Scalable Trusted Computing for Webinos

Designing a cross-device mobile web application environment with trusted infrastructure components

Anonymous authors

ABSTRACT

This paper discusses the security architecture of webinos, a cross-platform web application environment. We describe our plans for introducing techniques from trusted infrastructure to webinos in order to provide assurance, protect user data and increase the system's trustworthiness. Our main contribution is a set of proposals which show that existing concepts and technologies, such as Trusted Network Connect and the Platform Trust Service, are applicable to users with multiple personal devices. We suggest that the key to making trusted computing scale to home users is the introduction of a cloud-based third party, which we call the *personal zone hub*.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Design, Security, Standardization

Keywords

Web applications, Webinos, Trusted infrastructures

1. INTRODUCTION

Home computing is undoubtedly getting more complicated. Users own more devices, use more applications and share more data – some of it both private and valuable – than ever before. They have smart phones, TVs, in-car entertainment systems and laptops, all of which run cheap, easily-accessible applications available in their thousands on application marketplaces. This is a headache for developers, who have to produce applications for many different devices and operating systems without spending money which they will struggle to recoup. As a result, they might be forgiven for overlooking security issues and relying instead on operating systems and middleware to protect their users.

STC '11 Chicago, USA

Cross-device application platforms, such as *webinos*, are being developed to help solve this problem. Webinos provides a set of standard APIs for web applications and aims to provide the necessary access controls to secure user data and devices. The challenges are great: there are thousands of mobile applications available, many of which are malicious and may try to steal data or use the platform to launch denial of service attacks. Furthermore, the threats are more severe than for normal programs, as cross-device applications are increasingly connected, so the compromise of an application on a user's PC could potentially result in the subsequent compromise of their mobile phone. Applications are also often developed using *web technologies*, bringing many of the security and privacy problems associated with the web. This is therefore an area in which techniques from *trusted computing* might be productively used.

However, trusted computing techniques have often been developed primarily with corporate scenarios in mind [8]. Home users cannot benefit as they do not have their own IT department to manage the required support infrastructure. To address this problem, in this paper we identify how trusted computing specifications and systems can be applied to the emerging webinos security architecture. We attempt to mitigate some of the threats associated with a cross-device web application environment without imposing too large a burden on end users. Our major novel contribution in this paper is identifying that many trusted computing concepts and techniques are applicable to multi-device home user systems which are beginning to resemble corporate networks.

This paper is structured as follows. In section 2 we introduce the webinos project and give background information on trusted infrastructures. In section 3 we look at related systems to webinos. We describe the current webinos security architecture in detail in section 4 and in section 5 we outline our proposals to integrate trusted computing. In section 6 we conclude.

2. BACKGROUND

2.1 The webinos project

Webinos is a large EU project which aims to create a standardised cross-device web application environment. Web applications (or *widgets*) are essentially the same as websites, consisting of HTML, CSS, JavaScript, images and other media, and can be packaged into zipped archives and run locally, with access to device features such as cameras and sensors. Webinos will be implemented on at least four different classes of device: mobile phones, PCs, set-top boxes

and in-car entertainment systems. The main aims of the project are to provide:

- a seamless user experience on all personal devices;
- usable security and privacy for personal data;
- a standard set of APIs for developers accessing device features; and
- an open source implementation on multiple platforms.

Webinos will initially be implemented as a browser plugin, providing these capabilities to all compliant web applications. A wide range of applications are possible, such as games, location services, and social media utilities. This also includes applications which use personal or valuable data such as email, e-health, and mobile payment. Webinos security is therefore a priority.

