Journal of Network and Computer Applications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Security model oriented attestation on dynamically reconfigurable component-based systems

Liang Gu, Guangdong Bai, Yao Guo *, Xiangqun Chen, Hong Mei

Key Laboratory of High Confidence Software Technologies (Ministry of Education), School of Electronics Engineering and Computer Science, Peking University, Beijing 100871. China

ARTICLE INFO

Article history:
Received 1 August 2010
Received in revised form
14 January 2011
Accepted 9 March 2011

Keywords:
Remote attestation
Component-based systems
Security model
Security policy
Dynamically reconfigurable CBS

ABSTRACT

As more and more component-based systems (CBS) run in the open and dynamic Internet, it is very important to establish trust between clients and CBS in mutually distrusted domains. One of the key mechanisms to establish trust among different platforms in an open and dynamic environment is remote attestation, which allows a platform to vouch for its trust-related characteristics to a remote challenger. This paper proposes a novel attestation scheme for a dynamically reconfigurable CBS to reliably prove whether its execution satisfies the specified security model, by introducing a TPM-based attestation service to dynamically monitor the execution of the CBS. When only parts of the dynamic CBS are concerned, our scheme enables fine-grained attestation on the execution of an individual component or a sub-system in the dynamic CBS, such that it involves only minimal overhead for attesting the target parts of the CBS. With flexible attestation support, the proposed attestation service can attest a CBS at the granularity from an individual component to the whole CBS. As a case study, we have applied the proposed scheme on OSGi systems and implemented a prototype based on JVMTI for Felix. The evaluation results show that the proposed scheme is both effective and practical.

1. Introduction

Thanks to the achievements from both academic and industrial organizations in the past few years, component-based systems (CBS) have been widely applied in various applications, among them many are complex mission-critical systems. With the rapid development of Internet, many dynamically configurable CBS nowadays are deployed on open computer platforms across heterogeneous domains or over the public Internet, such as systems based on CORBA, .NET, J2EE and Web Services. These dynamic CBS can be configured dynamically and the components in CBS can be updated and replaced at runtime. The security and privacy of its clients greatly rely upon the sound operation of these systems, thus the trustworthiness of the execution of dynamic CBS is especially important.

In an open and dynamic environment, a client or user often pays close attention to whether the computation results of a software component in a dynamic CBS are of integrity, or a specific dynamic CBS runs as expected. For example, when an end user submits his personal information, such as a password or a credit card number, he may require that the corresponding process in the server-side CBS will protect his information properly and no other unauthorized processes are able to obtain it. Meanwhile, a system administrator may want to check whether the execution of a CBS runs as configured. Two aspects are concerned in order to confirm that a dynamic CBS behaves according to a given security model: *What mechanisms are employed to protect the execution of software components and CBS? How to confirm that these employed mechanisms are correctly enforced?*

Many mechanisms have been proposed to enhance the security of CBS (Lindqvist and Jonsson, 1998; Khan and Han, 2002; Sun et al., 2008). However, it is difficult for the existing security mechanisms to establish trust on dynamic CBS in an open and dynamic environment such as the Internet, because of the following reasons:

- First, the root of trust for these traditional security mechanisms would be vulnerable if applied to dynamic CBS. Existing security mechanisms for dynamic CBS are mostly based on pure software. However, software is vulnerable for attacks. Thus these security mechanisms themselves may also suffer from attacks;
- Second, the complexity of dynamic CBS is ever increasing and it may be comprised of third-party components. The system behaviors may be unpredictable and the management of these dynamic CBS becomes more difficult. Furthermore, the vulnerabilities in a component may compromise the whole CBS (Lindqvist and Jonsson, 1998);
- Third, in open networks, the client and dynamic CBS may run in heterogeneous and distributed environments in mutually distrusted domains, thus the traditional trust management

* Corresponding author. Tel.: +86 10 62753496.
E-mail address: yaoguo@sei.pku.edu.cn (Y. Guo).

1084-8045/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2011.03.014

mechanisms based on cryptographic protocols (Blaze et al., 1996; Li et al., 2002) are not adequate to establish trust between clients and the dynamic CBS;
- Furthermore, a dynamic reconfigurable CBS (Hnetynka and Plasil, 2006) can evolve because of component updates, runtime environment changes or user modifications. A CBS administrator may incidentally modify the CBS configuration into a fault state at runtime. The runtime deployment and update of components in CBS may also cause anomalies in the system.

The trusted platform module (TPM) (Trusted Computing Group, 2005) proposed by the trusted computing group (TCG) has received broad interests from both academia and industry. TCG attestation allows a challenging platform, usually referred to as a *challenger*, to verify the configuration integrity of a remote platform (i.e. an *attester*). Recent years have witnessed various evolutions out of the basic TCG attestation in many dimensions (Sailer et al., 2004; Haldar et al., 2004; Jaeger et al., 2006; Chen et al., 2006; Gu et al., 2008a; Kil et al., 2009). We use TCG attestation (Trusted Computing Group, 2005) as a building block to attest the security model for dynamic CBS execution.

In this paper, we propose a novel approach to attest whether the execution of a dynamic CBS is in compliance with the given security model of a challenger. A security model usually depicts the higher level specification which restricts the execution of systems. If a system satisfies a specific security model, it means that the system is at a specific security level. Security models are usually expressed in security policies in systems and the execution of a CBS is usually constrained by these policies. In order to prove whether the execution of a dynamic CBS is at a specific security level, we introduce an attestation service, which leverages the features of TPM, to monitor and record the evidences for attesting the correct enforcement of the security policy. As the dynamic reconfigurable CBS may change its security state rapidly, the proposed attestation service dynamically monitors the security related objects in the system.

