Usage Control Platformization via Trustworthy SELinux

Masoom Alam
IM | Sciences
Peshawar,Pakistan
mmalam@imsciences.edu.pk

Qi Li
Dept of Computer Science
Tsinghua University, Beijing, China
qili@csnet1.cs.tsinghua.edu.cn

Jean-Pierre Seifert
Samsung Information Systems America
San Jose, California,USA
j.seifert@samsung.com

Xinwen Zhang
Samsung Information Systems America
San Jose, California,USA
xinwen.z@samsung.com

ABSTRACT

Continuous access control after an object is released into a distributed environment has been regarded as the usage control problem and has been investigated by different researchers in various papers. However, the enabling technology for usage control is a challenging problem and the space has not been fully explored yet. In this paper we identify the general requirements of a trusted usage control enforcement in heterogeneous computing environments, and also propose a general platform architecture to meet these requirements.

Categories and Subject Descriptors

K.6 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Security and Protection; K.4.4 [**Electronic Commerce**]: Security

General Terms

security

Keywords

Trusted Computing, Usage Control, SELinux

1. INTRODUCTION

The traditional access control problem [9, 12, 16] is considered in closed environments where identities of subjects and objects can be fully authenticated and enforcement mechanisms are trusted by system administrators which define access control policies. However, with increasing distributed and decentralized computing systems, more computing cycles and data are processed on *leaf nodes*. This leads to two distinct access control problem spaces. The first one focuses on the reasoning of authorizations with subject attributes from different authorities. For example, in trust management [4, 8, 13, 17] systems, a user presents a set of attributes or credentials and another subject (e.g., a resource or service provider) can determine the permissions of the user based on the presented credentials. In this problem, objects are typically protected in a centralized server. The second problem focuses on continuous control on accesses to an object after it is distributed to other (decentralized) locations or platforms, which is called the usage control problem proposed by researchers in literatures [14, 19, 20, 23].

Although there is no precise definition in the literature, the main goal of usage control is to enable continuous access control to objects *after* an object is released to a different control domain from its owner or provider, especially in highly distributed and heterogeneous environments. Typically, a usage control policy is defined for a *target object* by its *stakeholder*, which specifies the conditions that accesses to the object on a *target platform* can be allowed. A stakeholder can be the owner of a target object, or a service provider that is delegated by the object owner to protect the object. An object in usage control can be static data, various types of messages, or user or subject attribute or even a credential. Thus, this makes the problem pervasive in many distributed computing applications such as healthcare information systems, Web Services, and identity management systems. Different from other distributed access control problems, such as trust management, in usage control, an object is located out of the controlling domain of a policy stakeholder such that (1) there are many aspects of access control decisions other than subject identities and attributes, and (2) an object stakeholder needs high assurance on the enforcement of the policy.

As Figure 1 shows, an object and its usage control policy are distributed from a data provider to a target platform. The policy is enforced in the platform to control the access to the object within a *trusted subsystem*. Typically, an access control decision is determined according to pre-defined factors specified in a policy, which, logically, can be defined based upon the information of the subject and the object of an access request, where the subject is an active entity trying to perform actions on the passive entity object. In closed access control systems such as in a local platform, policies are defined based upon the identities of subjects and objects. In traditional distributed access control systems such as trust management, policies are defined based on attributes or credentials that are certified by external authorities. However, in usage control, access control policies can be defined by very general attributes of subjects and objects, such as application-specific attributes and temporal status. Furthermore, as an object can be located on various platforms in a heterogeneous environment such as a mobile device, environmental restrictions and system conditions are mandatory decision factors in many applications, such as location-based service and time-limited access. An *ongoing* access should be terminated if these environmental or system conditions change which violate policies. For example, a mobile application might require that a service can be used only if a mobile device is in a particular location, which itself is activated by a user through the service agent deployed on the mobile device. Simply relying on traditional access control mechanisms in a target platform cannot satisfy these requirements since the decision factors (i.e., subject and object attributes) of these approaches are mostly static and pre-defined and cannot fit a *dynamic* computing environment.

