Shame on Trust in Distributed Systems

Trent Jaeger, Patrick McDaniel, Luke St. Clair
Pennsylvania State University

Ramón Cáceres, Reiner Sailer
IBM T. J. Watson Research Center

1 Introduction

Approaches for building secure, distributed systems have fundamental limitations that prevent the construction of dynamic, Internet-scale systems. In this paper, we propose a concept of a shared reference monitor or *Shamon* that we believe will provide a basis for overcoming these limitations. First, distributed systems lack a principled basis for trust in the trusted computing bases of member machines. In most distributed systems, a trusted computing base is assumed. However, the fear of compromise due to misconfiguration or vulnerable software limits the cases where this assumption can be applied in practice. Where such trust is not assumed, current solutions are not scalable to large systems [7, 20]. Second, current systems do not ensure the enforcement of the flexible, distributed system security goals. Mandatory access control (MAC) policies aim to describe enforceable security goals, but flexible MAC solutions, such as SELinux, do not even provide a scalable solution for a single machine (due to the complexity of UNIX systems), much less a distributed system. A significant change in approach is necessary to develop a principled trusted computing base that enforces system security goals and scales to large distributed systems.

Our proposal is to develop scalable mechanisms for composing a verifiable reference monitoring infrastructure that spans Internet-wide distributed systems. We refer to a set of reference monitors that provides coherent security guarantees across multiple physical machines as a *Shamon* [1]. While this may sound like a mere extension of the well-known reference monitor concept, we propose several key differences: (1) the credentials of secure hardware (e.g., Trusted Computing Group's Trusted Platform Module), rather than users, are used to authenticate individual reference monitoring systems in the *Shamon* ; (2) trust in the *Shamon* is based on attestation of reference monitoring properties: tamperproofing, mediation, and simplicity of design; (3) virtual machine monitoring is used to establish coarse-grained domains, which results in significant simplification of MAC policies; (4) policy analyses verify that these MAC policies satisfy the *Shamon* application's security goals when enforced by the *Shamon*; and (5) based on this restricted definition of trust, a focused logic is defined that enables scalable evaluation of this trust by components of the distributed system that is also resilient to dynamic changes in the application.

The *Shamon* approach addresses the fundamental challenges described above. First, trust is built from the bottom-up via secure hardware credentials that enable attestations of virtual machine-based enforcement for each machine. Second, the MAC policy enforced by the *Shamon* is used to prove enforcement of system security goals. We define a logical representation for verifying these criteria that enables scalable management of large *Shamon* even under changes in application configuration. Each of the five tasks that convert a reference monitor into a *Shamon* presents substantial research challenges, but we aim to demonstrate that each has tractable solution potential and that the resultant *Shamon* system will provide a foundation for large-scale distributed authorization. To motivate its design, we introduce our prototype application of the *Shamon* in the following section.

2 Application

The *Playpen* is a Xen-based, virtual machine (VM) environment for the students taking security courses at Pennsylvania State University. Each student is given their own virtual machines in the Playpen. Over the course of the semester, students are required to configure and build security apparatus to defend their machines against attacks from the faculty and TAs. The isolation, persistence, and mobility of the VM environment provides ideal conditions for pedagogy: users can experiment with security apparatus under the controlled environment.

The current Playpen is the prototype for a larger project supporting wide-area mobile and secure computing environments. The long term goal is to extend the Playpen to encompass all aspects of university life. In this, a user would be given one or more virtual machines that would migrate to the location where they are working. The central challenge of this work is to support the users' ability to move freely within the university environment. The system must securely support arbitrary migration to previously unknown hardware at a previously unknown location and share data with previously unknown collaborators. Note that while the environment aims at a single university system, we are not centrally-administered: there is different administration at each campus, and some departments also administer their own machines.

Consider a typical day of Alice the graduate student in

[1] The name is short for *Shared Monitor* and related to the word *shaman* meaning "... a medium ... who practices ... control over natural events" words removed for effect, not necessarily accuracy).

this new university. She wakes up at noon and goes to class. Alice joins a live coalition of class participants by logging into a host in her classroom. She exits the coalition at the end of class, and at lunch she surfs the Internet and exchanges personal communication within her protected environment at the local student union. After lunch, she heads to the laboratory and performs research and shares data with the other graduate students. At the end of the day, she meets with her advisor and shares summary data and exchanges results. She heads home and plays a massively multiplayer game with thousands of other gamers until dawn over the Internet.

Such is the nature of university life. The "roles" of Alice's computing environment and the environments in which she interacts evolve constantly; from class participant, personal communication, researcher, advisee, and gamer. Moreover, the set of hosts to which she has an association is also changing. What is interesting here is not that this somehow changes the way Alice lives, but that her computing environment follows her throughout her life.

