Trusted Mobile Computing

Ramón Cáceres and Reiner Sailer

IBM T. J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532, USA

{caceres, sailer}@us.ibm.com

Abstract. Mobility leads to unplanned interactions between computer systems as people use devices to access services in varied environments. Before two systems agree to interact, they must trust that each will satisfy the security and privacy requirements of the other. In this paper we introduce *trust overlays*, a systematic approach to building such trust. Our solution exploits the increasing availability of trusted computing hardware on commodity systems, including portable computers. We report that key pieces of this solution are coming into place, for example systems that provide distributed mandatory access control. We also point out that difficult challenges remain, for example how to set compatible security policies across administrative domains.

1. Introduction

In the context of computer systems, we can informally define *trust* as the expectation that a system will behave in a particular manner for a specific purpose [21]. Mobile computing presents many scenarios that require mutual trust between mobile devices and infrastructure systems.

For example, with SoulPad [3], the user carries an auto-configuring operating system (OS) and a suspended virtual machine (VM) on a portable device. When the user connects the device to a host PC, the PC boots the OS from the device and resumes the VM. In this scenario, the device must trust that the PC is not running additional software that will compromise the user's privacy. For instance, a VM environment on the PC could fool the OS into thinking it is booting on a bare physical machine, when in reality it is booting on a VM that can snoop on the user's data. At the same time, the PC must trust that the OS and VM it obtains from the device will not harm the infrastructure, say by launching a denial of service attack from the PC. Internet Suspend/Resume [10] introduces similar concerns, as it involves a host PC loading a user's VM from a mobile device or remote server.

There are many other more commonplace situations, for example downloading software or other digital content to a personal device from a public server; using a personal device to purchase goods or services from a public server; using a public PC to check personal mail stored on a remote server; and so on. The Trusted Computing Group (TCG) has identified similar scenarios that focus on the need for the infrastructure to establish trust on the mobile device [21]. We feel it is equally important for the mobile device to establish trust in the infrastructure.

Today the prevailing way to establish trust between systems is to exchange and verify cryptographic certificates via the Secure Sockets Layer protocol (SSL) [15]. Certificates verify the identities of communicating parties by proving the origin of data. However, they do not guarantee any system properties such as software integrity. It is common knowledge that various forms of malware (viruses, worms, etc.) tamper with software on large numbers of personal computers and servers on a daily basis. In addition, there are increasingly frequent reports of malware being developed for smart phones and other mobile devices, including a virus that can jump from a phone to infect a PC [22]. A system thus compromised can present a valid SSL certificate and yet behave maliciously.

We propose a more comprehensive solution to trust establishment based on *trust overlays*. As shown in Fig. 1, a trust overlay spans multiple systems connected via untrusted networks. On those systems that are members of an overlay, our solution verifies software integrity, enforces isolation between workloads, and secures communication. We build trust bottom-up by starting with trusted hardware and adding layers of trusted software. It is a system-level solution available to all applications running on the member platforms. An important goal is to reduce the security burden on applications in order to simplify application programming.

This paper makes two contributions. One, it identifies security and privacy properties required to establish trust across systems. Two, it describes how to provide such properties using a combination of technologies, some of which are established, some of which are emerging, and some of which require further work.

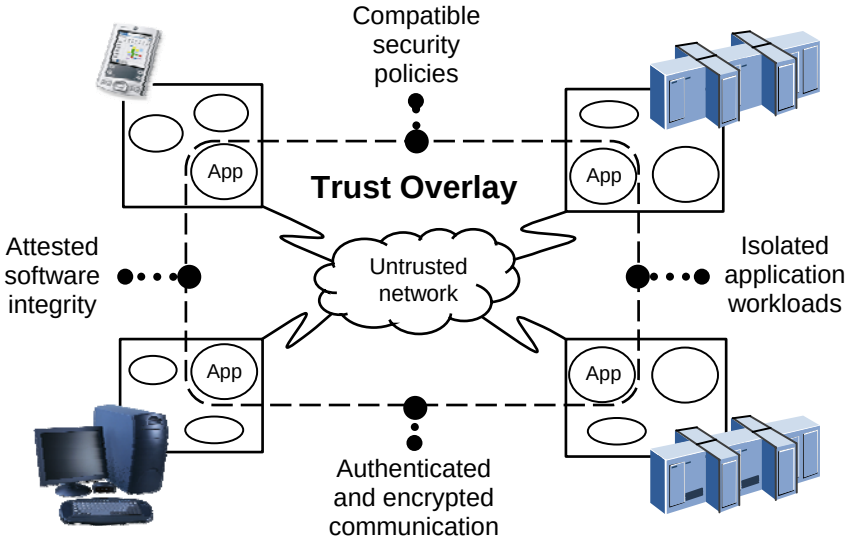

Fig. 1. A trust overlay provides security and privacy properties that span networked systems, including mobile devices, proxies, and servers.

