusted Execution

Cornelius Namiluko, Andrew J. Paverd, and Tulio De Souza

Department of Computer Science, Oxford University
Wolfson Building, Parks Road, Oxford OX1 3QD, UK
Email: [firstname.lastname]@cs.ox.ac.uk

Abstract. The web continues to serve as a powerful medium through which various services and resources can be exposed or consumed through web applications. Web application platforms such as *webinos* facilitate communication between the various smart devices in a personal network. Although modern web applications use various cryptographic techniques for authentication and encryption, the security of these techniques is directly linked to the security of the private (secret) keys. Although various techniques exist to protect these keys, we argue that the use of secure hardware can provide stronger security guarantees. In particular, we describe our work-in-progress experiments towards using functionality provided by a Trusted Execution Environment (TEE) in web applications. These experiments include an implementation of the *webinos* platform integrated with ARM TrustZone technology. Our preliminary results are promising in terms of both the feasibility of implementing this architecture and the performance of the system.

Keywords: ARM TrustZone, GlobalPlatform, Trusted Execution Environment, webinos,

1 Introduction

Wide-spread use of smart mobile devices has opened up a range of new possibilities for web applications. Feature-rich web applications can be designed to use resources such as web cameras, GPS receivers and Near Field Communication (NFC) transceivers to provide interactive services. As a result, personal information such as location, messages and contacts is increasingly being exposed to the web leading to various security concerns about the confidentiality, integrity and availability of this sensitive information. Rather than relying on software alone to manage access to resources on these devices, it has been proposed that protection should be included as part of the hardware platform [3]. Grawrock [3] argues that a viable approach towards device security is through trusted execution — a paradigm in which non-security sensitive operations cannot influence sensitive operations even though both take place on the same platform. This provides the capability to control access to sensitive information and resources. However,

trusted execution functionality is normally provided at a low level of abstraction and in order for applications running at a higher level of abstraction to utilize this functionality, there must be mechanisms to expose the functionality in a flexible manner without compromising security.

In this paper, we propose that the security of web applications can be enhanced through a device-independent framework that enables web applications to utilize the functionality provided by trusted execution. We focus on *webinos*[1], a state-of-the-art platform for running web applications across multiple devices. Details of the webinos architecture are presented in Section 2. We propose that webinos can be enhanced to take advantage of trusted execution functionality provided by ARM TrustZone technology described in Section 3. In Section 4, we present our enhanced webinos architecture in which various cryptographic operations can be performed in the trusted environment in order to protect sensitive information such as cryptographic keys. We describe our ongoing experimental work in Section 5 and present a preliminary evaluation in Section 6. Our preliminary results are promising both in terms of the feasibility of implementing this architecture and the performance of the system.

2 webinos Architecture

Devices such as mobile phones, smart TVs, home appliances, energy meters and even cars are now capable of connecting to the Internet, leading to the so-called *Internet of things*. It is often the case that an individual will own multiple Internet-connected devices, each providing specific functionality or services. In order to take advantage of the composite set of services, there must be a mechanism for interconnecting these devices and facilitating resource sharing.

The *webinos* platform is an example of a system that achieves this objective. Based on node.js technology, the webinos platform provides an infrastructure for securely executing web applications across multiple devices. Through a set of APIs [9] such as geolocation, NFC and contacts, the webinos platform facilitates access to these services and resources by web applications. Using webinos, a device can access services and resources provided by a different device within the user's personal network (called a *Personal Zone)*. To enable this, each device runs a webinos component called a Personal Zone Proxy (PZP) and all devices in a particular zone are interconnected either through peer-to-peer communication or using a central component called the Personal Zone Hub (PZH).

There are several use cases for webinos [8] but for the purpose of this paper, we consider a specific scenario in which a user wishes to use a smart TV to watch a video stored on his or her smartphone. Figure 1 shows the overall webinos architecture relevant to this scenario.

Since the smart TV is webinos-enabled, a web application running on the TV can make the appropriate API calls to request the video from the smartphone.

