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 nontrivial. 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*

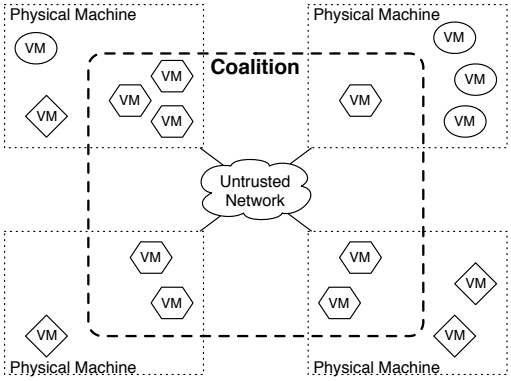


Figure 1: Example of a distributed coalition. Virtual machine (VM) instances sharing common Mandatory Access Control (MAC) labels on multiple physical hypervisor systems are all members of the same coalition.

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.

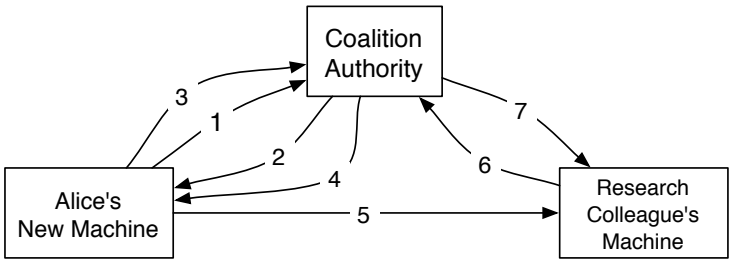
Below, we assess the five *Shamon* features from Section 1 relative to Alice's requirements.

***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



1: Register physical machine (e.g., TPM) with verification infrastructure
2: Confirm successful registration
3: Join coalition request (attest enforcement and security goals)
4: Proof of verification for join
5: Alice sends request (IPsec negotiation)
6: Accept proof of verification
7: Notify change to member status

Figure 2: Coalition member join process: Secure hardware enables verification of *Shamon* enforcement of security goals which is conveyed when Alice participates with another member.

isolated by default. As a result, a much simpler MAC policy for inter-VM communication can be defined [16] than for a UNIX system (e.g., SELinux strict policy).

VM isolation is likely too restrictive for Alice's tasks, as we discussed earlier, because there will be a benefit in transferring information from applications in one VM to another. Because VMs will be application-focused, we surmise that it will be more appropriate for applications to dictate the enforcement of these security requirements rather than operating systems (although they may be implemented via OS mechanisms). Recent work in information-flow languages [9] and security-aware code [6] may provide a basis for integrity protections based on a combination of VM protections and limited application protections [21]. For example, Alice's document application may prove that it can handle the receipt of low integrity text data via limited, filtering interfaces.

***Shamon* Security Goal Verification** A consistent security policy includes all facets of MAC policy definition, including subjects and object labeling mechanisms (i.e., deciding how labels are assigned), permission assignments (i.e., assigning permissions to subjects), and label transitions (i.e., changes in subject labels for a process), if any. Since this is provided to the *Shamon* upon joining a coalition, the MAC policies used in the coalition may be measured via attestation. Remote parties can then verify that the MAC policy satisfies coalition security goals in the context of the *Shamon* and other coalitions running on that system. Recent work in MAC policy analysis shows that information flow security properties can be verified for very complex policies [8, 12, 22]. In general, verification involves detection of information flow problems where secret data may be leaked or low integrity data may enter a trusted application. Thus, verification can detect a VM that is leaking Alice's research information or receiving an untrusted executable. Using application-level enforcement approaches as described above would enable limited I/O for applications entrusted to perform such operations using only approved interfaces.

***Shamon* Trust Logic** In order to reason about the trust state of the *Shamon* across large system, we need a representation for this state. We propose a basic predicate logic (space limits prevent its discussion) that defines a few specific predicates that enable reasoning about attestations meeting enforcement goals, MAC policies meeting security goals, and virtual machines running on enforcing machines. We believe that a logic focused on these specific types of properties will enable practical system constructions, but much research is to be done. The challenges in designing such a logic include: (1) accuracy in representing real behavior; (2) ability to efficiently handle coalition dynamics (i.e., use monotonic logic under changing conditions); and (3) limitation of manual policy specification. Alice's machine must be able to correctly determine that no attack on the enforcement of coalition security goals (i.e., the *Shamon*) can go undetected (within a reasonable probability), even when the coalition membership changes (perhaps due to a detected attack), and with little or no input from Alice or her system's administrators.