Facing the challenges for establishing trust on the dynamic CBS in open environments, our scheme employs several techniques to provide reliable attestation on their executions. For parties from mutually distrusted domains in open networks, our scheme employs TPM as the strong root of trust for attesting dynamic CBS in open environments. With these runtime evidences, a challenger can attest the execution of dynamic CBS in two steps: first, it confirms that the security policy is correctly enforced by the runtime security mechanism; second, it confirms that the enforced security policy of CBS is in compliance with the expected security model. Furthermore, when only an individual software component or a subsystem in the CBS is concerned, our scheme is able to attest its execution by checking itself and other components on which it depends, without attesting the whole CBS. As a result, our scheme is able to provide flexible attestation on dynamically reconfigurable CBS with minimal overhead.

This paper makes the following main contributions:

- To the best of our knowledge, this is the first work for applying TCG-based attestation techniques specifically on dynamically reconfigurable component-based systems to attest the security model of CBS execution.
- The proposed attestation scheme provides a flexible and fine-grained attestation mechanism which leverages the features of component-based software and TCG technologies to reliably attest the execution security of dynamic CBS in open networks.
- With TPM, our scheme has a strong root of trust for trust evaluation on a dynamic CBS. With our scheme, it is reliable to conclude whether a software component or a CBS executes as expected according to the specified security model.
- We applied the proposed scheme on the standard OSGi platforms, and implemented a prototype of attestation service for Felix (Apache Felix, 2010), which is an OSGi framework instance. The performance of the prototype is studied in the case study and it demonstrates our scheme in practical usage.

The rest of the paper is organized as follows: Section 2 introduces the background, including two motivating scenarios, TCG attestation and security model. Section 3 presents our solution for attesting CBS. Section 4 introduces the case study and evaluation of our scheme: attestation on an OSGi system. Section 5 introduces related work and Section 6 concludes the paper.

2. Background

2.1. Motivating scenarios

We will first introduce two typical scenarios for our scheme.

Online shopping system: Many online shopping systems are implemented based on J2EE. At the checking out stage, it usually involves submitting a user's personal information. The consumer can feel more comfortable, if the system can attest that it does not reveal any personal information to untrusted processes in the system. With the help of our attestation scheme, online consumers can request an attestation on all components related to the process, to ensure that his personal information is protected as expected.

CBS administration: The administrator of a CBS may rely on runtime monitoring and reporting mechanisms to check whether the system executions as expected. However, in a dynamic and open environment, the monitoring and reporting mechanisms require a strong root of trust to guarantee their trustworthiness. With the support of attestation, the administrator can evaluate the trustworthiness state of the CBS reliably, and carry out the administration activities more reliably.

2.2. Trusted computing and remote attestation

A series of TCG specifications have been released in the past 10 years and TCG still keeps on updating these specifications according to evolving application requirements. The trusted platform module (TPM) is introduced as the core in these specifications. The TPM is a tamper-resistant module, designed to resist all software attacks and moderate hardware attacks. It encloses a non-volatile storage, a set of platform configuration registers (PCRs) and an engine for cryptographic operations.

TCG remote attestation was introduced to attest the configuration integrity of remote platform. A typical TCG-attestation is carried out as follows. First, the challenger sends the remote attestation request with a random nonce to the attester platform. After receiving the challenge request, the attester platform retrieves the corresponding stored measurement log (SML), and calls TPM to sign the relevant PCR values using the nonce and its attestation identity key (AIK). Then the attester platform collects the credentials vouching for the TPM. The signature on the PCR values and related SML records, together with the credentials, is sent back to the challenger as the attestation response. The challenger verifies the signature and compares the received platform measurements with known-good ones, which can be retrieved from its local storage or a trusted third party.

Existing remote attestation schemes mostly come into three categories: integrity attestation (Sailer et al., 2004; Jaeger et al., 2006; Garfinkel et al., 2003), property-based attestation (Chen et al., 2006; Poritz et al., 2004; Sadeghi and Stble, 2004) and

Please cite this article as: Gu L, et al. Security model oriented attestation on dynamically reconfigurable component-based systems. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.014

semantic attestation (Haldar et al., 2004; Shi et al., 2005; Alam et al., 2008; Gu et al., 2008a, 2008b; Nauman et al., 2009; Gu et al., 2009; Kil et al., 2009; Baiardi et al., 2009). Integrity attestation is based on TCG attestation and mostly tries to attest the configuration integrity of platforms. In order to protect the configuration information of the attested platform, property-based attestation was proposed to attest platforms by checking the specified properties. The property is certificated via checking the configuration state by a trusted third party. The semantic attestation is used for proving some higher level properties of the target system or platform.

2.3. Security model and security policy

A security model is a high level specification or an abstract machine description of what the system does (Goguen and Meseguer, 1982). A security model defines some high level rules for information flow in the system. For information flow security, confidentiality and integrity are the most concerned factors. So most of the existing security models come into two categories: confidentiality model and integrity model. The typical confidentiality model is Bell–La Padula model (1973). Biba model (1977) and Clark–Wilson model (1987) are the most representative integrity models. Other famous security models include role-based access control (RBAC) model (Sandhu, 1998), lattice model of information flow (Denning, 1976). The security level of a system can be obtained by checking whether the information flow in the system satisfies certain security models.

In practical usage, security policies convey specific security models. A security policy is a set of rules governing subjects and objects in system, and it specifically restricts the behaviors of subjects (processes and users) in system (Boutaba and Aib, 2007). For example, it can specify which subjects can access which objects. A security policy defines the security requirements for a given system.