The webinos architecture is pictured in Figure 1. All of a user's personal devices exist in the user's *personal zone*. Devices run web applications in a modified browser which is part of the webinos runtime. Assuming permission has been granted, applications can access webinos device APIs and local context data. These APIs also provide a mechanisms for communicating with other devices in the personal zone, so that a user's TV can automatically display photos taken from the user's smartphone. Each personal zone has a master device – the *personal zone hub* (PZH) – which is responsible for device discovery, routing communication within the zone and connecting devices in different zones. We expect this to be either a cloud-hosted virtual device, or functionality provided by a home router. The PZH synchronises user data between devices and provides mechanisms for remote management. All other webinos-enabled devices contain a *personal zone proxy* (PZP) which is responsible for communicating with the personal zone hub. It is also the trusted component on the platform, storing access control policies and certificates. If a personal zone hub is inaccessible, perhaps due to loss of network access, two webinos devices can still discover and communicate with each other using another local communication medium (e.g. Bluetooth). In this situation, the personal zone proxy will temporarily take on some of the functionality of a personal zone hub. When webinos is first installed on a device, the installation process will involve registration with the personal zone hub, which will add it to its list of known devices.

Personal zones are initially aimed at including one users' devices, with shared devices connecting to different personal zones depending on the current user. However, a personal zone might also be used in corporate settings to contain all company devices. Further elaboration of personal zones is expected to allow for more complex scenarios.

2.2 Trusted Computing

Trusted computing is a paradigm developed and standardised by the Trusted Computing Group [11]. It aims to enforce trustworthy behaviour of computing platforms by identifying a complete 'chain of trust', a list of all hardware and software that has been used. If a platform owner can reliably find out exactly what software and hardware is in use, they should be able to recognise and eliminate any malware, viruses and trojans. A great deal of infrastructure is required to make this idea practical, including new hardware, modifications to applications and databases of known, trustworthy platform configurations.

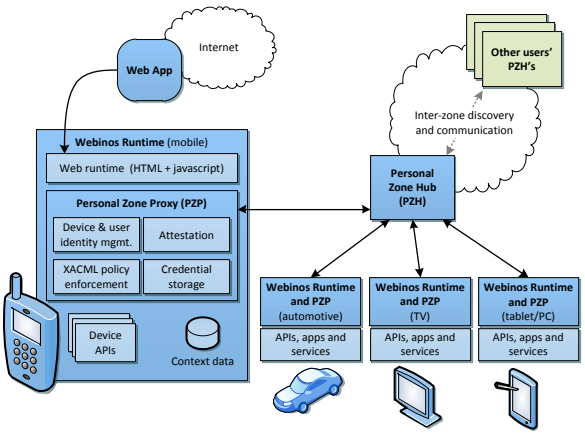

Figure 1: Overview of the webinos architecture

The technologies proposed by the TCG are centred on the Trusted Platform Module (TPM). In a basic server implementation, the TPM is a chip connected to the CPU. It provides isolated storage of RSA keys and Platform Configuration Registers (PCRs). PCRs can be used to hold *integrity measurements*, in the form of 20 byte SHA-1 hashes. PCRs cannot be written do directly, but their content can be modified by taking the current register value to the supplied input, hashing it, and stores the result in the PCR. A PCR value therefore reflects a *list* of hashes. In order to work out what individual inputs have been added to a PCR, a separate log must be kept. When this log is replayed, by rehashing every entry in order, that final result should match the PCR value.

The limited functionality offered by the TPM is ideal for recording the boot process of a platform. The idea being that, starting from the BIOS, every piece of code to be executed is first hashed and extended ('measured') into a PCR by the preceding piece of code. As a result, no program can be executed before being measured, and because the PCRs cannot be erased, this means that no program can conceal its execution from the TPM. A platform is said to support *authenticated boot* when it follows this process as it provides a way for the platform's boot process to be authenticated against reference values at a later time.

2.2.1 Remote attestation

In order for users to assess a remote machine, the TPM supports *remote attestation*, a process that allows a platform to report the integrity measurements collected during authenticated boot. When challenged, the TPM can create a signed copy of its PCR values. This is given to the challenger for inspection, along with the measurement log. PCRs are signed using a private key held by the TPM, guaranteeing the key's confidentiality. This is called an Attestation Identity Key (AIK) and the public half must be certified by a certificate authority (a 'Privacy CA'). Full details are on the TCG website [11].