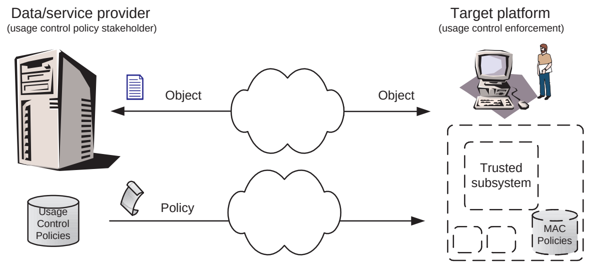

Figure 1: Abstract architecture

As usage control is naturally distributed, another challenge to enforce usage control policies is the trustworthy of the security enforcement mechanism. Typically, an access control decision is made and enforced by a reference monitor, which has the requirements of being tamper-proof, always-invoked, and small enough [5, 10] — which is relatively easy to achieve at least in closed systems. Note that in trust management systems, policy enforcement is still within the stakeholder's control domain of an object. However, as objects or services are deployed to different domains from its stakeholder, a mandatory requirement for usage control is the trustworthy enforcement of security policies by the reference monitor. Here, through trustworthiness, a stakeholder needs to ensure that (1) all factors for usage control decisions can be obtained and their information (e.g., attribute values or environmental conditions) are authentic, (2) correct decisions are made based on these factors, (3) the reference monitor enforces access control decisions correctly, and (4) all accesses to a target object on a target platform have to go through the reference monitor. Overall, by a "trusted subsystem" we mean that it is expected to behave in a "good" manner and this manner can be especially verified by the policy stakeholder.

Previous work on usage control focus on high level policy specifications and conceptual architectures [14, 19, 20, 23], while the enabling and trusted mechanisms are mainly relied on digital rights management (DRM) approaches. However, DRM mechanisms cannot support general attributes and trusted enforcement in ubiquitous environments. Most importantly, DRM approaches cannot provide an overall solution for usage control in open and general-purpose target platforms, since they usually rely on software-enabled payment-based enforcement in relatively closed environments, e.g., through a media player by connecting to a dedicated license server. Another intuitive solution is to use cryptography algorithm. For example, a stakeholder can encrypt a target object such that it only can be decrypted on a target platform with a particular application. Fundamentally, this has the same problems as the DRM approach, since a typical DRM scheme relies on encryption/decryption with a unique key shared between a client and content server [1, 3]. Particularly, cryptography alone cannot protect the key during the runtime on a target platform such as to build a trusted subsystem [18]. For example, malicious software can easily steal a secret by exploring some vulnerability of the protection system, either when the secret is loaded in some memory location, or when the secret is stored locally.

As one of the main contributions of this work, we consider the integrity of a subsystem in access control mechanisms. With this, not only traditional subject and object attributes are considered in access control decisions, but also the integrity information of subjects and objects, and any other components necessary in a trusted subsystem. The overall goal of our approach is to build a "virtually closed" and trusted subsystem for remote usage control policy enforcement.

The present paper is organized as follows. Section 2 summarizes the principles to build a distributed usage control system. We describe a platform architecture to build a trusted subsystem in Section 3. We eventually conclude this paper and present our ongoing work in Section 4.

2. DESIGN PRINCIPLES

In our work, we have identified the following general security requirements and design principles for usage control.

Requirement 1: Need high assured but usable security mechanism Typically in usage control, objects are located out of the domain of a stakeholder such that high assurance of policy enforcement is desired. However, as usage control is such pervasive that it happens in open and general-purpose platforms and "usable security" mechanism is strongly desired for cost-effective objective. For example, leveraging local host access control mechanism to enforce usage control policy is desirable if the mechanism can be trusted to do the "right" thing. That is, the goal of pervasive usage control is not to provide a perfect solution for security but to be "good-enough" [22].

Requirement 2: Need a comprehensive policy model Traditional security systems distinguish policy and mechanism [15]. However, early policy systems such as Bell-LaPadula [6] and Biba [7] are too restrictive for convenient use of applications. They support simple policies such as one-way information flow but provide insufficient and inflexible support for general data and application integrity, separation of duty and least privilege requirements. Besides these, usage control considers many constraints or conditional restrictions such as time and location as aforementioned. Traditional policy models cannot supports these and we need a comprehensive policy model to support the variants of security requirements.