A CBS usually employs access control policies to restrict the interactions among components in a system. So the problem of attesting whether a CBS behaves as expected can be reduced to two sub-problems: whether the security policy satisfies the expected security model; whether the security policy is correctly enforced.

3. Attestation on dynamically reconfigurable component-based systems

Challengers may expect that the target CBS behaves in a specific manner. These kinds of expectations on a CBS usually reflect some higher level specifications on it. Specifically, the security policy for CBS at runtime specifies such kinds of expectation. We will reduce the problem of how to attest a CBS to the problem of how to attest whether the execution of the CBS satisfies a specified security model. Our scheme for attestation on a CBS needs to attest two objectives: whether the security policy enforcement mechanism on the specified CBS is correctly enforced; whether the enforced security policy satisfies the specified security model.

Attestation on CBS involves three phases. In the preparation phase, the attestation objects are identified according to the specific security model and other application requirements. At runtime, attestation service records the execution of relevant parts in the CBS, as well as the security policy enforcement *PE*. At verification time, challenger *CH* can check these proofs to attest whether the expectation is satisfied, i.e., whether the execution of a CBS is in compliance with a given security model.

Threat model: In our scheme, we employ the threat model of TCG attestation (Trusted Computing Group, 2005). We assume that sophisticated physical attacks on the security chip (TPM) are not applicable for ordinary application usages. So the trust on TPM on the host platform of the target CBS can be attested even in the case of other normal physical attacks and software based attacks. In the software layer of the host platform, the execution of the TPM driver should be guaranteed. Based on the TPM, the runtime environment of CBS can be attested accordingly, either statically or dynamically. For the target CBS, the administrator or owner of the host platform can manipulate the states of the CBS, including its components and configurations. The attackers on the host platform can modify the components and configurations in the target CBS.

3.1. Scheme overview

3.1.1. Scheme architecture

The architecture of our scheme is shown in Fig. 1. Two parties are involved: the challenger (*CH*) and the platform (H_r) which hosts the CBS. The challenger can be a remote user of the CBS, a local system administrator, or even a process. The platform H_r is supposed to be equipped with a TPM that serves as the root of trust. The security policy of CBS is enforced by the policy enforcement (*PE*) in its framework layer. *PE* consists of a policy decision point, policy enforcement point and policy records. We introduce an attestation service *AS* with corresponding attestation policy in the runtime environment layer for CBS. The *AS* employs the TPM to dynamically monitor and record the runtime execution of CBS

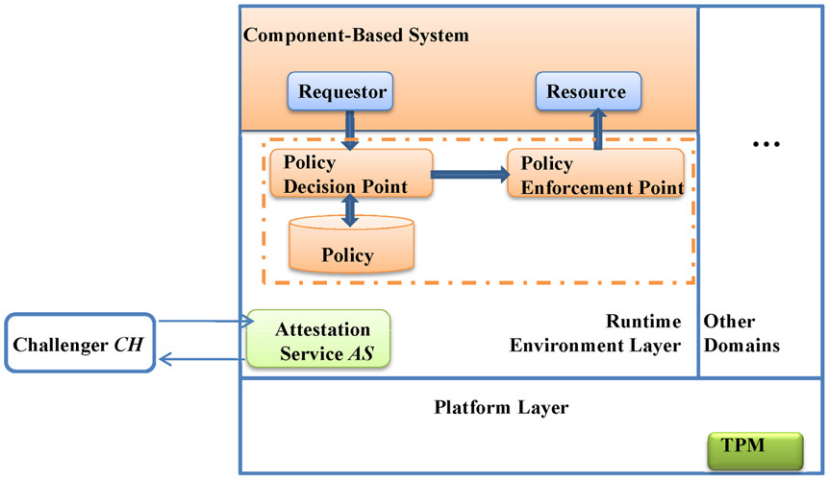

Fig. 1. The architecture for attesting component-based systems.

Please cite this article as: Gu L, et al. Security model oriented attestation on dynamically reconfigurable component-based systems. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.014

and the function of *PE*. In order to support the secure domain for runtime environment layer, the platform layer may employ trusted virtual machine (Garfinkel et al., 2003) to provide an isolated environment. The communications between *CH* and *AS* can be protected by cryptographic protocols.

3.1.2. Attestation service

AS is introduced in the foundational layer in the runtime environment of CBS. *AS* should be able to monitor the execution of CBS and the security policy enforcement mechanism *PE*. According to the specific security model and other attestation requirements, all objects required to be attested are identified as an Attestation Objects List (*AOL*). *AS* is responsible for recording the states and behaviors of these attestation objects. After all required proofs for attestation are collected, *AS* delivers them to the challenger.

The attestation policy enables the attestation service to support flexible attestation according to different application requirements. For example, when parts of the CBS are concerned in a specific attestation, only these related objects are included in the *AOL*. The attestation policy can be configured before the execution of CBS and reconfigured according to the challenger's requests at runtime. Above all, the attestation policy tells the attestation service about which objects are required to be monitored, and how to monitor them for some special applications.

3.2. Security model

A security model consists of two parts: (i) the system characteristic model: the system model for non security related system features; (ii) the security property model: the security objective.