2. Properties and Components of Trust Overlays

Fig. 1 shows an example of a trust overlay spanning four systems: a mobile device and three stationary systems. The stationary systems represent proxies and servers. Proxies offload computation and communication from resource-limited devices; they often act as intermediaries between devices and servers. All the systems communicate over an untrusted network such as the public Internet.

The purpose of a trust overlay is to provide four security and privacy properties: attested software integrity, isolated application workloads, authenticated and encrypted communication, and compatible security policies. The rest of this section discusses these properties together with the hardware and software components necessary to implement them.

2.1. Software integrity

To establish mutual trust, systems must prove their integrity to each other through a process called *attestation.* Attestation allows a remote party to verify that the software stack running on a system is the one expected and has not been tampered with. Secure attestation is made possible by cryptographic hardware that is resistant to software attacks.

An example of such hardware is the Trusted Platform Module (TPM) [21]. The TPM specification is an open standard. TPM chips are widely deployed on laptop and desktop PCs, and are becoming increasingly available on server-class machines. An effort is underway to produce a similar specification tailored to the constraints of small mobile devices. We can expect TPM-like hardware for such devices in the near future.

TPM enables secure attestation by providing secure storage as well as cryptographic primitives like hashes and signatures. Attestation typically works bottom-up through the software stack by having each level *measure* the next higher level and store the result in the TPM. A common measurement is to compute a hash of a software component just before it is loaded for execution. For example, on a standard PC, the BIOS would measure the boot loader, which would measure the operating system kernel, which would measure applications. At any point the TPM chip can be requested to produce a signed message containing the measurement results so far.

A number of TPM-based attestation schemes have been developed. Trusted Platform on Demand (TPoD) implements a trusted boot sequence by attesting to BIOS and GRUB boot loader integrity [12]. The Integrity Measurement Architecture (IMA) extends the trust chain established by TPoD by attesting to the load-time integrity of the Linux OS and its applications [17].

Fig. 2 shows a general representation of the layers used for attestation. Concrete examples for each layer are:

- *Root of Trust:* TPM or a secure coprocessor like the IBM 4758 and 4764 [4].
- *Supervisor:* Linux operating system or Xen virtual machine monitor [5].
- *Container:* Java virtual machine or Xen virtual machine.

Fig. 2 also depicts a trust overlay containing a mobile device and a server that have attested their integrity to each other. In deference to the resource limitations of mobile devices, the device is shown to run a simpler software stack, perhaps a Symbian OS supporting one Java VM and application workload at a time. In contrast, the server is shown to run a more complex stack, perhaps a Xen hypervisor supporting multiple Xen VMs, each running a different OS and workload.

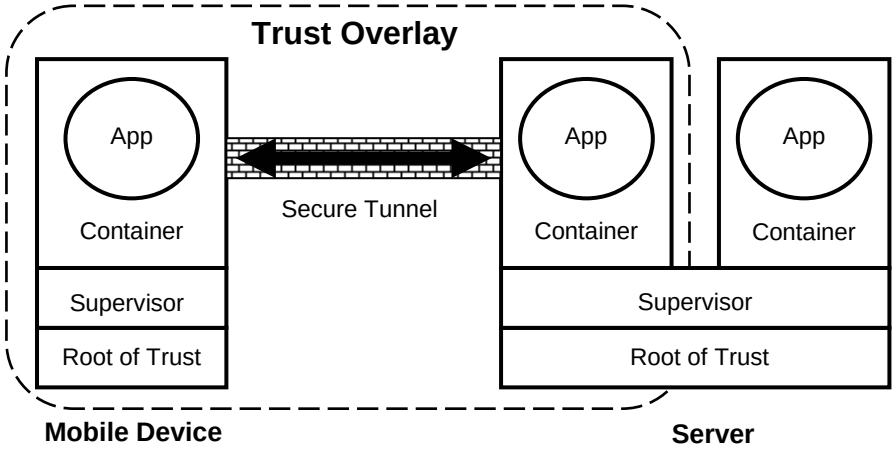

Fig. 2. Layers of trusted hardware and software combine to enforce software integrity, workload isolation, and secure communication.

2.2. Workload isolation

Attested software integrity is necessary but not sufficient for establishing distributed trust. Many usage scenarios also place restrictions on information flow that cannot be left to applications to enforce. A comprehensive solution to the distributed trust problem must provide system-level guarantees on isolation of application workloads.

Mandatory access control (MAC) has proven to be an effective mechanism for making such guarantees [2]. MAC policies ensure that system security goals are achieved regardless of user action, in contrast with discretionary policies that let users grant rights to the objects they own. Security-Enhanced Linux (SELinux) adds MAC to the Linux kernel in order to control resource access by application processes [14]. The sHype security architecture adds MAC to hypervisors like Xen in order to control resource access by virtual machines [18].