The software running at the platform can be identified by matching the hash values in the attestation with reference data. This requires a list of *reference integrity measurements* (RIMs) contained within a reference manifest database [9].

In addition to attesting static platform configuration data,

TCG specifications also describe a *Platform Trust Service* (PTS), which is a trusted monitoring application installed on the endpoint. It is included in the chain of trust and can report on the runtime status of the platform, including the contents of configuration files or memory.

2.2.2 Trusted Network Connect

Trusted Network Connect (TNC) is a standard proposed by the Trusted Computing Group to specify how network infrastructures should communicate to protect endpoints and prevent the spread of malware [10]. It provides specifications for functionality to support auditing and access control based on platform integrity information and user authentication.

The TNC *Metadata Access Point* (MAP) server is a databases of diverse information about each endpoint device, including user authentication status, MAC addresses, policy compliance status, platform configuration, and so on. MAP servers can communicate with policy decision points in order to identify whether a certain endpoints should be granted access to the network. MAP clients (such as intrusion detection systems) can communicate with MAP servers to either consume or publish information about the state of an endpoint. TNC specifications have been designed to work with platforms with TPMs, so that integrity verification is integrated into the architecture.

TNC has also been designed to support isolation and remediation of endpoint platforms which fail in compliance checks.

3. RELATED SYSTEMS

In this section we discuss related application environments and systems developed with security as a priority feature. In particular, we look at two specific projects: Meego and Chromium.

3.1 Meego

Meego is a Linux-based operating system and software platform for native applications which provides a 'uniform set of APIs based on Qt' [7]. It is designed for a similar range of devices to webinos, including netbooks, handsets, in-vehicle systems, smart TVs and media phones.

Meego aims to protect the owner of a platform from unauthorised use of their personal data and damage to the device. Meego uses many trusted computing concepts to provide an updated 'Mobile Simplified Security Framework' (MSSF). The MSSF includes the use of a Trusted Execution Environment — a set of trusted services implemented in tamper-resistant hardware, such as a TPM. It also assumes that a secure (or measured) boot process will be followed where possible and that devices will be provisioned with various public keys for attestation, unique identification and and local cryptographic operations. The MSSF describes a secure distribution model for application to guarantee the authenticity of software sources.

Meego is a complete platform including an operating system, and takes advantage of this position to introduce a comprehensive security architecture at both a low and high level. Webinos does not have this option: it runs as an application and cannot make guarantees about the rest of the platform.

3.2 The Chromium Project

Chromium [4] and Chromium OS [3] are Google's open source browser and operating system designed specifically with web applications in mind. While the Chromium Project remains focused on PCs rather than other platforms, it has comparable security issues to webinos and Meego. The browser provides sandboxing and automatic updates to avoid runtime attacks, and browser plugins can have whitelists to only allow their use by trusted domains. Chromium OS is a lightweight Linux-based operating system which supports verified boot using custom firmware as the root of trust. The combination of the firmware and an on-demand filesystem hashing approach means that Chromium OS can potentially avoid persistent malware from being installed on the platform. It does not provide any attestation capability but does support encrypted storage. It provides a 'multi-tiered sandboxing strategy' for web apps.

Similarly to Meego, Chromium OS has the advantage of providing a complete platform for securing applications. Chrome OS is more comparable to webinos, but does not attempt to support the same set of functionality.

4. THE WEBINOS SECURITY ARCHITECTURE

4.1 Threat model and security goals

Webinos was designed with security and privacy as primary goals. For that reason, we have developed sets of domain models, misuse cases, personas and *attacker personas* to guide the specification of our security architecture as described in webinos deliverables [14, 15]. The following example scenarios show threats that are particularly important.