We introduced a general system model according to Goguen and Meseguer (1982) to depict the component behaviors and policy change behaviors in a CBS. A system can be represented as a state machine M, defined as (U, S, SC, Out, $Capt$, CC, f_{out}, f_{do}, f_{cdo}, S_0, t_0):

- U: the set of subjects, which can be users and components in CBS. Its element is denoted as u.
- S: the set of system states, which are determined by the states of all its subjects and objects, and its element is denoted as s.
- SC: the set of state changing commands and its element is denoted as sc.
- Out: the set of all possible outputs.
- $Capt$: the set of capability tables which specify the permission of subjects, and these capabilities are determined by the system policy. Let $C = SC \bigcup CC$, then $Capt = \{U \times S \rightarrow \wp(C)\}$, where $\wp(C)$ is the powerset of C.
- CC: the set of policy state changing commands which change the $Capt$, and its element is denoted as cc.
- $f_{out}(s,Capt,u) : S \times \wp(Capt) \times U \rightarrow Out$, a function, which gives specific outputs when the system is in a specific state s, with a specific user and a specific capability set.
- $f_{do}(s,Capt,u,sc) : S \times \wp(Capt) \times U \times SC \rightarrow S$, a system state change function.
- $f_{cdo}(s,Capt,u,cc) : S \times \wp(Capt) \times U \times CC \rightarrow \wp(Capt)$, a capability set state change function.
- S_0: the initial machine state.
- t_0: the initial state of capability set.

According to the system model, system behaviors can be considered as the execution of a sequence of state change commands (sc or cc). The capability table restricts the runtime behaviors in CBS. The above model can express both static policies and dynamic policies.

The system state set S can be expressed in a more concrete way according to a specific security model. For simplicity, we employ BLP confidentiality model as an example. According to Bell and La Padula (1973), we express S in a more concrete model which concerns mostly about the state of a system, while the policy change behaviors are not specially concerned:

- U: the set of subjects, which can be users and components in CBS. Its element is denoted as u. The set of trusted subjects is $U_T \subseteq U$, and the set of subjects with undetermined trust state can be denoted as U', where $U' \cup U_T = U$.
- O: the set of objects and its element is denoted as o.
- L: the set of security levels and its element is l_k, and all elements in L is partially ordered.
- $f=(f_O,f_U,f_C)$: three level functions: object level function $f_O : O \rightarrow L$, subject level function $f_U : U \rightarrow L$, current subject level function $f_C : U' \rightarrow L$.
- A: access modes $A=\{e,r,a,w\}$ corresponding to execute, read, append and write.
- M: access control matrix $M : S \times O \rightarrow \wp(A)$.
- b: current access right $b = (s_i,o_i,x) \in S \times O \times A$;
- S: a set whose elements are system states, which are determined by the states of all its subjects and objects, and its element is denoted as s, state $s=(b,f,M,H)$ with $f=(f_O,f_S,f_C)$ and H is a hierarchically structured set of objects;

Based on the above concrete model, we may express security properties in the specified security model. For example, the "no read up" property in the BLP model can be expressed as: $\forall s \in S, o \in O : ((s,o,r) \in b \vee (s,o,w) \in b) \Rightarrow f_S(s) \geq f_O(o)$.

With the system model and the specified security property, the runtime security policy can be implemented accordingly. Meanwhile, according to security property requirements, i.e., the specified security property, a verifier can check whether the security policy satisfies the specified expectation.

3.3. Security model oriented attestation

At runtime, a security policy instance is enforced to control the behaviors of CBS. If a security policy is in compliance with a security model, the security policy can be viewed as an instance of the security model. In order to attest the security model of the specified CBS, the challenger needs to check whether the enforced security policy satisfies the security model and whether the security policy is correctly enforced.

Compliance between security policy and security model: The policy change behavior can be denoted as: $Capt_{i+1}=f_{cdo}(s,Capt_i,u,cc)$. $Capt_{i+1}$ and $Capt_i$ are security policy instances and they are supposed to be in compliance with the expected security model. A security model depicts some properties on the capability sets. Thus, the compliance checking between the security model and the policy is to check whether the capability set holds these properties.

For an enforced security policy instance, we may transfer these access control rules into specified models, such as the state machine model we just introduced. Then it is possible to employ some automated tools to check whether the security policy is in compliance with the specified security model or whether the security policy has the specified security property. In the past years, many techniques for property verification on policies have been proposed (Kolovski et al., 2007; Zhang et al., 2005; Fisler et al., 2005). With the broad adoption of eXtensible Access Control Markup Language (XACML), it becomes practical to transfer these security policy instances into XACML, and then carry out a formal

Please cite this article as: Gu L, et al. Security model oriented attestation on dynamically reconfigurable component-based systems. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.014

verification on the XACML based policy according to the specified security model.

Trusted policy change behaviors: In a CBS, the components and reflective components can be updated dynamically, and as a result, the security policy can be dynamically reconfigured at runtime. So the enforced policy can have many different versions during the execution of CBS. All these different versions of security policy are required to be recorded, in order to attest the policy change behavior which transfers the system state from $Capt_i$ to $Capt_{i+1}$.

Trusted enforcement: The policy enforcement mechanisms should be attested to make sure that these recorded policies are correctly enforced at runtime. The *AS* monitors these security mechanisms and records their states immediately before they execute.

Trusted behaviors for system state change: A system state change behavior is denoted as $s_{i+1}=f_{do}(s_i,Capt,u,sc)$, where the s_i and s_{i+1} are the system states before and after the behavior. In order to attest that the behavior is correctly executed, we need to attest that state s_i and the commands *sc* are trusted. Meanwhile, the identification of *u* should also be trusted. The trustworthiness of *Capt* can be verified by the compliance checking.

3.4. Identifying attestation objects

A straight way for attesting the whole CBS is to monitor all executed objects. However, as the constraints of a specific security model, as well as different granularity requirements on the target CBS, it is not necessary to monitor all objects in the system. In our scheme, it involves a preparation phase to identify attestation objects for runtime attestation. In the preparation phase, all attestation objects involved can be identified by analyzing the attestation requirements, including the given security model and expected security policies, as well as granularity and scaling requirements. The task of identifying attestation objects can be carried out by different parties, such as challengers, system administrators and program developers. We assume that the preparation phase is carried out in a trusted domain or by a trusted party.

