

Trust-Oriented Access Control Based on Sources of Information Flow

Hui FENG, Weinan LI, Wenchang SHI, Zhaohui LIANG, Bin LIANG

Key Laboratory of Data Engineering and Knowledge Engineering(Renmin University of China), MOE, Beijing, China

School of Information, Renmin University of China, Beijing, China

{fenghui, lwansl023, wenchang, lzh, liangb} @ruc.edu.cn

Abstract— To equip a computer system with trust measurement capability, measurement mechanisms must be built into the system. Access Control is indispensable to ensure effective work of such mechanisms. Existing access control models are not good enough to support trust measurement because they were not devised with this goal in due consideration. Trust is often evaluated in term of integrity, which can be naturally measured using information flow. To support trust measurement, this paper proposes an access control model called Trust-oriented Access Control based on Sources of Information Flow (TACSIF). It uses sources of information flow to describe the integrity level of an entity, which is the destination of that flow. Integrity levels of both subjects and objects are fundamental elements for TACSIF to make access authorization. They are used to define access control rules, which form access control policies of the TACSIF. The TACSIF enforces access control in accordance with its access control policies. To improve its applicability, the TACSIF introduces the concept of constrained subjects to handle network information flows. By embedding trust measurement elements into the model, the TACSIF may provide a good way to support implementation of system mechanisms for trust measurement, especially for one that is based on information flow.

Keywords— Access Control, Information Flow, Integrity, Trust, Measurement

I. INTRODUCTION

In computer system environments, trust is often evaluated in term of integrity. It is a natural way to evaluate integrity of a system using system information flow. To support trust measurement in operating systems, this paper proposes an access control model, called Trust-oriented Access Control based on Sources of Information Flow, or TACSIF for short, in which Sources of Information Flow is critical to integrity. On the one hand, Sources of Information Flow is used to describe integrity levels of system entities. On the other hand, integrity levels are dynamically updated to reflect the practical information flows in a system. In TACSIF, access control rules are developed mainly based on integrity levels. Access authorizations are granted according to the integrity levels of subjects and objects in a system. When an information flow is moving on, the system updates integrity levels of subjects or objects according to the Update rules. Furthermore, access control policies are generated in accordance with the access

control rules. Since usability is important as well as security in an access control model, we try to increase usability of TACSIF by introducing the concept of Integrity Constrained Subject. We add access control rules and policies to handle network information flows on the basis of this concept.

The rest of this paper is organized as follows. We first give the basic definitions of TACSIF in section 2, and then present the detailed architecture of TACSIF in section 3, discussing the primary access control rules and policies. In order to increase its applicability, we introduce the concept of Constrained Subject in section 4, adding relevant access control rules and policies. Related works about access control are reviewed in section 5. Finally, we summarize the paper in section 6.

II. BASIC DEFINITIONS OF TACSIF

Various approaches have been proposed to describe information flow. For example, PRIMA describes information flow by analysing system security policies [1]; BIND uses code segments as extra output information to record information flow in the system [2]; Panorama defines a series of training sets and depicts real information flow in a system via capturing system operations [3]. In this paper we adopt the following definitions used by TACSIF.

A. Information Flow and Source of Information Flow

Information flow refers to the transferring of information from one entity to another. In an information flow, the entity from which information comes out is called Source of Information Flow (SIF), the one into which information goes is called Destination of Information Flow (DIF). If there is no entity in between the SIF and the DIF of an information flow, this information flow is called a direct information flow from the SIF to the DIF. Otherwise, that information flow is called an indirect information flow from the SIF to the DIF. If an entity is the DIF of an information flow, we call the SIF of the information flow as an SIF of the entity. All SIFs of an entity form the Set of SIF of the entity, for simplicity, which may be called the entity's SIF Set.

B. Access Operations

We define two kinds of abstract operations: *observe* and *modify*. The abstract *observe* operation includes the traditional

read and *execute* operations, the *receive* operation in socket communication, and the *read* operation in inter-process communication. The abstract *modify* operation comprises the traditional *write*, *create* and *fork* operations, the *send* operation in socket communication, and the *write* operation in inter-process communication.

We use $observe(s,o)$ to denote that subject s performs *observe* operation on object o , and use $modify(s,o)$ to denote that subject s performs *modify* operation on object o .