The security challenges of this environment are non-trivial. The physical machines within the open university environment are largely unknown and often compromised.[2] The applications are as diverse as the environments in which Alice lives, from classroom to research to gaming. Furthermore, the collaborations in which Alice participates change hourly, and often form and disband organically. It is clear that: (1) supporting this environment requires significant security, and (2) current commodity environments (e.g., distributed file systems and VPNs) do not support it. Note that large corporate environments are similar–users will move freely through a largely insecure complex and use data and applications as needed.

Research that enables articulation of finer-grained policies across distributed systems, for distributed file access (e.g., [15, 3]) and trust management (e.g., [4, 14]), often assume trust in the trusted computing base as well. An exception is the Taos operating system approach [2] which has a form of secure booting for establishing trust in the infrastructure. However, building trust in a single machine is insufficient. We need to build trust in enforcement across distributed applications within the distributed system (e.g., each of Alice's roles) and ensure that distributed authorization policy enforces the security goals of those applications. In order for this to be truly useful, it must enable large distributed applications to be supported.

The five key design requirements identified in the preceding section are reflected in the university environment: users need to vet the many untrusted machines in some reliable and secure way; they need to vet the policy enforcement infrastructure (simplicity, tamperproofing, and mediation); they need to articulate an inter-host (sharing) security policy; they need to ensure that all hosts sharing data provide a consistent view of security; and it must scale – there are over 41,000 students at Penn State spread out over 24 campuses.

[2]Would any sane person completely trust a host in a open university laboratory? Seriously.

3 Coalitions and *Shamon*

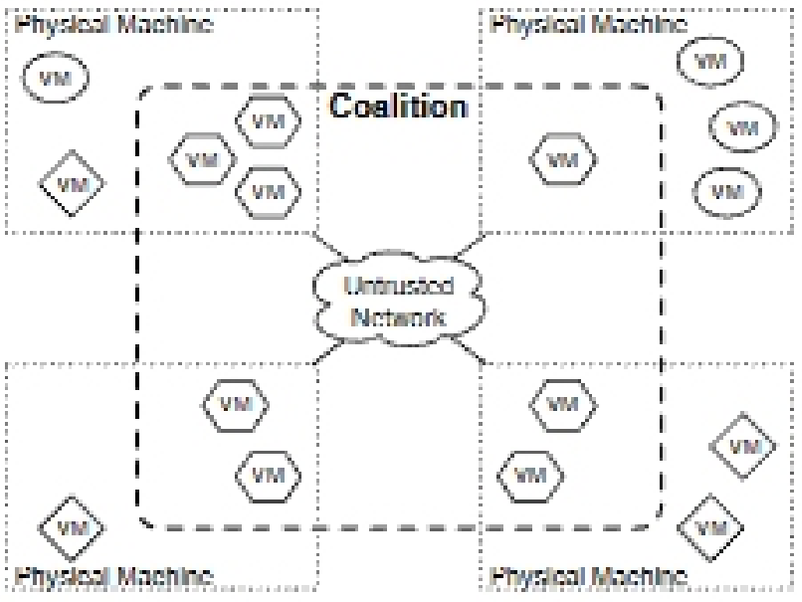

Figure 1: Example of a distributed coalition. Virtual machine (VM) instances sharing common Mandatory Access Control (MAC) labels on multiple physical hypervisor systems are all members of the same coalition.

Figure 1 illustrates the conceptual idea for future distributed applications. A distributed application is a *coalition* of VMs that executes across multiple physical platforms. Each member of the coalition may reside on a different physical machine, which may itself execute multiple coalitions. The physical machines themselves each have a reference monitor capable of enforcing MAC policies over all of their VMs.

We define the *Shamon* as follows. A *Shamon* is a set of reference monitors serving a coalition by enforcing its security goals. A reference monitor may belong to multiple *Shamon*, so its enforcement must ensure the satisfaction of the security goals for each. The challenge is to establish trust in the *Shamon* reference monitors' enforcement of a coalition security goal. This trust must be upheld in a scalable fashion as VMs join the coalition or migrate between machines. In so doing, the *Shamon* provides authorization across an entire coalition as if it were a single machine.

For Alice, each VM in a coalition represents an instance of her work on a specific task. She may work on her research in the lab, in class, or in the student union, and her research coalition enables these VMs to communicate. However, her gaming and browsing VMs would not be part of this coalition. In fact, the research coalition enables isolation of the research VMs from the gaming and browsing VMs even when they are running on the same machine at the same time.