In different applications, challengers may want to attest the CBS at different granularity levels. As shown in Section 2.1, an administrator has to consider the security of the whole system, while an end user may only be interested in parts of the CBS, like an individual component or a subsystem. Accordingly, the attestation on a CBS can be handled in two different levels of granularity: the whole CBS level or an individual component level. In order to attest a subsystem in CBS, it is only necessary to attest all individual components in the subsystem and their dependent objects in the system. Thus our scheme considers the solution for identifying attestation objects in two aspects: What objects should be monitored to attest the execution of a whole CBS? What objects should be monitored to attest the execution of an individual component or a subsystem in a CBS?

3.4.1. Identifying attestation objects according to security model

When the behavior of a whole CBS is concerned, the attestation objects should be identified according to the given security model. With the guarantee that the security policy is correctly enforced, it is not necessary to monitor and record all subjects in the CBS. For example, in the BLP model, the "no read up" property guarantees that lower level entities can not read information from the subjects in the higher level. The correct enforcement of BLP can guarantee this property. So when the challenger needs to attest that the information in the higher level components is not leaked out to the lower level, the *AS* only has to monitor the security enforcement mechanism and states of higher level components, without recording the states and behaviors of the lower level components. So the candidate components to be monitored and recorded can be reduced according to a specific security property. As a result, *AS* does not have to attest all components in the system.

3.4.2. Identifying attestation objects according to granularity

When only parts of the CBS is concerned, such as an individual component or a subsystem, the attestation on the CBS can be carried out in a fine-grained and flexible approach: the target software component can be attested with its dependent objects in the system, according to the given security model. When only parts of the CBS are involved, we will first identify the attestation objects according to granularity and scale, and then reduce the attestation object list according the security model. We will first introduce how to identify attestation objects when only an individual component is concerned. Then we will discuss how to identify attestation objects for attesting a subsystem in the CBS.

Identifying attestation objects for attesting on an individual component: Normally, system functionalities are not solely encapsulated within one component. Therefore, the execution of one component can be affected by the changes on another component. In addition, replacing a new version of a specific component might involve replacing the components on which it depends. Informally, a dependency describes the relevance between two objects such as instructions, procedures, processes in one platform or even across two platforms. Traditional program dependency analysis is usually concerned with the code level (Horwitz and Reps, 1992). In this paper, we are particularly interested in the dependencies at the component level, which are the relations among software components within a system architecture.

The system dependency graph (SDG) (Horwitz and Reps, 1992) represents the data dependencies and the control dependencies among procedures of a program. These inter-procedural dependencies include: (1) parameter-out/write: the data flows from the first party to the second one; (2) parameter-in/read: the data flows from the second party to the first; (3) non-parameter method/function call: the first party calls the execution of the second one, without data exchange. Data dependency in an SDG refers to a program's read or write access to data objects such as configuration files. These dependencies can also be used to depict the information flows among components and other objects in a CBS. We may identify the directly and indirectly dependent objects of the target component with a solution similar to Gu et al. (2008a).

With these dependent objects and their dependencies relationships, we may once again reduce the dependent object list to contain only these objects whose executions are restricted according to the expected security model of the CBS.

Identifying attestation objects for attesting a subsystem in CBS: When an subsystem in CBS is concerned, we may first identify the dependent objects of all individual components in the subsystem and then join these dependent objects together as the attestation object list. Then we will reduce the attestation object list to contain only these restricted objects according to the expected security model of the CBS.

3.4.3. Updating attestation object list

As dynamically reconfigurable CBS can evolve because of component updates and security policy modifications, the attestation object list should be updated according to the dynamic system changes. When the security policies are modified at runtime, the set of restricted objects may be changed and the

Please cite this article as: Gu L, et al. Security model oriented attestation on dynamically reconfigurable component-based systems. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.014

attestation objects should be updated accordingly. For newly added objects in the security policy, they are added into the *AOL*; for objects removed from the security policy, they should also be removed from the *AOL*. The component management in CBS may also cause security changes. When a component is installed or uninstalled in the CBS, the target component and its dependent objects should be added into or removed from the *AOL* accordingly.

3.5. Attestation procedure

The attestation procedure has two phases: measurement and verification.

The measurement phase monitors and records the execution of CBS according to the attestation object list. *AS* employs TPM to record the states of related objects in *AOL*. The configuration of the runtime environment layer and platform layer should be recorded in order to attest its initialization integrity. Then the state of security policy should be recorded before and after each policy change event in CBS. As discussed in the previous sections, *AS* needs to monitor and record the following activities related with objects in *AOL*: the life cycle management of components in the CBS, the enforcement of security policy and the restricted behaviors of specified components. At the end of the measurement phase, TPM generates a signature on these records with *TPM_Quote*. Then the attestation service returns the policy files and records to the challenger.

In the verification phase, the challenger verifies the runtime measurements to check whether the CBS behaves as expected in following steps:

1. The challenger checks the integrity of these records.
2. The challenger verifies the validation of AIK to attest the TPM.
3. The challenger verifies the measurements according to the TPM Quote.
4. Finally, the challenger checks whether the policy instances are in compliance with specified security model and returns the attestation result.

If any of the above steps fail, the verification procedure will terminate with a failure result.