For any two entities in a system denoted as e_1 and e_2 , we use $iflow(e_1,e_2)$ to denote direct information flow that transfers from e_1 to e_2 . Apparently,

$$iflow(e_1,e_2) \Rightarrow observe(e_2,e_1) \vee modify(e_1,e_2)$$

C. Integrity Level of An Entity

In TACSIF, the integrity level of an entity is defined as a set that consists of all SIFs of the entity. In other words, the integrity of the entity is defined as the entity's SIF Set.

Generally, the more elements are there in the entity's SIF Set, the lower is the integrity level of the entity. On the contrary, the fewer elements are there in the SIF Set, the higher is the integrity level.

We use $L_i(e)$ to denote the integrity level of e , where e is an entity in a system.

D. Lower Bound and Upper Bound of an Entity's SIF Set

The lower bound of an entity's SIF Set is defined to be the set that consists of all SIFs of the entity at the time when there are minimal information flows transferring to that entity. It represents the highest integrity level of that entity, which is denoted as $low_bound(e)$, where e is the entity.

Similarly, the upper bound of an entity's SIF Set is defined to be the set that consists of all SIFs at the time when there are maximal information flows transferring to that entity. It represents the lowest integrity level of that entity, which is denoted as $up_bound(e)$, where e is the entity.

The integrity level of any entity e satisfies the constraint condition below:

$$low_bound(e) \subseteq L_i(e) \subseteq up_bound(e).$$

E. Dominate Relation

Suppose that e_1 and e_2 are two entities, whose integrity levels are $L_i(e_1)$ and $L_i(e_2)$ respectively. Let's define the dominate ($\succ$) relation as follow:

$$L_i(e_1) \succ L_i(e_2) \text{ if and only if } L_i(e_1) \subseteq L_i(e_2).$$

It means that $L_i(e_1)$ dominates $L_i(e_2)$, indicating that entity e_1 has a higher integrity level than entity e_2 .

Suppose that two entities e_x and e_y have integrity levels $L_i(e_x)$ and $L_i(e_y)$ respectively. We use $H(L_i(e_x), L_i(e_y))$ to denote the least upper bound of $L_i(e_x)$ and $L_i(e_y)$.

F. Threshold Integrity Level and Current Integrity Level

To identify the integrity level of an entity distinctively, we introduce the terms of Threshold Integrity Level (TIL) and Current Integrity Level (CIL). The TIL of an entity is identical

to the integrity level of the entity. The CIL of an entity is defined to be a set that consists of all SIFs that not only are elements of the TIL of the entity but also have already transferred information to the entity.

We use $L_{it}(e)$ to denote the TIL of entity e . $L_{it}(e)$ is the lowest integrity level of e , which implies the highest degree to which entity e may be contaminated.

We use $L_{ic}(e)$ to denote the CIL of entity e . $L_{ic}(e)$ is the real-time integrity level of e , which reflects the degree to which entity e may be contaminated currently.

G. Update of an Entity's Integrity Level

Updating an entity's integrity level means that after the transferring of an information flow, the system dynamically changes the integrity level of the DIF of the information flow.

H. Current Access

A current access refers to an access that has been performed by the system. A set that consists of all the current accesses is called a Current Access Set. We use A to denote an operation set, namely,

$$A = \{read, write, exec, fork\}.$$

We use b to denote a current access set, namely,

$$b = \{(s,o,a) \mid s \in S \wedge o \in O \wedge a \in A, s \text{ operates on } o\}.$$

Where, S is the set of subjects and O is the set of objects. $(s,o,read)$ means s reads from o ; $(s,o,write)$ means s writes to o ; $(s,o,exec)$ means s executes o ; $(s,fork)$ means s forks a new child process.

Also, we use $request(s,o,a)$ to denote that subject s sends a request of performing operation a on o to the system. We use $allow(s,o,a)$ to denote that the system authorizes s to perform operation a on o . Specifically, $request(s,fork)$ is used to denote that s requests to create a process, and $allow(s,fork)$ is used to denote that the system authorizes s to create a process.

III. ARCHITECTURE OF TACSIF

When subject s observes object o , information flow is from o to s . On this occasion, TACSIF requires that the integrity level of o be higher than that of s . On the contrary, when s modifies o , information flow is from s to o , TACSIF requires that the integrity level of s be higher than that of o .

