# Position Paper: Can the Web Really Use Secure Hardware?

Justin King-Lacroix[1]

Department of Computer Science, University of Oxford
justin.king-lacroix@cs.ox.ac.uk

**Abstract.** The Web has become the platform of choice for providing services to users in a range of different environments. These services can have drastically differing security requirements; for more sensitive services, it is attractive to take advantage of the wide deployment of secure hardware in the consumer space. However, the applicability of that hardware to any given security problem is not always obvious. This paper explores which use cases are appropriate, and which are not.

## 1 Introduction

The Web has transformed from a medium of information exchange to a platform for providing services. As desktop computers have become more powerful, Internet connections wider, and Web browsers more capable, Web applications have evolved to the point where they can now compete with their desktop counterparts on feature-set, performance, and price. Additionally, Web applications have several advantages: for one, the Web architecture requires application code to be downloaded from the remote server on every launch, placing the burden of application installation and maintenance on server operators (who are required to be IT-savvy) rather than end-users (who are not); for another, user data is frequently also stored on the server side, making it automatically accessible from anywhere on the Internet.

However, different applications can have very different security needs: an Internet banking application might need much stronger evidence of identity to authorise a financial transaction than a social networking website to authorise a post. This specific problem has traditionally been solved with two-factor authentication: in addition to their password (*something you know*), the user also possesses a secure element (*something you have*) which can generate a one-time code [15]. These elements can also be used to show liveness – certain banks ask users to re-authenticate before allowing certain types of transactions.

Much work has been done on user authentication in general, and two-factor authentication in particular [6]. These topics will therefore not be explored further in this paper. Instead, we ask the following two questions:

1. What other security properties might Web applications require?
2. Can secure hardware practically provide those properties?

## 2 Web application security

Concerning Web applications, we divide security properties into two broad classes: those internal to the application itself, guaranteed by programmers at design time, and those regarding – and provided by – its execution environment. Naturally, secure hardware can only help with the latter class.

To aid in discussion, we further divide the application's execution environment into three sections: its hardware environment, including the physical environment of that hardware; its software environment, which includes that both server- and client-side; and its network environment, since Web applications are inherently network-based.

Finally, we remark that any hardware security device can only be used to make meaningful assertions about components of the system that explicitly use it, or are implemented in terms of other components that themselves use it. This fact is well-understood, but is restated here because it is critical to the discussion that follows.

*Hardware* The application may require knowledge of hardware present or absent at the client, or may be required to present such information about the server. More strongly, it may be required to enforce restrictions on the nature of that hardware, its location, or other properties of its physical environment (such as the availability and quality of electrical power, or level of electromagnetic interference). Finally, the application may require knowledge or restriction of the identity of the devices on which it runs. The identity of the person currently using the device (or, indeed, whether there is even a user currently present) can also be considered part of its physical environment.

*Software* The application may wish to know or restrict the operating system (OS), OS patch level, Web browser or browser version, or installed software on any given device. The configuration of any of these components is also a concern; particular examples include the presence (or absence) of browser plugins, rights of the user account under which the browser is running, or modifiability of elements of the system's configuration. System state information, such as network location and address(es), CPU and memory load, and currently-running tasks, could also be required.

Equally, the application may be required to present such information about servers to its clients, so that the clients can make security decisions regarding the transmission of sensitive information.

*Network* Traffic between client and server may have confidentiality, integrity, or authenticity constraints. It may be that all traffic must be confidential to only the communicating parties, or that this only applies to some traffic. Equally, all traffic may need to be tamperproof, or only some, or may require different levels or types of integrity protection at different times. Finally, certain data may require proof of additional verification, such as obtaining explicit user consent, before it can be considered authentic.

## 3 Secure hardware in the consumer space

Consumers already have access to a variety of secure hardware. They are accustomed to authorising financial transactions using chip-and-PIN smart credit cards, inserting SIM cards into their mobile phones, and authenticating to banking (and other) Web applications using hardware tokens [14]. However, deploying a secure element may not always be cost-effective; in any case, all of these examples use secure elements purely for user authentication.