4. Case study: attestation on OSGi systems

Facing the increasing requirements for corporative environments, such as security, scalability, reliability, etc., the open service gateway initiative (OSGi) alliance (The OSGi Alliance, 2005) introduced the OSGi specification to facilitate the assembly of systems with existing components and services. As a typical CBS, the components in OSGi platform are implemented as bundles, which can be dynamically and remotely deployed, executed and maintained in open networks. With the widely adoption of OSGi applications, like home service, there is an urgent requirement for more dependable and trustworthy systems. However, the services in OSGi systems may come from different vendors, and the open environment makes it difficult to evaluate the trustworthiness of OSGi systems.

The security of OSGi is based on the Java security architecture and it involves three levels (OSGi, 2010): the Java virtual machine, the security features of Java language and the OSGi framework. Policy management is introduced as an important mechanism to support the execution of these three security levels. However, the management of security policy is delegated to an assumed trusted operator and it does not work for parties in different trust domains. Without a root of trust, these security mechanisms are not adequate to support trust establishment between clients and OSGi systems in an open environment.

As a demonstration, we applied our scheme to support trust establishment for OSGi systems. The proposed attestation architecture for OSGi systems is shown in Fig. 2. The host platform of the target OSGi system should be equipped with a TPM which serves as the root of trust. As a case study, we use Felix (Apache Felix, 2010) to provide the OSGi framework.

An attestation service is introduced to monitor these security enforcement mechanisms. We implemented the attestation service based on the Java virtual machine tool interface (JVMTI, 2010). The attestation service employs TPM to record the states and events of concerned targets at two levels: the JVM level, including Java security manager and class loader; and the OSGi framework level, including bundles and configuration files. The attestation policy is designed to support fine-grained and flexible attestation.

In order to attest the security model of the target OSGi system, the security policy enforcement mechanisms in Java virtual machine and OSGi framework should be monitored and recorded for attestation. At the Java virtual machine level, the security manager enforces the security policy and the class loaders are responsible for loading and initializing these classes. At the OSGi framework level, the life cycle management mechanism of bundles and services cooperates with the permission mechanisms to manage the coordination and communication among them. Three types of permissions management are concerned: admin permission, service permission and package permission. Admin permission restricts the access to administrative functions of the framework; service permission controls the access of services in the system; package permission limits exports and imports to trusted bundles. These security mechanisms cooperate to enforce the given security policy which conveys a specific security model, and they are monitored for attestation. The activities for permission changes are recorded with the permission configuration before and after the action. For bundles, their security policies may exist in the form of configuration files, and these configuration files are also monitored.

When only parts of the OSGi are concerned, such as a service, its dependent services set can be obtained via dependency analysis. The service dependency management can help identify the dependent services via analyzing the manifest of bundles, such as the subprojects of Felix: dependency manager (Apache Felix, 2010). With the dependent services set, the attestation policy for attestation service is configured accordingly. For a given security model, these concerned services are monitored, and the

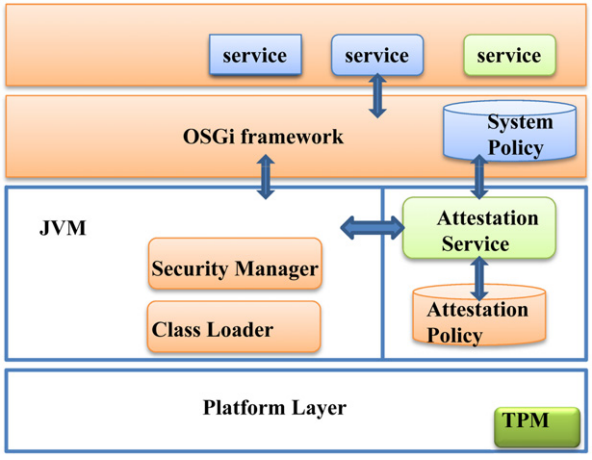

Fig. 2. Attestation architecture for OSGi systems.

Please cite this article as: Gu L, et al. Security model oriented attestation on dynamically reconfigurable component-based systems. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.014

attestation policy will provide the specification for them. The attestation service provides an interface to manage the attestation policy. With the attestation object list corresponding to the given security model, the attestation service can set the attestation policy accordingly.

Security evaluation: The security of the platform layer can be attested by its authenticated boot records. The security of the attestation service can be attested by checking the code integrity of Java Virtual Machine and attestation service module. With TPM, the monitoring process and measurements can be attested. After the initial stage, the sandbox mechanism in Java Virtual Machine protects the execution of Java program, so the trust chain can be built from TPM to the execution of bundles.

As TPM provides a strong support to record the states and events in OSGi systems, the shared object attacks (McGraw and Felten, 1999) can be identified. The root of trust for trust establishment between clients and OSGi system is based on the secure chip, TPM. Thus our scheme works even the platform operator is untrusted. According to the threat model in Section 3, the runtime records for the objects in the trust chain can be verified accordingly to attest whether the trust chain is built, and finally attest the enforced security model of Felix.

Performance evaluation: We implemented the prototype of our attestation service based on JVMTI (2010), Felix and TrouSerS-0.3.1. We evaluated the performance of our attestation service by monitoring the execution of Felix. Our experiment mostly concerns the following activities during the OSGi execution: the framework initialization and bundle life cycle management. These activities may result in security configuration changes on the OSGi System and they greatly affect the security state of the CBS. We studied the performance of *AS* on monitoring these two kinds of activities. The experiment was carried out on a Lenovo ThinkCenter M8000t desktop with Intel Core 2 Quad E8400 @ 3 GHz and 2G Memory. The host system is Ubuntu with kernel 2.6.28.14.