A. Access Control Rules

Rule 1 (read): $allow(s,o,read) \Rightarrow L_{ic}(o) \succ L_{it}(s)$ [End of Rule 2]

This rule means that before the system authorizes the *read* request, it must check whether the CIL of object o dominates the TIL of subject s . Only if that condition holds, s is allowed to read o .

Rule 2 (exec): $allow(s,o,exec) \Rightarrow L_{ic}(o) \succ L_{it}(s)$ [End of Rule 2]

This rule means that before the system authorizes the *exec* request, it must check whether the CIL of object o dominates the TIL of subject s . Only if that condition holds, s is allowed to execute o .

Rule 3 (update after read): After the $allow(s,o,read)$ authorization is granted, s reads o . If $L_{ic}(o) \succ L_{ic}(s)$, the CIL of s is not changed after the $read$ operation. If $\neg(L_{ic}(o) \succ L_{ic}(s)) \wedge (L_{ic}(o) \succ L_{it}(s))$, the CIL of s is updated in this way: $L_{ic}(s) = H(L_{ic}(o), L_{ic}(s))$. [End of Rule 3]

This rule means that after the $read$ operation, if the CIL of the object doesn't dominate the CIL of the subject, the system must update the integrity level of the subject to reveal the information flow. The system adds the SIFs in the CIL of the object to the CIL of the subject.

Rule 4 (update after exec): After the $allow(s,o,exec)$ authorization is granted, s executes o . Suppose that s becomes s' after the $exec$ operation. The CIL of s' is set in this way:

$$L_{ic}(s') = H(L_{ic}(o), L_{ic}(s)) \cup \{s'\},$$

Also, the TIL of s' is set in this way:

$$L_{it}(s') = H(L_{it}(o), L_{it}(s)) \cup \{s'\}. \text{ [End of Rule 4]}$$

This rule indicates that we need only to consider the integrity level of s' after the $exec$ operation. The system uses the integrity levels of the subject and the object to update the CIL and the TIL of s' . We can see that the update after the $exec$ operation is different from that after the $read$ operation.

Rule 5 (write): $allow(s,o,write) \Rightarrow L_{ic}(s) \succ L_{it}(o)$ [End of Rule 5]

This rule indicates that before the system authorizes the $write$ request, it must check whether the CIL of subject s dominates the TIL of object o . Only if that condition holds, s is allowed to $write$ to o .

Rule 6 (update after write): After the $allow(s,o,write)$ authorization is granted, s writes to o . If $L_{ic}(s) \succ L_{ic}(o)$, the CIL of o stays unchanged after the $write$ operation. If $\neg(L_{ic}(s) \succ L_{ic}(o)) \wedge (L_{ic}(s) \succ L_{it}(o))$, the CIL of o is updated in this way: $L_{ic}(o) = H(L_{ic}(o), L_{ic}(s))$. [End of Rule 6]

This rule means that after the $write$ operation the system uses the CIL of the subject to extend the CIL of the object. It adds the SIFs in the CIL of the subject to the CIL of the object to record the new information flow and reveal the system's current state.

Rule 7 (update after fork): Suppose that the CIL of subject s_1 is $L_{ic}(s_1)$ and s_1 is changed to be s_2 after it performs the $fork$ operation. The CIL of s_2 is set in this way: $L_{ic}(s_2) = L_{ic}(s_1) \cup \{s_2\}$, and the TIL of s_2 is set in this way: $L_{it}(s_2) = L_{it}(s_1) \cup \{s_2\}$. [End of Rule 7]

B. Access Control Policies

Based on the above access control rules and update rules, we can construct the access control policies of TACSIF.

Policy 1: When a $request(s,o,read)$ request is generated, it is processed as follows.

If $L_{ic}(o) \succ L_{ic}(s)$, the system will authorize the request, allowing s to read o .

If $\neg(L_{ic}(o) \succ L_{ic}(s)) \wedge (L_{ic}(o) \succ L_{it}(s))$, the system will authorize the request, and the CIL of s is updated in this way: $L_{ic}(s) = H(L_{ic}(o), L_{ic}(s))$.

Otherwise, the $request(s,o,read)$ request is denied. [End of Policy 1]

Policy 2: When a $request(s,o,exec)$ request is generated, it is processed as follows.

If $L_{ic}(o) \succ L_{it}(s)$, the system will authorize the request, allowing s to execute o . After the $exec$ operation, s is changed to a new version, which may be denoted as s' . The CIL of s' is set in this way: $L_{ic}(s') = H(L_{ic}(o), L_{ic}(s)) \cup \{s'\}$. The TIL of s' is set in this way: $L_{it}(s') = H(L_{it}(o), L_{it}(s)) \cup \{s'\}$.