The Trusted Platform Module [1] is a near-ubiquitous example of a hardware security module (HSM). It provides services for encryption, signing, and key storage. Moreover, the availability of cryptographic keys and decrypted plaintext can be made to depend on the software state of the system, which can equally be asserted to a remote party using *remote attestation*. [8,9] Assertions about the composition and construction of that system are made by its manufacturer(s), via the *endorsement* and *platform credentials*. Finally, the TPM also provides a cryptographically-strong device identity.

Trusted Execution Environments (TEEs) are also widely deployed, with ARM TrustZone [3] and Intel Trusted Execution Technology (TXT) [7] – both CPU architecture extensions – the most common examples. These provide for isolated code execution, which can allow for more flexible key usage policies than those available with an HSM, although TEEs based on these technologies often use the TPM for its measurement and attestation services [2,11]. In particular, the ability to execute arbitrary code could be leveraged to guarantee explicit user consent.

Notably absent from consumer devices is a secure sensor access mechanism: sensor data may be used by both Web and native applications (insofar as browsers and OSes provide useful APIs to that end), but there is currently no way of assuring the accuracy of the data therby obtained.

## 4 Secure hardware in Web applications: the difficulties

The Web is a highly heterogeneous environment: Web browsers can be found for most operating systems, on several hardware architectures, across multiple device form factors, with different interface paradigms. Ideally, the browser should abstract over these differences, presenting a uniform rendering surface for Web applications. In reality, that abstraction must leak information about its underlying layers to the application; for example, for usability reasons, the application may change its interface depending on the local form factor.

This goes double for security-related services: in order for a security service to be useful, both the service itself *and its implementation* must be trusted. Web security APIs must therefore expose low-level information to the application, in order that a meaningful trust decision can be made. Further, this information must be verifiable, which, for a service backed by secure hardware, requires the exposure of hardware-level APIs directly to the application.

This presents two problems. First, for secure hardware, or indeed security services in general, to be exposed to Web applications, either the entire software

stack above them must be trusted – which presents a verification and maintenance nightmare to application developers – or their APIs must be carefully designed not to trust said stack – which limits how they can be abstracted over, inhibiting the paradigm of generic service-provision that has contributed to the success of the Web. Second, in order to support any such security services, all browsers on all platforms must be modified to support them, an issue showcased most recently by HTML5's video codec fragmentation.

In short, either application developers must be able to make trust decisions about entire software stacks of arbitrary complexity, or browsers and operating systems must be modified to support specific security technologies, with little room for abstraction.

## 5 Secure hardware in Web applications: the possibilities

All is not lost: while secure hardware is challenging to apply to the Web space in general, it has several specific, but significant uses.

*Hardware* Secure hardware can provide very few meaningful guarantees at the hardware level, in no small part because much security hardware operates independently of other hardware in the machine.

CPU instruction set extensions are the obvious exception to this rule, and indeed they are excellent tools for isolating executing code. System devices like the IOMMU perform a similar function for hardware devices. However, neither of these tools can assist with issues related to network security: network abstractions are orthogonal to the memory isolation that they both provide.

In essence, secure hardware can make very few statements about the hardware environment in which an application is executing, other than its own presence and state. To provide any stronger guarantees, it must (like the TPM) be very closely integrated into that environment.

*Software* The TPM already provides a cryptographically-strong device-specific identity. Applications needing to limit their distribution to a small set of trusted machines thus already have a mechanism for doing so; this is supported by existing PKI infrastructure in browsers and OSes already, and so requires no modification to either. (This type of limitation is most useful for applications with designated administration nodes, whose state can be carefully maintained.)

Credential storage is another deployed use of secure hardware – the TPM and smart cards being well-known examples – that can be easily integrated into nearly any application. However, the fact that those credentials are resident in a hardware security module, and the properties of that module, *must* be known to the application in order for an informed security decision to be made. Additionally, depending on the nature of both the stored credentials and the hardware, credential revocation may be problematic.

Perhaps the most interesting application is a combination of TPM-based runtime state attestation, TXT's virtualisation-based software security, and recent

work on TPM virtualisation [4,10,13]: applications with high security demands can distribute read-only virtual machine images containing a restricted and well-understood software stack. The hardware TPM can identify the hypervisor running on the bare hardware, and the virtual TPM can then identify the software state of the virtual machine [5].