- An untrusted web application gains control of the platform and installs malware, with the aim of harvesting user data or setting up a botnet.
- Alice gives Bob's phone permission to access the hard drive on Alice's home PC. However, Bob's phone contains malware which uses its privileged access to infect Alice's PC.
- Ivan, an acquaintance of Alice, wants to spy on her device to gain access to personal data. He does so by registering one of his devices with her personal zone hub and synchronising with her data store.
- A trusted application is attacked at runtime (via a web content-injection attack or otherwise) and reveals user data or allows the execution of malicious code.
- Software running outside of webinos (e.g. other software on the platform) gains unauthorised access to stored webinos user and application data.

While we have considered hardware attacks as part of our security architecture in webinos, we do not focus on them in this paper.

4.2 Challenges

The webinos project faces a number of challenges in providing effective security. Firstly, webinos cannot dictate requirements on the operating system or boot process. This eliminates the option of designing an authenticated boot

system, for example. However, such functionality may be available in some user devices, such as on mobile phones, and webinos should take advantage of it where possible. As a result, we have focused on the use of security standards which are not platform specific but can incorporate trusted hardware.

Indeed, the second challenge facing webinos security is the variety of different devices which it must support. Each device may have different software configurations and security capabilities, and webinos must mitigate the possibility of one weak device resulting in the compromise of the entire personal zone. The range of devices also means that any use of attestation is likely to scale badly: any trusting party will struggle to know what a trustworthy software configuration is on every mobile or TV platform.

Performance and battery-life is also limited. This means that constant security evaluation and communication will not be possible on some devices, and that some trusted infrastructure approaches, such as virtualisation, may be impractical.

Finally, we have to consider usability issues. The most effective privacy and security controls are likely to be those which are easy to use and aligned to the task that the user is trying to carry out. We do not look at usability issues in this paper, but have followed various usability design methodologies throughout the project [14, 15].

4.3 Phase one security architecture

The first iteration of the webinos security framework [16] provides the following basic functionality, many parts of which are based on previous specifications from the Wholesale Application Community [12].

Each device has an identity key and certificate issued by the personal zone hub, asserting its membership of the personal zone. The PZH maintains a list of the certificates and synchronises them with all personal zone devices. The personal zone hub is capable of revoking devices by deleting them (or marking them as invalid) and synchronising with each device. The identity key is used to establish a long-lived TLS connection between the personal zone hub and personal zone proxy.

The personal zone proxy installed on every webinos device is considered the trusted computing base (along with the underlying kernel, shared libraries and other trusted software). It provides a policy enforcement mechanism which mediates all access to any device APIs, including local storage, location data and personal zone network access. This policy enforcement mechanism is based on XACML. Policies are stored locally but synchronised by the personal zone hub.

Applications are signed and come with certificates. If the certificates are not signed by trusted entities (defined as those with certificates stored in the PZP) their installation may be refused, depending on device settings. They may also be self-signed, again depending on device settings. Applications are updatable, as per the W3C Widget Update Working Draft [5]. The personal zone hub constantly checks that installed applications are up-to-date, so that individual devices do not need to. This information is frequently synchronised with PZPs. The hub can do the same with certificates for devices in other personal zones, as well as remote services and websites.

An attestation API is exposed to appropriately-privileged applications. This allows them to chose which hardware configurations they believe are acceptable, which may be useful for high-value functionality like payment systems. The API was designed to be entirely general and not specify a particular attestation protocol or root of trust. Currently the use of the attestation API is impractical in webinos: we cannot expect most application providers to know software configurations. However, it may prove useful in corporate scenarios where the company IT department creates an application and configures the platform.

Finally, the current security architecture specifies that encrypted storage is provided for both the webinos personal zone proxy component and (when requested) applications. This is to avoid loss of personal and valuable data when a device is lost.