5 Thinking About Usability

To illustrate the challenges in building a distributed system that can be practical, we examine two approaches used to build distributed systems that have had some success: (1) automated teller machine (ATM) networks and (2) virtual network computing (VNC). We do not mean to imply that coalitions supported by *Shamon* have the same requirements as ATMs or VNCs (although close to some versions of the latter), but reviewing the requirements of a successful distributed infrastructure may be illuminating in trying to build a more flexible approach. We then compare the environment of *Shamon* systems to that of ATM networks and VNC systems to identify challenges in making the *Shamon* approach practical.

5.1 Automated Teller Machine Networks

ATMs enable users to perform specific types of transactions using a distributed network of machines. Like the *Shamon* approach, the ATM networks define approaches for authentication (users and ATMs), attestation (banks verify ATM software), and security goals (specific to banking transactions).

While there have been rogue ATMs installed in the past that have been used to steal user's PINs, users have come to trust a visual authentication of ATMs. Such authentication

partly comes from physical security (i.e., located in a bank or public place) where someone might notice a rogue system. Of course, the fact that the banks assume the responsibility for most fraud is also important. In the *Shamon* system, we may want to leverage social notions of location and responsibility where possible. For example, a university lab may guarantee secure boot of its machines. However, more effort will be required in ensuring this guarantee than an ATM case. Otherwise, users will have to authenticate machines via coalition authorities using a strong cryptographic protocol. Users cannot enter secrets (e.g., ATM PINs) into machines unless a means for validating their authenticity can be provided.

The banks require attestation of the ATM systems before performing transactions on behalf of customers. Currently, attestation is done by the tamperproof, secure coprocessor. The cost of tamperproofing is too high for *Shamon* systems, so we can only assume Trusted Computing Group (TCG) Trusted Platform Modules (TPMs) or similar. The lack of tamperproofing means that physical security guarantees are required for security. Perhaps the agreements for physical security necessary to authenticate the coalition systems can be leveraged.

The most significant difference between an ATM and a *Shamon* system will be that the user can get their applications downloaded to the coalition machine and request enforcement of their security requirements by that machine. ATMs provide applications for the specific authenticated user, and no sharing of information between applications is permitted. A user may have multiple application VMs and must be able to share information among them. Also, these VMs may have to interact with other VMs on other machines. The coalition authority must support the user in identifying applications, security requirements, and ensuring that the security requirements are enforced. The individual machines must be able to interpret and enforce the security requirements. The expanded application suite probably has a bigger impact on users than coalition authorities or machines. Well-defined, but effective, means of usage will have to be determined.

5.2 Virtual Network Computing

In Virtual Network Computing (VNC), the interface to a particular base machine is exported to another remote machine where the user can interact with the base machine in the same manner as if the user was local to that machine. Such a mechanism has evolved from the export of individual terminals via XWindows [23] to the export of entire screens to machines with different operating systems via RealVNC [18] to the export of interfaces via HTTP to any machine via GoToMyPC.com [5]. The latter enables access to a base machine via a web browser on any remote machine with minimal previous configuration.

Security on these systems mainly focuses on user authentication and secure communication. RealVNC aims to provide a remote access, but the idea is that the user has a remote machine that is under her control (e.g., her personal laptop). Thus, attestation is not strictly necessary. GoToMyPC.com advertises access to your computer from any machine on the Internet, but security focuses on user authentication and secure communication rather than the integrity of the computer being used. The mobile user goes to the GoToMyPC.com website to be authenticated and setup a secure communication channel to the base machine. The website authenticates the user, but does not verify the integrity remote system. It is the user's responsibility to detect a malicious system much as in the ATM case, but a fraudulent or compromised computer system is much more likely in this case. *Shamon* attestation aims to verify the integrity of such machines to ensure mandatory integrity requirements are met.

In VNC systems, access control requirements are enforced on the base machine as that is where the computation takes place. Thus, VNCs export interfaces, but not access control requirements. As a result, there is no control of information once it reaches the remote machine. A compromised machine can leak data, so mandatory access control (e.g., multi-level security) cannot be enforced. The *Shamon* approach aims to ensure coherent security enforcement across the machines in each coalition.