We believe that virtual-machine environments augmented with mandatory access control are an ideal platform for providing the workload isolation we seek for trust overlays. VM monitors have naturally good isolation properties because they mediate VM access to all physical resources. The addition of MAC further allows us to reason formally about the correctness of information flow within the system.

To continue with the example in Fig. 2, the server could offer strong isolation guarantees by using the following software stack:

- *Supervisor:* Xen hypervisor with sHype security.
- *Container:* Xen virtual machine running SELinux.

The mobile device could offer more moderate guarantees by using this stack:

- *Supervisor:* Linux, Palm OS, Symbian, or Windows Mobile.
- *Container:* Java virtual machine with Java 2 Platform Security [20].

Isolation guarantees on mobile devices would be strengthened by the adoption of operating systems with mandatory access control. Possibilities include a stripped-down version of SELinux or a new operating system designed with this requirement in mind.

2.3. Secure communication

Another piece of the trust overlay picture is secure communication between the overlay members. Authentication and encryption are necessary to work over untrusted networks like the Internet. Establishing such communication is a solved problem with two well-known solutions:

- Internet Protocol Security (IPSec) [11].
- Secure Sockets Layer (SSL) [15].

Either of these solutions can be used to implement the *Secure Tunnel* between the mobile device and server shown in Fig. 2.

The operation of these secure tunnels needs to be integrated with the other aspects of trust overlays. For example, a tunnel must not be established if either attestation fails or communication between the endpoints is forbidden by the isolation requirements.

2.4. Compatible policies

The final aspect of trust overlays involves setting compatible security policies across all the systems in an overlay. The world at large is heterogeneous, with many different and sometimes competing administrative domains, particularly in the mobile computing context. It is not enough to set a *common* security policy, such as may be in force within a single administrative domain. What is needed is a way to negotiate and enforce different but *compatible* policies across administrative domains.

This is a difficult open problem. However, there is a great deal of activity around policy management throughout the security and privacy research community. We have started work in this area and plan to contribute to a solution.

3. Status and Future Work

Together with colleagues at the IBM T. J. Watson Research Center, we have prototyped on stationary computers the first three aspects of trust overlays described in the previous section. We have created a distributed mandatory access control system [13] that verifies software integrity, provides workload isolation, and establishes secure communication. The prototype uses the Trusted Platform Module as a root of trust, the Xen hypervisor with sHype security as a supervisor, and Xen virtual machines as containers. We have used this system to establish trust between the distributed components of a Berkeley Open Infrastructure for Network Computing (BOINC) [1] application.

Work remains to extend this distributed MAC system to mobile computing environments. However, we do not see any fundamental obstacles to applying our trust overlay concepts in such environments. For example, the TCG Mobile Phone Working Group [21] needs to finalize its standards before TPM-like functions are widely available on mobile devices. In addition, IMA-like functions would need to be implemented in mobile computing platforms such as Palm OS, Windows Mobile and Symbian, as has been done with Linux on stationary computers. Mandatory access controls would also need to be added to these mobile platforms, as has been done with SELinux on stationary computers.

Work also remains in the policy area. Our prototype attests to the integrity of the security policies in use by the member systems of a trust overlay. However, the current system deals only with the *syntax* of these policies; it has no automated support for verifying their *semantics*. We need to work out procedures for translating human-level security requirements to machine-level security policies in order to improve our ability to reason about the security properties provided by the members of a trust overlay. As mentioned earlier, we also need to develop a way to negotiate and enforce compatible, not necessarily identical, policies across administrative domains.

4. Related Work

This section presents a brief survey of related work that is not already mentioned in this paper. In the area of attestation, the Terra project [6] uses trusted third-party certificates to establish a remote basis for believing the authenticity of a virtual operating environment, and to demonstrate that both the environment and the applications running therein are unmodified. Smith [19] explores an approach for attesting to all software layers running inside a cryptographic coprocessor [4]. Haldar and colleagues [8] build upon a trusted Java environment to implement language-based VMs that enable remote attestation of complex, dynamic, and high-level application properties in a platform-independent way. Work by Sadeghi and Stüble [16] aims to enable evaluating which security properties a remote system upholds, while abstracting the details of which hardware and software components are used in the system.

In the area of enforcing security policies in a distributed system, Ioannidis and colleagues [9] introduce the concept of a Virtual Private Service (VPS). A VPS captures, in a single policy specification, the complete access-control requirements of a service to produce a consistent environment across multiple independent enforcement points. Finally, Trusted Virtual Domains [7] offer an abstraction of security properties so that computing services can be dependably offloaded into execution environments that demonstrably meet a desired set of security requirements. The work described in this paper complements this related work with a systematic approach to building trust in mobile computing environments.

5. Conclusion

We hope that this paper has conveyed the desirability and viability of trusted mobile computing. Our concept of trust overlays applies not only to mobile computing environments but to distributed systems in general. However, the dynamic nature of interactions between mobile devices and their surroundings makes the need for trusted computing particularly acute in the mobile context. We urge the mobile computing research community to address trust issues in their systems sooner rather than later.

Acknowledgements

This work has benefited from collaborations on many aspects of trusted computing with other members of the Secure Systems Department at the IBM T. J. Watson Research Center. This work has also benefited from discussions on Trusted Virtual Domains with colleagues at the IBM Tokyo Research Lab and IBM Zurich Research Lab.

References

1. D.P. Anderson, "BOINC: A System for Public-Resource Computing and Storage," *Proc. of Workshop on Grid Computing*, November 2004.
2. J. P. Anderson et al., "Computer Security Technology Planning Study," *Technical Report ESD-TR-73-51*, Vol. I+II, Air Force Systems Command, USAF, 1972.
3. R. Cáceres, C. Carter, C. Narayanaswami and M. Raghunath, "Reincarnating PCs with Portable SoulPads," *Proc. of 3rd ACM/USENIX International Conference on Mobile Systems, Applications and Services (MobiSys)*, June 2005.
4. J.G. Dyer, M. Lindemann, R. Perez, R. Sailer, S.W. Smith, L. van Doorn and S. Weingart, "The IBM Secure Coprocessor: Overview and Retrospective," *IEEE Computer*, October 2001.
5. B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham and R. Neugebauer, "Xen and the Art of Virtualization," *Proc. of the ACM Symposium on Operating Systems Principles*, October 2003.

6. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum and D. Boneh, "Terra: A Virtual Machine-Based Platform for Trusted Computing," *Proc. of 9th ACM Symposium on Operating Systems Principles (SOSP)*, 2003.
7. J. L. Griffin, T. Jaeger, R. Perez, R. Sailer, L. van Doorn. and R. Cáceres, "Trusted Virtual Domains: Toward Secure Distributed Services," *Proc. of 1st IEEE Workshop on Hot Topics in System Dependability (HotDep)*, June 2005.
8. V. Haldar, D. Chandra, and M. Franz, "Semantic Remote Attestation: A Virtual Machine Directed Approach to Trusted Computing," *Proc. of USENIX Virtual Machine Research and Technology Symposium*, May 2004.
9. S. Ioannidis, S. M. Bellovin, J. Ioannidis, A. D. Keromytis, and J. M. Smith, "Design and Implementation of Virtual Private Services," *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, June 2004.
10. M. Kozuch and M. Satyanarayanan, "Internet Suspend/Resume," *Proc. of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 2002.
11. P. Loshin, *Big Book of IPSec RFCs: Internet Security Architecture*, Morgan Kauffman Publishers, 1999.
12. H. Maruyama, F. Seliger, N. Nagaratnam, T. Ebringer, S. Munetoh, S. Yoshihama and T. Nakamura, "Trusted Platform on Demand," *Research Report RT0564*, IBM Corporation, February 1, 2004.
13. J. M. McCune, S. Berger, R. Cáceres, T. Jaeger, R. Sailer, "DeuTeRiuM – A System for Distributed Mandatory Access Control", *Research Report RC23865*, IBM Corporation, February 2, 2006.
14. National Security Agency, Security-Enhanced Linux (SELinux). `http://www.nsa.gov/selinux`
15. Netscape, SSL 3.0 Specification. `http://www.netscape.com/eng/ssl3`
16. A.-R. Sadeghi and C. Stüble, "Property-based Attestation for Computing Platforms: Caring about Properties, Not Mechanisms," *Proc. of New Security Paradigm Workshop (NSPW)*, 2004.
17. R. Sailer, X. Zhang, T. Jaeger and L. van Doorn, "Design and Implementation of a TCG-based Integrity Measurement Architecture," *Proc. of 13th USENIX Security Symposium*, August 2004.
18. R. Sailer, T. Jaeger, E. Valdez, R. Cáceres, R. Perez, S. Berger, J. L. Griffin and L. van Doorn, "Building a MAC-Based Security Architecture for the Xen Open-Source Hypervisor," *Proc. of 22nd Annual Computer Security Applications Conference (ACSAC)*, December 2005.
19. S. W. Smith, "Outbound authentication for programmable secure coprocessors," *Proc. of 7th European Symposium on Research in Computer Security (ESORICS)*, 2002.
20. Sun Microsystems, Java 2 Platform Security. `http://java.sun.com/j2se/1.3/docs/guide/security`
21. Trusted Computing Group. `https://www.trustedcomputinggroup.org`
22. PC World, "Virus Passes from PCs to Mobile Devices," February 28, 2006. `http://www.pcworld.com/news/article/0,aid,124887,00.asp`