This initial architecture outlines the bare minimum of protection for user devices running webinos. It uses existing techniques such as policy-driven access control, attestation and public key cryptography to protect the runtime from impersonation and to protect data from theft or eavesdropping. However, it fails in many places. The attestation API is impractical in general, so attacks from software on the platform will be difficult to identify and avoid. The lack of uniform hardware means that different devices will provide different levels of protection for storage and process isolation. This means that the webinos architecture is inevitably only as strong as the weakest device. Furthermore, there is little functionality available to provide assurance for other users – if Alice's PC wants to share data with Bob's phone, which is in a different personal zone, it is currently impossible for Alice and Bob to know that each others' devices are trustworthy. Furthermore, application developers have little assurance of their intellectual property, as no digital rights management is available for the application JavaScript source code.

The next section discusses planned improvements to this security architecture by making further use of the personal zone concept and introducing *trusted infrastructure* components.

5. CREATING A TRUSTED INFRASTRUCTURE FOR WEBINOS

The next iteration of the webinos security architecture aims to make further use of the personal zone concept – with the federated per-user personal zone hub – in order to provide usable trusted infrastructure capabilities. We do not provide implementation details due to the early stage of development, but describe how these capabilities can be used in this novel scenario.

5.1 Integrating a Platform Trust Service

The personal zone proxy can act as (or communicate with) a local Platform Trust Service (PTS) [9] for each device. The PTS is a trusted component and monitors the state of the platform. It will be customised to monitor which applications are running webinos, the system settings, and identify the other pieces of software running on the platform. Where the platform allows, the PTS will be able to attest to its integrity through a Trusted Platform Module or similar technology. We imagine this will be the case in most situations, and that a client-side PTS as part of the personal zone proxy will regularly attest to a server. Where this is not possible, the PTS will have to rely on operating system security and

will have a much less reliable root of trust.

The personal zone hub will act as the verifying party and have a server-side PTS component. It will be able to validate reports by the client-side PTS on each device and request further information. The hub is already responsible for collecting updates to webinos software and applications, and is well positioned to store sets of reference integrity measurements (RIMs) to validate attestation of these components on each device.

Lower-level components such as operating systems and the boot-chain are harder for the PZH to keep track of, as they will vary from each device. However, the PZH can register attested values when each device is first registered. When these values change, there are many options. One is to prompt the user for confirmation, but this might result in poor usability and could be a disincentive for users to update their devices. Instead, the hub can register the change in configuration and attempt to search for updates that confirm it, either from known software repositories [8] or from personal zone hubs belonging to other users. The PZH can also connect to vulnerability databases similarly to proposals by [17] and invalidate reference measurements where appropriate.

In a context where software configurations are better known or need to be restricted, the PZH can use any change in attestation data to temporarily remove the device from the personal zone. This would prevent malware on one device from affecting the whole personal zone.

5.2 Privacy CA

Assuming many of the platforms do use Trusted Platform Modules, or equivalents, there is a question of where the certificate authority for issuing AIKs will exist. This could be provided by a third party. However, for intra-zone attestation requirements (as discussed above), the PCA could reasonably exist on the personal zone hub. There are no privacy concerns, as the hub is already aware of identities. For attestation outside of a personal zone this may not be sufficient, but arguably the user's PZH might be considered a trusted authority should the main threat be that of malicious software running on the user's device rather than malicious hardware.

The personal zone hub can also take advantage of the strong identities offered by the TPM in order to pair the device to its personal zone. When a user device initially registers with the personal zone hub, it can request an AIK certificate from the PZH. The PZH will then record the device's identity and AIK for subsequent communication. Temporarily removing a device from the personal zone would involve invalidating the known AIK. Permanent removal would involve require invalidating the TPM's endorsement key, so that the device cannot attempt to re-register.