[1] webinos is an EU-funded project and affiliate program aiming to define and deliver an Open Source Platform and software components for the Future Internet http://www.webinos.org

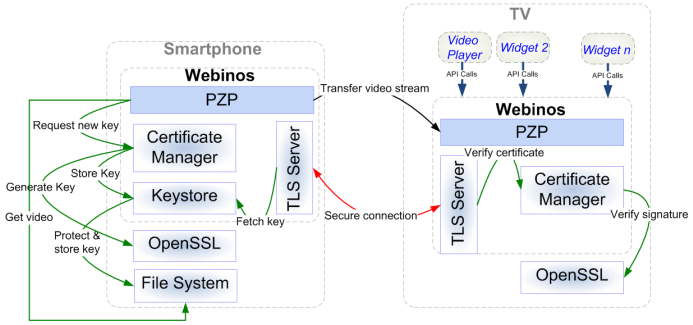

Fig. 1. Architecture of webinos — showing only the components discussed here

As part of this process, the TV must establish a secure communication link to the smartphone as shown in Figure 1. The key webinos components shown in the figure are explained below.

Certificate Manager — The certificate manager component provides functions for generating cryptographic keys and certificates for use in Transport Layer Security (TLS) connections. The implementation of this component relies on OpenSSL and runs on all webinos enabled devices. The component exposes a `genRSAKey` function, which returns either a 1024 or 2048 bit RSA keypair.

Key Store — The keys generated by the certificate manager and any passwords used in webinos are stored in a keystore. This allows for secure storage even across platform reset events. In the webinos architecture, the keystore functionality is provided using native platform mechanisms such as *gnome keyring*.

TLS Server — A TLS server is instantiated as part of webinos to support both client-server and peer-to-peer device connections. The server uses a secret key from the keystore and the cryptographic primitives provided by the underlying node.js platform which in turn uses OpenSSL to establish TLS connections.

3 Trusted Execution Environments

Various hardware-based mechanisms have been developed to provide enhanced security guarantees by building on well-established security principles such as defence in depth, least privilege and isolation. An architectural pattern that has emerged from these mechanisms is the *Trusted Execution Environment* (TEE). The fundamental concept of a TEE is that it allows specific software operations to be executed in isolation from the rest of the system. At present, the most common use case for a TEE is to provide a root of trust for other aspects of the system by performing certain security critical operations such as the management and storage of cryptographic keys in the TEE. Due to the hardware-enforced isolation from the rest of the system, it is possible to trust the software executed in the TEE without having to trust any other software on the system. In some cases, it is also possible to prove the degree of isolation to an

external entity. Various implementations of the TEE architectural pattern have been developed for different platforms by both industry and academia including the *Flicker* research project [4] and ARM TrustZone technology [1].

ARM Trustzone is a security technology that provides a hardware-enforced TEE on ARM platforms. Trustzone is implemented as a set of security extensions on certain processor cores based on the ARM version 6 or version 7 architecture [1]. As shown in Figure 2, Trustzone partitions the platform hardware into two distinct *worlds*, namely the *normal world* and the *secure world*. With the support of other TrustZone-enabled platform hardware elements, this ensures that components running in the normal world do not have access to resources belonging to the secure world. The general approach is to run a minimal secure kernel in the secure world in parallel with a feature-rich OS in the normal world.

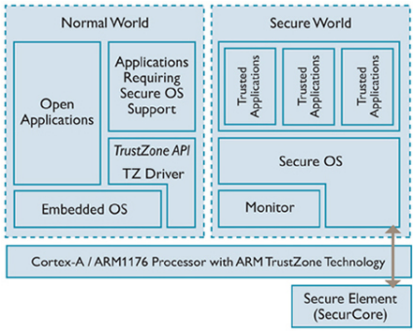

Fig. 2. ARM TrustZone Software Architecture [1]

By implementing TrustZone as a processor extension, it is possible for a single core to execute both the normal world and the secure world in a time-sliced manner thus eliminating the need for a separate security co-processor whilst still ensuring full isolation. Transitions between the normal and the secure worlds are managed by a secure software component running in a new processor mode called secure monitor mode, shown as the *Monitor* in Figure 2. Whilst TrustZone hardware is already available, the software required to use this functionality is still in a state of flux. Recently, the GlobalPlatform consortium released the GlobalPlatform TEE API Specification [2], in an effort to standardize access to TEE functionality across different devices.