Requirement 3: Need MAC mechanism for trusted subsystem on a target platform As in discretionary access control model (DAC), a root-privileged subject has the capability to change security configuration of the whole system such that the subsystem can be compromised either by malicious user or software. For example, a virus or worm can obtain the root permission of a system by exploring some vulnerabilities, e.g., with buffer-over-flow attacks. Thus, mandatory access control (MAC) mechanism is needed. For example, with SELinux, one can label the applications and all resources of a subsystem with a particular domain and define policies to control the interactions between this domain and others for isolation and information flow control purposes.

Requirement 4: Need a polity transformation mechanism from high level usage control policies to concrete MAC policies Typically, a stake holder's policy is specified in different format and semantics from those of the MAC policies on a target platform. For example, a stakeholder can be implemented as a Web Service, where a security policy is specified in XACML. This policy has to be transformed to a concrete policy that can be enforced on a target platform, which follows its local MAC policy model. Efficient and convenient policy transformation mechanism is needed such that security properties are preserved during the transformation, which means, the allowable permissions and information flows are the same in the policies before and after a transformation.

Requirement 5: Build trust chain for policy enforcement from *boot to applications* High assurance of a subsystem in a remote computing platform should origin from a root-of-trust, and then is extended to other system components through which policy enforcement mechanism is built. Typically, MAC mechanism is implemented in the kernel of the operating system (OS) on a platform. Thus, a trusted subsystem should include a trusted kernel and any other components booted before kernel, such as BIOS and boot loader. To obtain the trust of a MAC mechanism in a trusted subsystem, any other supporting components should also be trusted, including policy transformation and management, subject and object attribute acquisition, and the reference monitor itself. The fundamental goal of this trust chain is to achieve a trusted runtime environment for object access where the integrity of all related parts can be verified by a stakeholder.

Requirement 6: Built trusted subsystem with minimum trusted computing base Related to above requirement, to build practical and usable trusted subsystem, minimum trusted computing base (TCB) is desired. TCB includes all the components in the trust chain for policy enforcement during runtime. A large component in this chain results in high cost both on system development and verification since each trusted component requires detailed verification of the software implementation.

Our work follows these principles. Specifically, we propose a platform architecture with mandatory and minimum components. Our implementation is built with emerging trusted computing technologies with hardware-based root-of-trust. We leveraged the MAC mechanism in SELinux for policy enforcement, and we also develop a policy transformation mechanism from high level XACML policies to SELinux policies with an extended MAC policy model. Due to space limit, we just give an high-level view of our platform architecture in next section.

3. PLATFORM ARCHITECTURE

A trusted subsystem includes a root-of-trust, trust chain, and policy transformation and enforcement mechanism, and runtime integrity measurement mechanism. Figure 2 shows a target platform architecture to enforce usage control policies. The hardware layer includes a Trusted Platform Module (TPM), a Core Root of Trust Measurement (CRTM), and other devices. The TPM and CRTM provide the hardware-based root-of-trust for the whole platform. Similar to trusted or authenticated boot [11, 21], the booting components of the platform, including BIOS, boot loader, and operating system (OS) kernel, are measured and their integrity values are stored in particular Platform Configuration Registers (PCRs) of the TPM. Specifically, according to TCG specification [2], the CRTM is the first component to run when the platform boots. It measures the integrity of BIOS before BIOS starts, which in turn, measures the boot loader and then in turn the kernel and kernel modules, recursively. Along this booting and measurement sequence, particular PCR(s) are extended with the measured values, and the result is denoted as PCR_{boot}. TPM guarantees that PCR_{boot} is reset once the platform re-boots.

Upon a user's request on the target platform, a client application (e.g., a healthcare client software) is invoked to communicate with a data owner/provider to obtain an object. At the same time, a policy can be downloaded by the client application from a stakeholder, which can be the same as the data provider or not. For example, a data provider can delegate its policy specification and enforcement to a security service provider, which is the policy stakeholder when an object is downloaded and processed in a client platform.