Otherwise, the $request(s,o,exec)$ request is denied. [End of Policy 2]

Policy 3: When a $request(s,o,write)$ request is generated, it is processed as follows.

If $L_{ic}(s) \succ L_{ic}(o)$, the system will authorize the request, allowing s to write to o .

If $\neg(L_{ic}(s) \succ L_{ic}(o)) \wedge (L_{ic}(s) \succ L_{it}(o))$, the system will authorize the request, and the CIL of o is updated in this way: $L_{ic}(o) = H(L_{ic}(o), L_{ic}(s))$.

Otherwise, the $request(s,o,write)$ request is denied. [End of Policy 3]

Policy 4: When a $request(s,fork)$ request is generated, it is processed as follows.

The system authorizes the request, allowing s to create a new process, denoted as s' . Then both the CIL and TIL of s' are set. The CIL of s' is set in this way: $L_{ic}(s') = L_{ic}(s) \cup \{s'\}$, and the TIL of s' is set in this way: $L_{it}(s') = L_{it}(s) \cup \{s'\}$. [End of Policy 4]

IV. THE CONCEPT OF INTEGRITY CONSTRAINED SUBJECTS

In order to receive and transfer some special information, such as network data, we introduce the concepts of Integrity Constrained Subject and Integrity Constrained Set. They are used to verify the influence of certain SIFs on subjects' integrity level so as to support reasonable information exchange request.

A. Integrity Constrained Subject/Set

Let s_i be a subject in a system, $set_i(s_i)$ be a set of system entities. For any element e_x in $set_i(s_i)$, there exists an information flow from e_x to s_i . If none of this kind of information flows may compromise the integrity of s_i , then we call s_i an Integrity Constrained Subject (ICSub), and $set_i(s_i)$ an Integrity Constrained Set (ICSet) corresponding to s_i . For simplicity, we use S_i to denote the set that consists of all ICSubs in a system.

ICSubs are actually a kind of privileged subjects that may increase their integrity levels in some sense. Each of them can keep its access rights after receiving information that comes from entities in a corresponding ICSet.

In a specific system, ICSubs and ICSetS are determined as a part of initialization when the system is started up. The ICSet corresponding to an ICSub confines the range of integrity levels that the ICSub may increase. An ICSub can not increase its integrity level arbitrarily. It must pass the system's measurement before being allowed to increase its integrity level.

We use $measure(s)$ to denote the result given by the system after it measures the integrity of subject s . The value of the result may be either 1 or 0, where 1 indicates sound integrity and 0 indicates compromised integrity.

That $measure(s_i)$ equals to 1 implies that s_i has not been contaminated, which means that s_i is trusted and the *read* operation that s_i performs may be trusted as well. Therefore, s_i is allowed to increase its integrity level. On the contrary, that $measure(s_i)$ equals to 0 implies that s_i is not trusted and hence the *read* operation that s_i performs is trusted neither. In this case, s_i is not allowed to increase its integrity level. Suppose that an ICSub s_i executes a certain piece of executable code. If the content of the code is modified, s_i may be regarded as having become a different subject, which may be denoted as s_i' . If $measure(s_i')$ equals to 1, s_i' is trusted and may be allowed to increase its integrity level. If $measure(s_i')$ equals to 0, s_i' is untrusted and is not allowed to increase its integrity level.

B. Extended Rules

Rule 8 (update after ICSub's read):

Suppose that s_i is an ICSub and o is an object. The rule presented as $allow(s_i, o, read) \Rightarrow L_{ic}(o) \succ L_{it}(s_i)$ is enforced. After the *read* operation, if $measure(s_i)$ equals to 1, the CIL of s_i is updated in this way:

$$L_{ic}(s_i) = H((L_{ic}(o) - set_i(s_i)), L_{ic}(s_i)).$$

If $measure(s_i)$ equals to 0, the CIL of s_i is updated in this way: $L_{ic}(s_i) = H(L_{ic}(o), L_{ic}(s_i))$.