4 TEE-enhanced webinos Architecture

The philosophy of enabling uniform and secure resource access across multiple devices makes webinos a suitable platform for exposing secure hardware functionality to web applications. The webinos platform already facilitates discovery and access control for cross-device resource sharing. This platform uses the device OS and native applications to expose several APIs to webinos widgets (web

applications that run on devices). These APIs are an abstraction of the resources provided by each device. We view the functionality provided by a TEE as another type of resource that could be made accessible in a similar manner. We argue that the security of web applications can be enhanced through the use of a TEE and that the webinos architecture can be extended to provide this functionality to web applications with minimal modifications. Figure 3 illustrates our proposed webinos architecture enhanced with trusted execution in which the platform is divided into an isolated *secure domain* for security-sensitive operations and a *feature-rich domain* for all other operations.

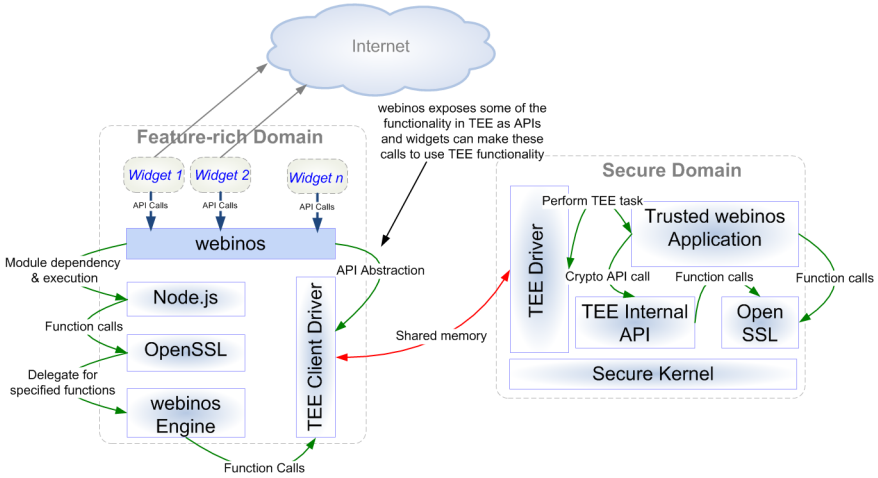

Fig. 3. An architecture of webinos enhanced with TEE

4.1 Feature-rich Domain

The feature-rich domain includes the main device OS and the majority of webinos functionality as well as any installed webinos widgets. The following components from this domain are important in our enhanced webinos architecture:

OpenSSL with Modular Engine Functionality — Since version 0.96, the OpenSSL library has been designed to support ENGINES [5] — modules that can be dynamically or statically loaded to provide alternative implementations of cryptographic functions. OpenSSL ENGINES are sometimes used for interfacing with secure hardware such as a smartcard or Trusted Platform Module (TPM)[2].

The webinos TEE OpenSSL Engine — As shown in Figure 4, the webinos TEE engine is an OpenSSL ENGINE that runs in the normal world and communicates with the secure world through the TEE client driver. This allows

[2] https://github.com/ThomasHabets/openssl-tpm-engine

specific cryptographic operations to be performed in the secure world. In particular, the webinos TEE engine provides encryption, decryption and signature operations using keys that are only accessible in the secure world. These keys are referenced using *key_id* fields as shown in Listing 1.1.

Listing 1.1. Sample of cryptographic operations provided by webinos TEE engine

```
int webinos_private_encrypt(char* key_id, ...);
int webinos_private_decrypt(char* key_id, ...);
int webinos_sign(char* key_id, ...);
```

4.2 Secure Domain

The enhanced architecture uses the principle of least privilege to isolate a secure domain from the feature-rich domain. The following aspects of the secure domain are important in our enhanced architecture:

Secure kernel — The secure kernel provides common functionality for the secure domain and ensures that all operations in this domain are completely transparent to the feature-rich domain. The secure kernel also enforces strict isolation between different trusted applications in the secure domain.

Trusted webinos application — The trusted webinos application is responsible for authenticating the source of the requests from the feature-rich domain. This could be achieved by inspecting the requests or using technology similar to the integrity measurement architecture (IMA) as described in [6]. This trusted application also provides key life-cycle management functionality. This application is based on the GlobalPlatform TEE Internal Specification [2], but can also utilize the cryptographic library included in the secure domain.

Secure domain cryptographic library — The kernel in the secure domain is built with support for cryptographic functions. This is provided by libraries such the OpenSSL or PolarSSL that have been installed in the secure domain.