The Felix initialization is referred to the process for Felix to boot into its control console, which is comprised of resolving the system bundle, reloading any cached bundles, and activating the system bundle. The initialization process involves the whole CBS and it boots the system into a specific state. Thus the initialization process is a key activity for the whole CBS, and it should be monitored and recorded for attestation. The initialization is implemented as part of method *main* in class *org.apache.felix.main*. During the initialization, all installed bundles and related configurations are loaded into the system, causing an overhead on monitoring the process. As shown in Table 1, we carried out two groups of experiments on framework initialization: with and without the attestation service. For each group, the number of installed bundles in the system varies from 10 to 25. Each number in Table 1 is the average time of the 20 runs of initialization. As it involves many loading activities, the cost for monitoring the initialization is non-negligible. However, as it only runs once for a typical system, it is acceptable for a long running system in order to support higher security guarantee.

Table 1
The initialization time for Felix (millisecond).

# of bundles	10	15	20	25	30
Without *AS*	220.6	232.2	265.2	300.4	346.2
With *AS*	474.8	586.8	729.6	868.1	1008.4

The bundle management activities may also result in changes on the CBS security state. We studied the cost of *AS* for monitoring bundle installation and update in Felix. The performance results are shown in Table 2. The cost is mostly caused by invoking TPM for recording the bundle's state. The costs for these activities only hold a small portion in the whole life cycle of bundles. As a result, it is acceptable for transactions with low frequency on bundle installation and update in Felix.

5. Related work

Recent studies on remote attestation and security models are already introduced in Section 2. This section will briefly introduce the related work of CBS security.

Recent studies on component security (Khan and Han, 2002; Lindqvist and Jonsson, 1998; Clarke et al., 2003; Sun et al., 2008; Yan and MacLaverty, 2006) concern mostly twofold: how to build secure components and secure composite systems from components, and how to evaluate component security properties. Security policy is widely used to support the security of CBS (Goeminne et al., 2006). Some studies (Pistoia et al., 2007; Sun et al., 2008) concern the policy validation. For the security evaluation perspective Muskens and Chaudron (2004) introduce an integrity measurement mechanism in CBS, however, the mechanism itself is fragile because of software attacks. Certifying security of software components (Ghosh and McGraw, 1998) should run in a trusted domain and it can not solve the problem of trust establishment between clients and dynamic CBS in mutual distrust domains. However, security certification can serve as a technique to support verification. Information flow control is employed in the component-based framework for securing embedded systems (Abdellatif et al., 2010). An autonomic trust management solution was introduced for component based software systems (Yan and Prehofer, 2010).

6. Conclusion

In this paper, we propose a new attestation scheme to support trust establishment between clients and a dynamic CBS in an open environment. For a whole dynamic CBS, our scheme is capable of proving whether its execution is in compliance with the specified security model. With TPM, our scheme has a strong root of trust to resist the software attacks. For an individual component in a dynamic CBS, our scheme attests the execution of the component by monitoring the execution of the component and its dependent components. Our scheme is fine-grained and

Table 2
The performance of transactions with or without the attestation service (ms).

Bundle		Without *AS*	With *AS*	Bundle		Without *AS*	With *AS*
Shell remote	Install	20	41	File install	Install	26	48
	Update	12	35		Update	4	28
Dependence manager	Install	33	57	http.api	Install	22	45
	Update	4	30		Update	4	25
Dependence manager shell	Install	29	51	http.proxy	Install	22	46
	Update	4	27		Update	4	28

Please cite this article as: Gu L, et al. Security model oriented attestation on dynamically reconfigurable component-based systems. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.014

can support flexible attestation on different parts of a dynamic CBS. We also implemented a prototype of attestation service based on JVMTI for Felix. The evaluation results show that our scheme is effective and practical.

Acknowledgements

This work is supported by the National Basic Research Program of China (973) under Grant No. 2009CB320703 and the Science Fund for Creative Research Groups of China under Grant No. 60821003.

References

Abdellatif T, Rouis N, Saidane W, Jarboui T. Enforcing the security of component-based embedded systems with information flow control. In: International conference on communication in wireless environments and ubiquitous systems: new challenges (ICWUS), 2010; 2010. p. 1–6.

Alam M, Zhang X, Nauman M, Ali T, Seifert J-P. Model-based behavioral attestation. In: SACMAT '08: Proceedings of the 13th ACM symposium on access control models and technologies. New York, NY, USA: ACM; 2008. p. 175–84.

Apache Felix, URL 〈http://felix.apache.org/site/index.html〉; 2010.

Baiardi F, Cilea D, Sgandurra D, Ceccarelli F. Measuring semantic integrity for remote attestation. In: Chen L, Mitchell CJ, Martin A, editors. Proceedings of the second international conference on trusted computing, trust 2009, Oxford, UK, April 6–8, 2009, Lecture notes in computer science, vol. 5471. Springer; 2009. p. 81–100.

Bell DE, La Padula JL. Secure computer systems: mathematical foundations. Technical Report ESD-TR-73-278, MITRE Corporation; 1973.

Biba KJ. Integrity considerations for secure computer systems. MTR-3153, Rev. 1, The Mitre Corporation; 1977.

Blaze M, Feigenbaum J, Lacy J. Decentralized trust management. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy. IEEE Computer Society Press; 1996. p. 164–73.

Boutaba R, Aib I. Policy-based management: a historical perspective. J Network Syst Manage 2007;15(4):447–80.

Chen L, Landfermann R, Löhr H, Rohe M, Sadeghi A-R, Stüble C. A protocol for property-based attestation. In: STC '06. New York, NY, USA: ACM Press; 2006. p. 7–16.