5.3 Trusted Network Connect

The next step in using trusted infrastructure concepts in the webinos architecture is to integrate Trusted Network Connect functionality. We intend to place a MAP database and server component in the PZH in order to collect data about the status of each device in the personal zone. This will include authentication data – where and how the user has logged in – as well as details about the identity of the platform (links to hardware identities such as MAC addresses and hard drive serial number) which are particularly useful for devices without trusted hardware, device capabilities and state information such as the identity and version of each piece of running software.

MAP data can be used for a number of purposes within a personal zone. PZPs can subscribe to MAP data in order to use authentication information, which can identify where the user is, and may prevent suspicious login from unusual locations. Applications may also want to query MAP data to work out whether the user needs to re-authenticate for a particular action. Details on platform capabilities and available software may also be useful when working out where certain valuable or data should be synchronised.

MAP data is more useful, however, when devices communicate between zones. *Federated* TNC could be implemented to allow the personal zone hub of one user to decided about the security state of a foreign device. For example, if Alice's phone tries to connect to Bob's PC, in order to access some multimedia, Bob's PZH can use assertions about Alice's phone made by Alice's PZH to identify whether it should be trusted. This can happen regardless of whether Alice's phone has communicated with Alice's PZH already. The result of this additional check could require Alice to re-authenticate, or update her software on her phone, before being allowed access to Bob's PC. The advantage of this system is that it is unrealistic to expect every personal zone to assess the risk of other users' devices, but it is plausible that each zone hub could assess the risk of its own devices and communicate this information. We note the similarity to 'secure message routers' proposed by Watanabe et al. [13].

5.4 PZH assurance and attestation

The systems discussed in the previous sections rely heavily on the trustworthy nature of the personal zone hub. Were it to be compromised, the privacy and security implications are severe: rogue devices could be added to personal zones, policies can be modified and user identities and data stolen. Therefore, further attention must be paid to protecting the personal zone hub and providing assurance that it is behaving in a trustworthy manner.

We imagine that the personal zone hub will be implemented in one of two possible locations: either on a cloud, supported by a service provider such as the user's mobile phone operator or identity provider, or on the user's home network router. We initially consider just the cloud hosted option.

To provide assurance to end users, PZHs can attest their software configurations to each device in the personal zone. The PZH is likely to have a well-known software identity – the hash of the virtual appliance – and therefore be practically capable of both sealing any confidential data to its platform configuration and attesting its software state to users. The management overhead should be small: if the hub is hosted by a trusted party, this party can issue software certificates. It can also issue new reference measurements for any updates to the hub before they occur, and therefore user devices can be assured of the integrity of any upgrades. Should the hub fail an attestation challenge, users should consider contacting their service provider or moving the hub virtual instance to a different location. Details of remediation are left as future work, but are vital in providing usable assurance.

Personal zone hubs could attest to one another when inter-zone communication occurs. This is necessary when relying

on the other hub to provide information about the security state of foreign devices, as suggested in the previous section. An elaboration of this idea would be to create networks of attesting PZHs so that runtime attacks could be identified and avoided without user interaction.

Finally, one of the PZH's requirements is that it is capable of revoking user devices from a personal zone in situations where the device is lost or stolen. This means that it must have a user interface for device management. Our primary concern, therefore, is that users are properly authenticated before accessing this interface and that attackers cannot easily add or remove devices from the zone. While trusted computing cannot necessarily help, two-factor authentication via another personal zone device (or a dedicated secure token) would mitigate the problem to some extent.

5.5 Further improvements

There are several further opportunities to improve the webinos security architecture with trusted infrastructure components. So far, we have assumed the existence of a secure and trustworthy operating system beneath the webinos system. We cannot provide this capability due to the scope of the webinos system. However, it may be possible to take advantage of Late Launch [6] functionality to provide some secure features even without a trustworthy operating system. This might include initial device registration, or reporting by the Platform Trust Service. We intend to investigate work by Bugiel and Ekberg [2] as a potential solution.