When a usage control policy (e.g., an XACML policy file) is

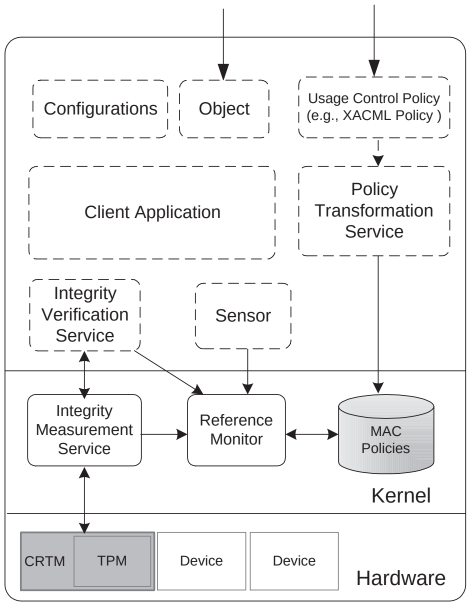
Configurations
Object
Usage Control Policy (e.g., XACML Policy)
Client Application
Policy Transformation Service
Integrity Verification Service
Sensor
Integrity Measurement Service
Reference Monitor
MAC Policies
Kernel
CRTM
TPM
Device
Device
Hardware

Figure 2: Platform architecture for usage control policy enforcement

downloaded from its stakeholder, it is transformed by the policy transformation component to MAC policies such that they can be enforced by the reference monitor. The client application is the target process that can manipulate the object and is to be protected by MAC policies. Also, MAC policies should also include the rules to control accesses to the object from others and any configurations for the client application and the overall security system (e.g., local security policy management).

As aforementioned, usage control policies typically include environmental authorization factors such as time and location. A sensor is a component that sends these environmental information to the policies and thus can be considered. For example, in a mobile application where a service only can be accessed in a particular location, the sensor reports the physical (e.g., through a cellular network provider or GPS) or logical (e.g., through a Wi-fi access point) location of the device, such as home, office, and an airport.

In the kernel level of the platform, the reference monitor captures every possible access attempt to the object and queries the MAC policies before allows the access. A fundamental requirement for the reference monitor is that it has capture all kinds of access attempts, from storage in the local file system to the memory space of the object. Also, the reference monitor controls the interactions between the client application and others, locally and remotely, according to loaded MAC policies.

The integrity measurement service (IMS) is a mandatory component in a trusted subsystem, which starts right after the kernel is booted. The main function of IMS is to measure mandatory components which consist of the TCB to enforce usage control policies. All measured events and the integrity values are stored in a measurement list and corresponding PCRs are extended. Particularly,

- The reference monitor is measured after the kernel is booted.
- The client application, object, and configurations are measured right before the client application is invoked.
- The integrity of usage control policy (e.g., XACML policy file downloaded from its stakeholder), policy transformation

service, and the sensor are measured when they are invoked and before run.

- MAC policies are measured when they are loaded, either when the platform boots or during runtime (i.e., loaded by the policy transformation service).
- Any other applications or services that communicate with the client application.

In general, to only allow accesses to a target object from authorized application, and control the information flow between this application and others, IMS should measure not only the policy enforcement services such as policy transformation and platform sensor, but also all applications that interacts with the client application running on the same platform.

As part of policy specifications, integrity verification service verifies corresponding integrity values measured by the IMS and generates inputs to the reference monitor. As a typical example, the client application only can access the target object when its "current" integrity is a known good value, where the current integrity is the one measured by the IMS.

Note that although we use data objects (e.g., files) through this paper, our usage control mechanism is applicable to other types of objects such as messages and streams. The essential requirement for the object is that its authenticity and integrity can be verified such that, as an input for the application on a client platform, the initial state of the platform can be trusted.