We envision different requirements for managing Alice VMs than in a traditional VM isolation system. First, the security focus is to separate Alice's workloads based on trust (i.e., trusted from untrusted) or domain (i.e., research from school work), but total isolation is too restrictive. For ex-

ample, some data from an untrusted domain (e.g., Google search results) may be useful in a trusted domain (e.g., research paper). These are finer-grained and more flexible security requirements than are typical of VM systems. Second, the VMs will be more dynamic and composed into larger systems than is traditionally the case. Also, some VMs may be destroyed on a frequent basis. In addition, large-scale coalitions with changing memberships may be constructed for particular causes that Alice may join, such as conferences, auctions, rallies, or social events.

4 *Shamon* Challenges

The basic mechanism for composing a coalition is shown in Figure 2. Each machine that will join a coalition must have a credential registered with the coalition authority, such that statements made on behalf of the machine (e.g., attestations) can be verified (messages 1 and 2). Joining requires attestation of *Shamon* properties which results in a proof of acceptance from the authority (messages 3 and 4). This proof is used to communicate with another coalition member, and this coalition member establishes a *dependency* on this proof and any status changes from the authority (messages 5 through 7).

The scalability of the approach comes from reusing coalition authority attestations at join time and deferring proof of integrity until communication is initiated. These advantages are similar to typical PKI approaches where the proof of the possession of a private key is generated by an authority and verification of this possession is done when communication is initiated.

There are some important differences, however, between this approach and PKI. First, a major benefit is that trust is built from machines rather than individuals. Thus, trust in the trusted computing base is built in a bottom-up manner along with the booting of the trusted computing base itself. Keys can be stored and used in secure hardware rather than in application software. Second, a major challenge is that a reference monitor's status may change, motivating a revocation of the member. Remember that we only depend on the reference monitor and coalition MAC policy being correct. Normally, these will not change, but we need a lightweight mechanism to convey the status quo without missing a compromise. In theory, TPM statements of the integrity value (and a nonce for freshness) could be provided and checked frequently at a low cost, except the current TPMs are slow and use public key cryptography. The benefit of bottom-up mechanism to establish trust should motivate an investigation into making efficient integrity maintenance practical.

Below, we assess the five *Shamon* features from Section 1 relative to Alice's requirements.

***Shamon* Authentication** In order to verify a *Shamon* for joining a coalition, we must be able to authenticate the machine upon which the *Shamon* runs. Secure hardware of the machine, such as the Trusted Computing Group's Trusted Platform Module [1] (TPM) is capable of generating credentials that can be certified by an authority (e.g., using Direct Anonymous Attestation). Such credentials can be used to register the machine for use in a coalition via such an authority, called a *coalition authority*, as shown in Figure 2. Note that since TPMs are not tamperproof, some degree of physical security is required. For Alice, different physical requirements may be necessary for different coalitions: the research coalition may require machines protected by the university, whereas her coalition for completing her tax return may only require machines that meet her physical security requirements. Acceptable models combining physical security and credentials are a research challenge.

***Shamon* Attestation** When Alice picks a machine to join a particular coalition, this machine must prove that it can join the *Shamon*. This involves the following steps: (1) provide an integrity measurement of *Shamon* components using remote attestation protocols based on the TPM; (2) obtain the coalition's MAC policy from a coalition authority; and (3) construct a proof of its consistency with this policy (e.g., in labeling) and its ability to enforce the security goals required. Remote attestation approaches have been developed that enable measurement of trusted code and information flow policies for trusted applications [11]. We envision a significant benefit from having the coalition determine the MAC policy for the application rather than trying to configure systems *a priori*. Note that we can piggy-back the attestation verification on the negotiation of secure communication channels (done as a result of message 5), such as for Labeled IPsec [10].

User Authentication Also, the users must authenticate themselves to the *Shamon* infrastructure. The challenge is that the user does not necessarily have trust in the physical platform that she is using at a particular time. Even a machine that may appear to be shutdown may actually be compromised (e.g., by a virtual machine rootkit [13]). A challenge for the user is to establish that she can submit her authentication secrets without fear of losing them. The user must be to verify a statement from an authority she trusts (e.g., a coalition authority) that vouches for the fresh attestation of the platform that she is actually using. Enabling a user to verify the authenticity of a machine requires a trusted path in general, although some social mechanisms may be effective in controlled environments, such as a campus (e.g., trusted labeling, such as proposed for room access [17]).

***Shamon* MAC Policy Simplicity** The basis for simplifying MAC policy is the use of virtual machine communication as the basis for security guarantees. In a Xen-based system, sHype [19] controls inter-VM communication by authorizing only Xen grant tables (i.e., shared memory), Xen event channels (i.e., basic IPC), and Linux IPsec tunnels (i.e., network communication via Xen's domain 0) must be controlled. Other system resources, such as disk space and memory are partitioned for virtual machines, so they are