We have also not looked in detail at web-specific threats, such as those related to content injection and request forgery. These can exploit otherwise trusted applications and send private data to unauthorised parties. Part of the solution to is to make sure that applications only communicate with trusted sites, through client-side access control implemented via the Widget Access Request Policy (WARP) [1]. The next step is to integrate attestation and assurance into these systems, so that web servers which have been exploited by malware (or insider attacks) cannot take advantage of their trusted status.

6. CONCLUSION

We have introduced webinos, a cross-device web application runtime environment. We identified unique challenges facing the project and proposed several ways in which trusted infrastructure can help provide security.

Our main contribution is proposing that trusted computing functionality may be applied in a personal setting, rather than in a corporate network. This is due to the concept of a *personal zone* of user devices and the introduction of a per-user *personal zone hub*, which can reduce the overheads of securing multiple devices. The proposals allows for the integration of a Privacy Certificate Authority, Platform Trust Service and Federated Trusted Network Connect capabilities. These allow for additional security both within and between personal zones. We believe this new approach will scale well, due to its inherently federated nature, and that by pushing functionality to the cloud-based personal zone hub we may alleviate some of the load on power and bandwidth-constrained mobile devices.

For future work we will be implementing the security architecture discussed in this paper.

7. ACKNOWLEDGMENTS

The research described in this paper was funded by EU FP7 *webinos* Project (FP7-ICT-2009-5 Objective 1.2).

8. REFERENCES

[1] R. Berjon. Widget Access Request Policy: W3C Candidate Recommendation. `http://www.w3.org/TR/widgets-access/`, April 2010.

[2] S. Bugiel and J.-E. Ekberg. Implementing an application-specific credential platform using late-launched mobile trusted module. In *Proceedings STC '10*, pages 21–30. ACM, 2010.

[3] Chrome OS security overview. `http://dev.chromium.org/chromium-os/chromiumos-design-docs/security-overview`, 2011.

[4] Chromium browser security. `http://dev.chromium.org/Home/chromium-security`, July 2011.

[5] M. Cáceres, R. Tibbett, and R. Berjon. Widget Updates: W3C Working Draft. `http://www.w3.org/TR/widgets-updates/`, September 2010.

[6] D. Grawrock. *Dynamics of a Trusted Platform.* Intel Press, February 2009.

[7] Meego security architecture. `http://wiki.meego.com/Security/Architecture`, June 2011.

[8] R. Sailer, T. Jaeger, X. Zhang, and L. van Doorn. Attestation-based policy enforcement for remote access. In *Proceedings of CCS '04*, pages 308–317. ACM, 2004.

[9] The Trusted Computing Group. TCG Infrastructure Working Group Architecture Part II - Integrity Management, November 2006.

[10] The Trusted Computing Group. TNC Architecture for Interoperability Specification Version 1.4, May 2009.

[11] The Trusted Computing Group. Website, 2009.

[12] WAC 2.0 Core Specification: Widget Security and Privacy. `http://www.wacapps.net/web/portal/wac-2.0-spec`, January 2011.

[13] Y. Watanabe, S. Yoshihama, T. Mishina, M. Kudo, and H. Maruyama. Bridging the Gap Between Inter-communication Boundary and Internal Trusted Components. In *Proceedings of ESORICS '06*, volume 4189 of *LNCS*, pages 65–80. Springer, Sept. 2006.

[14] Webinos project deliverable: User expectations on security and privacy. `http://webinos.org/content/webinos-User_Expectations_on_Security_and_Privacy_v1.pdf`, February 2011.

[15] Webinos project deliverable: User expectations on security and privacy (updates). To appear at `http://webinos.org`, September 2011.

[16] Webinos project deliverable: Phase 1 security framework. To appear at `http://webinos.org`, July 2011.

[17] S. Yoshihama, T. Ebringer, M. Nakamura, S. Munetoh, and H. Maruyama. WS-attestation: efficient and fine-grained remote attestation on Web services. In *Proceedings of IEEE ICWS '05.*, pages 743–750, July 2005.