4. CONCLUSIONS AND FUTURE WORK

Usage control focuses on the problem of enforcing security policies on a remote client platform with high assurance and verifiable trust. In this paper we identified general security requirements for usage control and proposed a general framework for this problem. The main idea of our approach is to build a trusted subsystem on an open platform such that a policy stakeholder can deploy sensitive data and services on this subsystem. We propose an architecture with a hardware-based TPM as the root-of-trust and consider integrity measurement/verification and other environmental restrictions in our MAC policy model.

We are implementing a prototype system based on a mobile reference platform. We are also exploring automated policy transformations in mobile computing environment. In addition, as our architecture is extensible, extra components can be included in the TCB of a trusted subsystem for increasing security requirements. Particularly we are investigating how to enforce some kind of obligation policies in our architecture.

5. REFERENCES

[1] Fairplay. http://en.wikipedia.org/wiki/FairPlay.

[2] *TCG Specification Architecture Overview*. https://www.trustedcomputinggroup.org.

[3] Windows media digital rights management (DRM). http://www.microsoft.com/windows/windowsmedia /drm/default.aspx.

[4] M. Abadi, M. Burrows, and B. Lampson. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, 1993.

[5] J. P. Anderson. Computer security technology planning study volume II, ESD-TR-73-51, vol. II, electronic systems division, air force systems command, hanscom field, bedford, MA 01730. http://csrc.nist.gov/publications/history/ande72.pdf, Oct. 1972.

[6] D. E. Bell and L. J. LaPadula. Secure computer systems: Mathematical foundations and model. *Mitre Corp. Report No.M74-244, Bedford, Mass.*, 1975.

[7] K. J. Biba. Integrity consideration for secure computer system. Technical report, Mitre Corp. Report TR-3153, Bedford, Mass., 1977.

[8] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 164–173, Oakland, CA, May 1996.

[9] D. E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5), May 1976.

[10] Department of Defense National Computer Security Center. *Department of Defense Trusted Computer Systems Evaluation Criteria*, December 1985. DoD 5200.28-STD.

[11] J. Dyer, M. Lindemann, R. Perez, R. Sailer, L. van Doorn, S. W. Smith, and S. Weingart. Building the ibm 4758 secure coprocessor. *IEEE Computer*, (10):57–66, 2001.

[12] M. H. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communication of ACM*, 19(8), 1976.

[13] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid. Access control meets public key infrastructure, or: assigning roles to strangers. In *Proc. of IEEE Symposium on Security and Privacy*, pages 2–14, 2000.

[14] M. Hilty, D. Basin, and A. Pretschner. On obligations. In *Proc. of 10th European Symp. on Research in Computer Security*, September 2005.

[15] B. Lampson. Computer security in the real world. *IEEE Computer*, (6):37–46, June 2004.

[16] B.W. Lampson. Protection. In *5th Princeton Symposium on Information Science and Systems*, pages 437–443, 1971. Reprinted in *ACM Operating Systems Review* 8(1):18–24, 1974.

[17] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Proc. of IEEE Symposium on Security and Privacy*, pages 114–130, 2002.

[18] P. Loscocco, S. Smalley, P. Muckelbauer, R. Taylor, J. Turner, and J. Farrell. The inevitability of failure: The flawed assumption of computer security in modern computing environments. In *Proceedings of the National Information Systems Security Conference*, October 1998.

[19] J. Park and R. Sandhu. The UCON$_{\mathrm{abc}}$ usage control model. *ACM Transactions on Information and Systems Security*, 7(1):128–174, February 2004.

[20] A. Pretschner, M. Hilty, and D. Basin. Distributed usage control. *Communications of the ACM*, (9):39–44, September 2006.

[21] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *USENIX Security Symposium*, pages 223–238, 2004.

[22] R. Sandhu. Good-enough security: Toward a pragmatic business-driven discipline. *IEEE Internet Computing*, (1):66–68, January/February 2003.

[23] R. Sandhu, K. Ranganathan, and X. Zhang. Secure information sharing enabled by trusted computing and PEI models. In *Proceedings of ACM Symposium on Information, Computer, and Communication Security*, Taipei, Taiwan, March 21-24 2006.