Clark DC, Wilson DR. A comparison of commercial and military security. In: Proceedings of IEEE symposium on security and privacy, Washington DC; 1987. p. 184–94.

Clarke D, Richmond M, Noble J. Saving the world from bad beans: deployment-time confinement checking. SIGPLAN Not 2003;38(11):374–87.

Denning DE. A lattice model of information flow. Commun ACM 1976;19(5): 236–43.

Fisler K, Krishnamurthi S, Meyerovich LA, Tschantz MC. Verification and change-impact analysis of access-control policies. In: 27th International conference on software engineering (ICSE 2005) 15–21 May 2005, St. Louis Missouri, USA. ACM; 2005. p. 196–205.

Garfinkel T, Pfaff B, Chow J, Rosenblum M, Boneh D. Terra: a virtual machine-based platform for trusted computing. In: Proceedings of the 19th ACM symposium on operating systems principles (SOSP'03). Bolton Landing, NY, USA: ACM SIGOPS; 2003. p. 193–206.

Ghosh AK, McGraw G. An approach for certifying security in software components. In: Proceedings of the 21st NIST-NCSC national information systems security conference; 1998. p. 42–8.

Goeminne N, Jans GD, Turck FD, Dhoedt B, Gielen F. Service policy enhancements for the OSGi service platform. In: Proceedings of the 9th international symposium on component-based software engineering, CBSE 2006, Västerås, Sweden, June 29–July 1, 2006, Lecture notes in computer science, vol. 4063. Springer; 2006.

Goguen JA, Meseguer J. Security policies and security models. In: Proceedings of the IEEE symposium on security and privacy; 1982. p. 11–20.

Gu L, Cheng Y, Ding X, Deng RH, Guo Y, Shao W. Remote attestation on function execution. In: Chen L, Yung M, editors. INTRUST, Lecture notes in computer scienceSpringer; 2009. p. 60–72.

Gu L, Ding X, Deng RH, Xie B, Mei H. Remote attestation on program execution. In: Proceedings of the 3rd ACM workshop on scalable trusted computing, STC 2008, with CCS'08, Alexandria, VA, USA, October 31, 2008. ACM; 2008. p. 11–20.

Gu L, Ding X, Deng RH, Zou Y, Xie B, Shao W, Mei H. Model-driven remote attestation: attesting remote system from behavioral aspect. In: International symposium on trusted computing (TrustCom). IEEE Computer Society; 2008. p. 2347–53.

Haldar V, Chandra D, Franz M. Semantic remote attestation—a virtual machine directed approach to trusted computing. In: The third virtual machine research and technology symposium (VM '04). USENIX; 2004. p. 29–41.

Hnetynka P, Plasil F. Dynamic reconfiguration and access to services in hierarchical component models. In: Gorton I, Heineman GT, Crnkovic I, Schmidt HW, Stafford JA, Szyperski CA, Wallnau KC, editors. Proceedings of the 9th International Symposium on component-based software engineering, CBSE 2006, Västerås, Sweden, June 29–July 1, 2006, Lecture notes in computer science, vol. 4063. Springer; 2006. p. 352–9.

Horwitz S, Reps T. The use of program dependence graphs in software engineering. In: ICSE '92 Proceedings of the 14th international conference on Software engineering. New York, NY, USA: ACM Press; 1992. p. 392–411.

Jaeger T, Sailer R, Shankar U. PRIMA: policy-reduced integrity measurement architecture. In: SACMAT '06. New York, NY, USA: ACM Press; 2006. p. 19–28.

JVMTI. Sun microsystems, Inc. JVM Tool Interface (JVMTI), URL 〈http://java.sun.com/j2se/1.5.0/docs/guide/jvmti/〉; 2010.

Khan KM, Han J. Composing security-aware software. IEEE Software 2002;19(1): 34–41.

Kil C, Sezer EC, Azab AM, Ning P, Zhang X. Remote attestation to dynamic system properties: towards providing complete system integrity evidence. In: DSN. IEEE; 2009. p. 115–24.

Kolovski V, Hendler JA, Parsia B. Analyzing web access control policies. In: Williamson CL, Zurko ME, Patel-Schneider PF, Shenoy PJ, editors. Proceedings of the 16th international conference on World Wide Web, WWW 2007Banff, Alberta, Canada: ACM; 2007. p. 677–86. May 8–12, 2007.

Li N, Mitchell JC, Winsborough WH. Design of a role-based trust-management framework. In: SP '02: Proceedings of the 2002 IEEE symposium on security and privacy. Washington, DC, USA: IEEE Computer Society; 2002. p. 114–30.

Lindqvist U, Jonsson E. A map of security risks associated with using cots. Computer 1998;31(6):60–6.

McGraw G, Felten EW. Securing Java: getting down to business with mobile code. John Wiley and Sons; 1999. pub-WILEY:adr.

Muskens J, Chaudron M. Integrity management in component based systems. EUROMICRO Conference 2004:611–9.

Nauman M, Alam M, Zhang X, Ali T. Remote attestation of attribute updates and information flows in a UCON system. In: Chen L, Mitchell CJ, Martin A, editors. Proceedings of the second international conference on trusted computing, trust 2009, Oxford, UK, April 6–8, 2009, Lecture notes in computer science, vol. 5471. Springer; 2009. p. 63–80.

OSGi About the OSGi service platform, URL 〈http://www.osgi.org/documents/collateral/TechnicalWhitePaper2005osgi-sp-overview.pdf〉; 2010.

Pistoia M, Fink SJ, Flynn RJ, Yahav E. When role models have flaws: static validation of enterprise security policies. In: ICSE. IEEE Computer Society; 2007. p. 478–88.

Poritz J, