DROPPING THE TRUSTED COMPUTING BASE FOR APPLICATIONS ON PRODUCT SYSTEMS

Anudeep
Student Masters of Technology
Department of CSE
Sri Guru Granth Sahib World University
Fatehgarh Sahib, Punjab, India

ABSTRACT

Today we have powerful, feature-rich computer systems plagued by muscular, feature-rich malware. Flow malware work the vulnerabilities that are endemically to the huge computing that needs to be trusty to steady our semiprivate substance. This thesis presents a structure called Flicker that alleviates security-conscious developers from the headache of making significance out of this codification fund, allowing them to ore on the precaution of their own encipher. Since today's inheritance operating systems give credible be misused for the foreseeable future, we figure Flicker to coexist with these systems. Flicker allows code to execute in isolation from new software while trusting as few as 250 lines of more cipher orders of ratio smaller than symmetrical minimalist virtual tool monitors. Flicker also enables many significant attestation of the codification executed and its inputs and outputs than preceding proposals, since exclusive measurements of the security-sensitive portions of an application to be included. Flicker leverages hardware support provided by artifact processors from AMD and Intel that are widely open today, and does not order a new OS or a VMM. Flicker's properties support equal if the BIOS, OS and DMA-enabled devices are all leering. We also perform a detailed case study of Flicker on an AMD method and distribute Quiver to foursome server-side applications.

We also perform a detailed case study of the use of Flicker to reduce the trusted computing base to which users' input events are exposed on their own computers, circumventing entire classes of malware such as keyloggers and screen scrapers. This case study involves the development of a system called Bumpy that allows the user to specify strings of input as sensitive when she enters them, and ensures that these inputs reach the desired endpoint in a protected state. The inputs are processed in a Flicker-isolated code module on the user's system, where they can be encrypted or otherwise processed for a remote web server. A trusted mobile device can provide feedback to the user that her inputs are bound for the intended destination. We describe the design, implementation, and evaluation of Bumpy, with emphasis on both usability and security issues.

1. INTRODUCTION

The size and complexity of modern operating systems makes them difficult to analyze and vulnerable to attack. Applications built on these OSes inherit their vulnerabilities, as the hierarchical privilege structure of these OSes yields total control of applications to the OS. Thus, even the simplest application function operates with a trusted computing base (TCB) consisting of the union of all operating system and device driver code. Other applications running with super-user privileges can readily modify the OS on which they run, thus adding codebase to the TCB. If the OS is running on top of a virtual machine monitor, then it too must be trusted. Regardless of how a computer system is designed, the security of a given application remains intimately tied to the user who is operating it. There are numerous ongoing projects to build a new "secure" OS and there have been many attempts in the past [9]. While we applaud these efforts, the reality is that computer systems are not always chosen for their technical merits. The availability of essential applications, network effects, and economies of scale apply tremendous pressure to retain legacy software. The size of the installed base of today's popular commodity OSes renders a clean-slate approach infeasible. Rather, we must coexist with these systems, while adhering to the axioms of isolation, small code size, and user-friendly design. Trusted Computing technologies based on a Trusted Platform Module (TPM) security chip such as remote attestation and sealed storage have been proposed with the goal of improving the resilience of commodity computing platforms against software-based attacks. These technologies have received criticism for their ability to erode users' privacy because they require trusted third parties that uniquely identify the user's platform and then possibly the user through product registration information [2]. Additionally, integrity measurement and remote attestation based on a static root of trust leak information about all software loaded during the current boot cycle, even if a remote challenger is only interested in a single application on the user's computing platform.
MINIMIZATION INFRASTRUCTURE
We develop Flicker, a secure execution architecture that allows security sensitive code to execute in complete isolation from all other software (including the operating system and VMM, if present). Flicker's properties hold even if the BIOS, OS and DMA-enabled devices are all malicious. This dramatic reduction in the size of the TCB for an application enables meaningful software attestation and facilitates formal security analysis of the software remaining in the TCB.

TCB REDUCTION FOR SENSITIVE USER INPUT

We have already described how vulnerability in any part of a system's OS renders users' sensitive data insecure regardless of what application they may be running. On top of this untrustworthy OS sits a complex and monolithic web browser, which faces protection and assurance challenges similar to those of the OS [11]. It is not surprising that trusting this software stack for the protection of private data in web transactions often leads to data compromise. In this we provide a detailed case study where we develop Bumpy, a system that builds on Flicker for protecting user input to web pages. This case study is more thorough than our server-side Flicker applications, and illustrates many of the challenges arising from the use of Flicker to secure client-side applications.