After the ICSub reads from an object, the system checks whether the subject is contaminated. If it is not contaminated, the system updates integrity level of the subject, adding the SIFs in the CIL of the object to the CIL of the subject, except those that are also in the ICSet. [End of Rule 8]

Rule 9 (update after ICSub's exec):

Suppose that s_i is an ICSub and o is an object. The rule presented as $allow(s_i, o, exec) \Rightarrow L_{ic}(o) \succ L_{it}(s_i)$ is enforced. After the *exec* operation, s_i is changed. The changed version of s_i is denoted as s_i' . If $measure(s_i')$ equals to 1, the CIL of s_i' is updated in this way:

$$L_{ic}(s_i') = H((L_{ic}(o) - set_i(s_i)), L_{ic}(s_i)) \cup \{s_i'\}$$

and the TIL of s_i' is updated in this way:

$$L_{it}(s_i') = H(L_{it}(o), L_{it}(s_i)) \cup \{s_i'\}.$$

If $measure(s_i')$ equals to 0, the CIL of s_i' is updated in this way: $L_{ic}(s_i') = H(L_{ic}(o), L_{ic}(s_i)) \cup \{s_i'\}$, and the TIL of s_i' is updated in this way: $L_{it}(s_i') = H(L_{it}(o), L_{it}(s_i)) \cup \{s_i'\}$. [End of Rule 9]

C. Extended Policies

Policy 5: When a $request(s_i, o, read)$ request is generated, it is processed as follows.

If $L_{ic}(o) \succ L_{ic}(s_i)$, the system authorizes the request, allowing s_i to read o .

If $\neg(L_{ic}(o) \succ L_{ic}(s_i)) \wedge (L_{ic}(o) \succ L_{it}(s_i))$, the system authorizes the request and measures the integrity of s_i . If $measure(s_i)$ equals to 1, the CIL of s_i is updated in this way:

$$L_{ic}(s_i) = H((L_{ic}(o) - set_i(s_i)), L_{ic}(s_i)).$$

If $measure(s_i)$ equals to 0, the CIL of s_i is updated in this way: $L_{ic}(s_i) = H(L_{ic}(o), L_{ic}(s_i))$.

Otherwise, the $request(s_i, o, read)$ request is denied. [End of Policy 5]

Policy 6: When a $request(s_i, o, exec)$ request is generated, it is processed as follows.

If $L_{ic}(o) \succ L_{it}(s_i)$, the system authorizes the request. After the *exec* operation, subject s_i is changed to a new version, which may be denoted as s_i' . If $measure(s_i')$ equals to 1, the CIL of s_i' is updated in this way:

$$L_{ic}(s_i') = H((L_{ic}(o) - set_i(s_i)), L_{ic}(s_i)) \cup \{s_i'\}$$

and the TIL of s_i' is updated in this way:

$$L_{it}(s_i') = H(L_{it}(o), L_{it}(s_i)) \cup \{s_i'\}.$$

If $measure(s_i')$ equals to 0, the CIL of s_i' is updated in this way: $L_{ic}(s_i') = H(L_{ic}(o), L_{ic}(s_i)) \cup \{s_i'\}$, and the TIL of s_i' is updated in this way: $L_{it}(s_i') = H(L_{it}(o), L_{it}(s_i)) \cup \{s_i'\}$.

Otherwise, the $request(s_i, o, exec)$ request is denied. [End of Policy 6]

V. RELATED WORKS

Access control is fundamental to operating system security. Lots of research achievements have been obtained both in confidentiality protection and integrity protection [4-7]. The BIBA model is a classic access control model in integrity protection [5], based on which plenty of work has been carried out. Security is one aspect of a system and usability is another. Among the design principles of the UMIP model, the first one is "good enough security with a high level of usability, rather than better security with a low level of usability" [8]. It is essential that a model trades off between security and usability. Traditional access control models often show weakness in practical implementation. To increase security at the cost of sacrificing usability is unacceptable for users. At present, a trend of research in the field of access control model is combining security with usability.

This paper is inspired by the thought that information flow is closely related to system integrity. We try to build a trust-oriented access control model with considerations to information flow and the balance of security and usability.

Based on existing access control models, our TACSIF model inherits some key principles of security policies of the CW-Lite model [9] and formulates dynamic update rules of

integrity levels. It explores a more flexible way to implement information flow based access control.

The thought of SIF is in some sense inspired by the DLM model [10]. DML uses the set of readers of data to describe confidentiality level of data. It can flexibly control users' access to system objects. DLM focuses on protecting data confidentiality in decentralized environments. It uses user sets to enable multiple users to control access to data simultaneously.