*Network* Paradoxically, by virtue of being implemented at a layer of abstraction below both the Web application and, indeed, the browser, the network environment is the easiest to augment with secure hardware. Combining TPM-based keys with TLS is already a well-understood technique; more recent work involves using a TEE to provide more sophisticated key policy options than are possible with only the TPM [12].

## 6 Conclusion

Deployed consumer security hardware is an attractive choice for fulfilling the security needs of Web applications. However, its applicability is not always clear. In general, the need to modify the Web browser and the underlying OS limits the number and scope of technologies that can be usefully employed.

However, there are two major cases in which secure hardware can be of use. The first is in securing services at a lower layer of abstraction than the Web application, and whose security is thus negotiated and provided transparently. TLS, and developments thereon, are an excellent example of this kind of security. In a similar vein are TPM-based keys and cryptographic transactions, with their integration into OS cryptographic infrastructure in a manner transparent to the browser. However, some trust decisions must be able to be made about the implementations of each layer in use; this is an open problem.

The second is in a 'pass-through' mode, where security services are explicitly exposed by all layers in between the hardware and the application. This mode is more difficult to achieve, since every intermediate layer must be partially or fully rewritten to support the change. However, it has seen limited success with two-factor authentication – where the abstraction layers between the hardware interface and the Web application are few.

In either case, however, secure hardware is limited in the guarantees that it can provide. It cannot be used to make assertions about other devices in the machine unless those devices are designed (or forced) to interact with it. Equally, it cannot be used to make assertions about high-level constructs – such as OS user account or application identity – unless the software that implements those constructs directly interacts with it.

## References


1. Trusted Platform Module main specification 1.2 part 1: Design principles. Tech. rep., Trusted Computing Group
2. GlobalPlatform TEE System Architecture. Tech. rep., GlobalPlatform Inc. (2011)

3. ARM Holdings: ARM Architecture Reference Manual
4. Cooper, A.: Towards a trusted grid architecture. Ph.D. thesis, University of Oxford (2010)
5. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: a virtual machine-based platform for trusted computing. ACM SIGOPS Operating Systems Review 37(5), 193–206 (2003)
6. Grosse, E., Upadhyay, M.: Authentication at scale. In: Proceedings of the IEEE Symposium on Security and Privacy. SP'13, vol. 11, pp. 15–22 (2013)
7. Intel Corporation: Trusted eXecution Technology (TXT) – Measured Launched Environment Developer's Guide
8. King-Lacroix, J., Martin, A.: BottleCap: a credential manager for capability systems. In: Proceedings of the 7th ACM Workshop on Scalable Trusted Computing (2012)
9. Martin, A.: The ten page introduction to trusted computing. Research Report CS-RR-08-11 (2008)
10. McCune, J.M., Li, Y., Qu, N., Zhou, Z., Datta, A., Gligor, V., Perrig, A.: TrustVisor: Efficient TCB reduction and attestation. In: Proceedings of the IEEE Symposium on Security and Privacy. pp. 143–158. SP'10 (2010)
11. McCune, J.M., Parno, B.J., Perrig, A., Reiter, M.K., Isozaki, H.: Flicker: an execution infrastructure for TCB minimization. In: Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems. pp. 315–328. Eurosys'08 (2008)
12. Paverd, A., Martin, A.: Hardware security for device authentication in the smart grid. In: First Open EIT ICT Labs Workshop on Smart Grid Security. SmartGridSec12, Berlin (2012)
13. Perez, R., Sailer, R., van Doorn, L.: vTPM: virtualizing the trusted platform module. In: Proceedings of the 15th Conference on USENIX Security (2006)
14. Vasudevan, A., Owusu, E., Zhou, Z., Newsome, J., McCune, J.: Trustworthy execution on mobile devices: What security properties can my mobile platform give me? In: Trust and Trustworthy Computing, Lecture Notes in Computer Science, vol. 7344, pp. 159–178 (2012)
15. Weir, C.S., Douglas, G., Carruthers, M., Jack, M.: User perceptions of security, convenience and usability for ebanking authentication tokens. Computers & Security 28(12), 47–62 (2009)