HUMAN-VERIFIABLE AUTHENTICATION

Trusted computing technology depends on a third party to certify a computing platform and its TPM as complying with the relevant specifications without leaking the exact identity of the platform. Unfortunately, the resulting certificates do not provide strong physical identification of the relevant computing platform to the platform owner. This limitation is particularly egregious when a user wants to verify the security properties of a public computer, e.g., in an Internet cafe. In this we develop Seeing is Believing (SiB), a technique leveraging two-dimensional barcodes and camera phones to create a visual channel that provides demonstrative identification of communicating devices to the human user(s). SiB works even when devices share no prior context, or when the prerequisite of a trusted third party or public key infrastructure (PKI) may undesirably inflate the user's TCB, if such an authority exists at all. These devices may be components in one person's computing environment, such as her keyboard, display, mobile phone, laptop or digital camera, or they may be devices belonging to two different people. The most relevant use of SiB for this thesis is to ascertain the identity of the TPM in a user's computer. This enables one to bind a third-party certificate to the physical identity of a particular platform, thereby allowing users or administrators who take an interest in security to effectively manage their own systems.

1.1 ARCHITECTURAL RECOMMENDATIONS

One result of our performance evaluation of Flicker is that we use the available secure-virtualization hardware features far more frequently than intended during their original design. In this we summarize the primary sources of overhead for Flicker and make architectural recommendations for next-generation hardware to better support Flicker.

2.PROBLEM STATEMENT

2.1 PROBLEM DEFINITION

We define the class of adversaries we consider. We also define our goals and explain why the new hardware capabilities do not meet them on their own.

2.1.1 ADVERSARY MODEL

At the software level, the adversary can subvert the operating system, so it can also compromise arbitrary applications and monitor all network traffic. Since the adversary can run code at ring 0, it can invoke the SKINIT instruction with arguments of its choosing. We also allow the adversary to regain control between Flicker sessions. We do not consider Denial of Service attacks, since a malicious OS can always simply power down the machine or otherwise halt execution to deny service. At the hardware level, we make the same assumptions as does the Trusted Computing Group with regard to the TPM. In essence, the attacker can launch simple hardware attacks, such as opening the case, power cycling the computer, or attaching a hardware debugger. The attacker can also compromise expansion hardware such as a DMA-capable Ethernet card with access to the PCI bus. However, the attacker cannot launch sophisticated hardware attacks, such as monitoring the high-speed bus that links the CPU and memory.

3. CONCLUSION AND FUTUREE WORK

Networked computer systems have grown in complexity to a level we can no longer control. To date, formal methods have not reached a level of maturity or scalability to prove these systems correct. Consequently, we use TCB minimization to approach correctness for security sensitive parts of applications, observing that for many applications the security sensitive operations are relatively small. We have developed a method for isolated code execution on commodity systems with a TCB that adds as few as 250 lines of code to application-specific functionality. TPM based attestation can be used to convince a verifier, which may be local or remote that the application-specific code did execute with the desired protections. Our work finally gives application developers the opportunity to write secure applications without relying on the security of layer upon layer of legacy software, and without breaking compatibility with today's commodity systems.

The use of our system can be attested to both a local verifier and a remote webserver, giving the user and/or webserver (and hence the organization that operates it, e.g., a bank) the ability to determine with higher assurance than can be obtained today that the user's inputs were protected. We have shown that trust relationships predicated on authentic public key exchange can be established without depending on a trusted third party or public key infrastructure, further reducing the amount of trust that must be placed in entities beyond the user's control. This gives the conscientious user the opportunity to configure its own input and verification devices, and the savvy user the ability to retain total control over her own privacy relevant inputs.

References

[1] Advanced Micro Devices. AMD64 architecture programmer's manual: Volume 2: System programming. AMD Publication no. 24594 rev. 3.11, December 2005. 19, 24, 25

[2] D. P. Anderson. BOINC: A system for public-resource computing and storage. In Proceedings of the Workshop on Grid Computing, November 2004.

[3] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@Home: An experiment in public-resource computing. Communications of the ACM, 45(11):56–61, 2002. 54

[4] W. A. Arbaugh. Improving the tcpa specification. IEEE Computer, 35(8):77–79, 2002. 18

[5] W. A. Arbaugh, D. J. Farber, and J. M. Smith. A reliable bootstrap architecture. In Proceedings of the IEEE Symposium on Security and Privacy, May 1997. 31, 172

[6] D. Balfanz and E. W. Felten. Hand-held computers can be better smart cards. In Proccedings of the USENIX Security Symposium, August 1999. 119, 175

[7] D. Balfanz, D. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In Proceedings of the Symposium on Network and Distributed Systems Security (NDSS), February 2002. 89, 120, 122, 124, 125, 180

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), 2003. 35, 170, 171

[9] P. R. Barham, B. Dragovic, K. A. Fraser, S. M. Hand, T. L. Harris, A. C. Ho, E. Kotsovinos, A. V. Madhavapeddy, R. Neugebauer, I. A.Pratt, and A. K. Warfield. Xen 2002. Technical Report UCAM-CLTR-553, University of Cambridge, January 2003. 30

[10] L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar. Device-enabled authorization in the Grey system. In Proceedings of the 8th Information Security Conference (ISC), September 2005. 139

[11] BeagleBoard.org. BeagleBoard revision B6 system reference manual revision 0.1. BeagleBoard.org, November 2008. 104

[12] D. E. Bell and L. J. LaPadula. Secure computer systems: Unified exposition and Multics interpretation. Technical report, MITRE MTR-2997, March 1976. 18