The IFEDAC model is a discretionary access control model [11]. It employs access rights of discretionary access control to describe integrity levels of subjects and objects, and uses traditional information flow tracking techniques to update integrity levels. In order to improve its practicability, the model introduces some exception processes and uses them to receive network data. TACSIF absorbs some spirits of IFEDAC in defining integrity levels to enhance usability. Although both TACSIF and IFEDAC enforce the low watermark concept in defining access control policies, the former is a mandatory access control model, while the latter is a discretionary one. Based on SIFs, TACSIF is designed to support system integrity measurement. To enhance discretionary access control, IFEDAC uses information flow to conduct dynamic access control, but whether an access may be authorized is determined by the owner. In this model information flow primarily serves to change integrity levels. In order to deal with the issue of receiving network data, TACSIF introduces the concept of ICSubs and ICSets and develops relevant access control rules and policies.

VI. CONCLUSIONS

Governed by access control models, access control mechanisms of an operating system is of paramount significance for the operating system to facilitate trust measurement. It is hard for existing access control models to be adapted well enough to effectively support trust measurement mechanisms. To handle this situation, this paper proposes a new access control model called TACSIF with regards to trust measurement, where trust is measured in terms of integrity. Contributions of the paper may be summed up in three points. First, it presents an exploration to formulate integrity levels of system entities via sources of information flow. Second, it provides a way to support dynamic change of integrity levels of system entities when a system is running in order to improve flexibility. Third, it introduces the concept of integrity constrained subjects to relax the strictness of traditional access control models so as to balance security and

usability, and hence improve practicability of the model. By theoretical analysis, we may demonstrate that TACSIF is of good security characteristics. With prototype implementation, we may illustrate that applying TACSIF to real system is of good feasibility. Due to space limitation of the paper, those are not discussed in detail. The weakness of the current TACSIF model is that it is of certain difficulties in efficiently determining integrity levels of entities in very large distributed systems. Solutions to this issue will be explored in the future.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their insightful comments that helped improve the presentation of this paper. The work of the paper was supported in part by National Natural Science Foundation of China (61070192, 91018008, 60873213, 60703103), National 863 High-Tech Research Development Program of China (2007AA01Z414), Natural Science Foundation of Beijing (4082018) and Open Project of Shanghai Key Laboratory of Intelligent Information Processing (IPL-09-006).

REFERENCES

- [1] T. Jaeger, R. Sailer, U. Shankar, "PRIMA: Policy Reduced Integrity Measurement Architecture," in Proc. SACMAT'06, 2006.
- [2] E. Shi, A. Perrig, L. V. Doorn, "Bind: A Fine-Grained Attestation Service for Secure Distributed Systems," Proceedings of the 2005 IEEE Symposium on Security and Privacy, 2005.
- [3] H. Yin, D. Song, M. Egele, et al, "Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis," The 14th ACM Conference on Computer and Communications Security (CCS), 2007.
- [4] David E. Bell and Leonard J. LaPadula, "Secure Computer System: Unified Exposition and MULTICS Interpretation," MTR-2997 Rev. 1, The MITRE Corporation, Bedford, MA, USA, Mar 1976.
- [5] K. J. Biba, "Integrity Considerations for Secure Computer Systems," ESD-TR-76-372, Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Bedford, MA, USA, Apr 1977.
- [6] R. Sandhu, E. Coyne, H. Feinstein, C. Youman, "Role-Based Access Control Models," IEEE Computer, Vol.29, No.2, Feb 1996.
- [7] J. Park, R. Sandhu, "The UCONABC Usage Control Model," Transactions on Information and System Security (TISSEC), 2004.
- [8] N. Li, Z. Q. Mao, H. Chen, "Usable Mandatory Integrity Protection for Operating Systems," The IEEE Symposium on Security and Privacy, 2007.
- [9] U. Shankar, T. Jaeger, R. Sailer, "Toward Automated Information-Flow Integrity Verification for Security-Critical Applications," Proceedings of the 13th Annual Network and Distributed System, 2006.
- [10] A. C. Myers, B. Liskov, "Protecting Privacy Using the Decentralized Label Model," ACM Transactions on Software Engineering and Methodology (TOSEM), 2000.
- [11] N. Li, Z. Q. Mao, H. Chen, et al, "Trojan Horse Resistant Discretionary Access Control," in Proc. SACMAT'09, 2009.