5 Case Study: Securing TLS Sessions

It is informative to consider the scenario introduced in Section 2 because it involves a complete life-cycle of a cryptographic operation, which is a common scenario in most Internet-connected systems. We consider how TLS sessions are established using the enhanced webinos architecture. This process involves securely generating, storing and using cryptographic keys in TLS connections as shown in Figure 4.

In order to demonstrate the feasibility of realizing this new architecture, we have undertaken an investigation to understand how webinos makes use of cryptographic operations and we have performed a preliminary experiment in which an OpenSSL engine is modified to take advantage of functionality provided by a TEE. In this experiment we do not explicitly consider secure persistent storage of cryptographic keys. This could be achieved using functionality from the GlobalPlatform specification (e.g. `TEE_CreatePersistentObject`) [2] and supported by technology such as a TPM MOBILE also running in the TEE [7].

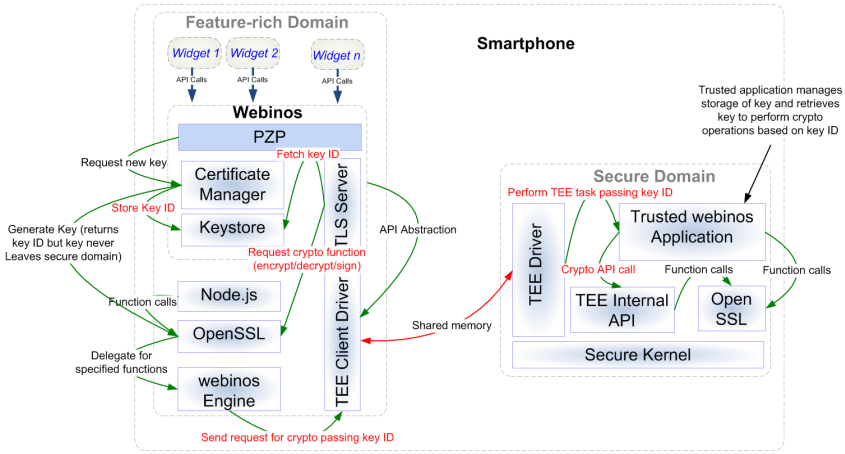

Fig. 4. The workflow of webinos when enhanced with TEE

5.1 Development Environment

The OpenVirtualization project[3] aims to create an open source software stack from TrustZone that implements the GlobalPlatform TEE API [2]. In order to accelerate the development of TrustZone software, Winter et al. [11] have created a software development and emulation framework for ARM TrustZone as part of the SEPIA research project. This framework includes a modified port of the *qemu-system-arm* processor emulator that supports TrustZone extensions for certain ARM processors and is available as an open source project called *qemu-trustzone*[4]. In order to develop a proof-of-concept implementation of our enhanced architecture, we use a combination of the OpenVirtualization software and the qemu-trustzone emulator. The OpenVirtualization project provides a software development kit (SDK) for building the normal world and secure world kernels. At present, the SDK only supports Linux 2.6.38-rc7 as the normal world kernel and so we are using a Debian "squeeze" root filesystem image for the ARM architecture. We have successfully built the OpenVirtualization images, as well as the normal world and secure world kernels and have run the system using qemu-trustzone. We have used this emulated system for our initial experiments and obtained some preliminary results as described in the following section.

6 Evaluation

In this section, we evaluate the extent to which our enhanced webinos architecture makes TEE functionality available to mobile web applications. webinos provides an abstraction layer, in the form of APIs, that enables web applications

[3] http://www.openvirtualization.org/
[4] https://github.com/jowinter/qemu-trustzone

to make use of resources and services provided by other devices. Building on this approach, there are two primary ways in which webinos can be used to bring the security benefits of a TEE to web applications: The first is to provide direct access to TEE functionality through dedicated APIs and the second is to enhance the security of current webinos APIs using TEE. An example of the first approach would involve extending the Secure Elements API in webinos [10] to allow access to TEE functionality. An example of the second approach would be to redesign an existing component such as a policy manager so security-sensitive parts of the component are executed in the secure domain. By leveraging open standards upon which webinos is built, the enhanced architecture, therefore, enables TEE functionality to be exposed to web applications in a platform neutral manner. This allows web applications to use the same interfaces to TEE functionality across different devices, leading towards cross-platform security.