6 Conclusions

In this paper, we proposed the concept of a shared reference monitor or *Shamon* as the basis for building distributed systems that enforce intended security goals. This approach leverages the secure hardware for building trust from the bottom-up, virtual machines for simplifying MAC policy for verification, and a trust logic for representing *Shamon* trust state. Initial work shows promise, but key challenges remain, such as accurately tracking the integrity of individual systems.

References

[1] Trusted Computing Group. `http://www.trustedcomputinggroup.org/`, Mar. 2005.

[2] M. Abadi, E. Wobber, M. Burrows, and B. Lampson. Authentication in the Taos Operating System. In *Proceedings of the 14th ACM Symposium on Operating System Principles*, pages 256–269, Asheville, NC, USA, 1993.

[3] E. Belani, A. Vahdat, T. Anderson, and M. Dahlin. The CRISIS wide area security architecture. In *Proceedings of the 7th USENIX Security Symposium*, Jan. 1998.

[4] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote Trust Management System, version 2. IETF RFC 2704, Sept. 1999.

[5] Citrix Online, Inc. GoToMyPC: Remote Access to Your PC from Anywhere, July 2006. http://www.gotomypc.com/.

[6] V. Ganapathy, T. Jaeger, and S. Jha. Retrofitting legacy code for authorization policy enforcement. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, May 2006.

[7] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. In *Proceedings of the 19th ACM Symposium on Operating System Principles(SOSP 2003)*, Bolton Landing, NY, USA, Oct. 2003.

[8] A. L. Herzog, J. D. Guttman, D. R. Harris, J. D. Ramsdell, A. E. Segall, and B. T. Sniffen. Policy analysis and generation work at MITRE. In *Proceedings of the first Annual Security-enhanced Linux Symposium*, March 2005.

[9] B. Hicks, K. Ahmadizadeh, and P. McDaniel. From Languages to Systems: Understanding Practical Application Development in Security-typed Languages. Technical Report NAS-TR-0035, Penn State NSRC, 2006.

[10] T. Jaeger, D. King, K. Butler, S. Hallyn, J. Latten, and X. Zhang. Leveraging ipsec for mandatory per-packet access control. In *Proceedings of the Second IEEE Communications Society/CreateNet International Conference on Security and Privacy in Communication Networks*, Baltimore, MD, USA, Aug. 2006.

[11] T. Jaeger, R. Sailer, and U. Shankar. Prima: Policy-reduced integrity measurements architecture. In *Proceedings of the 11th Symposium on Access Control Models and Technologies*, Lake Tahoe, NV, USA, June 2006. To appear.

[12] T. Jaeger, R. Sailer, and X. Zhang. Analyzing integrity protection in the SELinux example policy. In *Proceedings of the 12th USENIX Security Symposium*, pages 59–74, Aug. 2003.

[13] S. T. King, P. M. Chen, Y. Wang, C. Verbowski, H. J. Wang, and J. R. Lorch. Subvirt: Implementing malware with virtual machines. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, May 2006.

[14] N. Li, B. N. Grosof, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security (TISSEC)*, 6(1):128–171, Feb. 2003.

[15] D. Mazieres, M. Kaminsky, M. F. Kaashoek, and E. Witchel. Separating key management from file system security. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP '99)*, pages 124–139, 1999.

[16] J. McCune, S. Berger, R. Cáceres, T. Jaeger, and R. Sailer. Deuterium: A system for distributed mandatory access control. Research Report RC23865, IBM T.J. Watson Research Center, Feb. 2006. In submission.

[17] J. McCune, A. Perrig, and M. Reiter. Seeing is believing: Using camera phones for human-verifiable authentication", booktitle = "proceedings of the 2005 ieee symposium on security and privacy", address = Oakland, CA, USA, month = may, year = 2005.

[18] RealVNC Ltd. About RealVNC, July 2006. http://www.realvnc.com/.

[19] R. Sailer and *et al*. Building a MAC-based security architecture for the Xen opensource hypervisor. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005)*, Miami, FL, USA, Dec. 2005.

[20] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, USA, Aug. 2004.

[21] U. Shankar, T. Jaeger, and R. Sailer. Toward automated information-flow integrity verification for security-critical applications. In *Proceedings of the 2006 ISOC Networked and Distributed Systems Security Symposium (NDSS'06)*, San Diego, CA, USA, Feb. 2006.

[22] Tresys technology, SETools policy tools for SELinux. http://www.tresys.com/selinux/selinux_policy_tools.shtml.

[23] X.Org. X.Org Foundation, July 2006. http://www.x.org/.