Although the implementation of the architecture is ongoing, preliminary results are promising. We have successfully run webinos in the normal world OS and have proxied all cryptographic functions through the engine with minor modifications. We are able to make calls to the secure world using the TEE client library provided by the OpenVirtualization SDK and have successfully built part of the OpenSSL library into the secure world. We are currently implementing the communication interfaces between the engine the secure domain. One of our objectives is to demonstrate the feasibility of implementing this proposed architecture and, as discussed in the previous sections, this appears to be feasible and will only require minimal changes to webinos or the underlying node.js platform. Another important consideration for this kind of architecture is system performance. Using the current implementation, a preliminary performance analysis has been performed to determine how this compares to other implementations such as [6]. The overall performance of the system in [6] is limited by the performance of a separate cryptographic co-processor, the TPM, which has not been designed as a high-performance device. In [6], the overall time taken for the primary cryptographic operation (an RSA signature operation) was in the order of 1000 milliseconds but in our current experiment, preliminary tests have shown that the equivalent operation can be performed in the order of 10 milliseconds because all computations take place on the main CPU. This level of performance latency would be essentially unnoticeable in web applications because it is lower than average communication latencies over the Internet. Although this result has been obtained using the TrustZone emulation framework [11] on a different hardware architecture, we expect to demonstrate similar results on real-world hardware in the near future.

7 Conclusion

Web applications make use of resources on mobile devices such as cameras or navigation systems to create an interactive experience for users. As a result, they are becoming an attractive channel for attacks against mobile devices. Web applications can take advantage of platform features such as Trusted Execution

Environments to protect sensitive information. However, TEE features are normally provided at a low level of abstraction and so in order for these features to be useful for web applications, a platform independent and standards-based approach is essential. We have proposed, demonstrated and tested an enhanced webinos architecture augmented with ARM TrustZone technology. Although this work is ongoing, preliminary results from our experiments are promising in terms of feasibility of implementation and system performance.

Acknowledgements

The work described here is funded by the webinos project with collaboration from the SEPIA project. The authors thank Johannes Winter, Ronald Toegl and Martin Pirker for their support and insights on TrustZone technology and *qemu*. We also thank the OpenVirtualization community and support team for their assistance. Andrew Paverd is funded by the *Future Home Networks and Services* project sponsored by British Telecom.

References

1. ARM. TrustZone. Last accessed: April 2013 http://www.arm.com/products/processors/technologies/trustzone.php.
2. GlobalPlatform. TEE System Architecture v1.0; TEE Internal API Specification v1.0; TEE Client API Specification v1.0;. Technical report, 2011.
3. D. Grawrock. *Dynamics of a Trusted Platform: A Building Block Approach*. Intel Press, 2009.
4. Jonathan M. McCune, Bryan J. Parno, Adrian Perrig, Michael K. Reiter, and Hiroshi Isozaki. Flicker: an execution infrastructure for TCB minimization. In *Eurosys '08 Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems*, volume 42, page 315, April 2008.
5. OpenSSL Project. OpenSSL Engine Documentation. http://www.openssl.org/docs/crypto/engine.html Last accessed: April 2013.
6. Andrew J. Paverd and Andrew P. Martin. Hardware Security for Device Authentication in the Smart Grid. In *First Open EIT ICT Labs Workshop on Smart Grid Security - SmartGridSec12*, Berlin, Germany, 2012.
7. Trusted Computing Group. TPM MOBILE with Trusted Execution Environment for Comprehensive Mobile Device Security. Technical Report June, 2012.
8. webinos Consortium. webinos Use Cases and Scenarios - v1.0. Technical report, 2009. Last accessed: April 2013 http://www.webinos.org/content/webinos-Scenarios_and_Use_Cases_v1.pdf.
9. webinos Consortium. webinos Phase II API Specifications. Technical report, 2012. Last accessed: April 2013 http://www.webinos.org/content/webinos-phase_II_device,network,and_server-side_API_specifications.pdf.
10. webinos Consortium. webinos Secure Elements API, 2012. http://dev.webinos.org/deliverables/wp3/Deliverable34/secureelements.html Last accessed: April 2013.
11. Johannes Winter, Paul Wiegele, Martin Pirker, and Ronald Toegl. A flexible software development and emulation framework for ARM TrustZone. In *INTRUST 2011*, 2011.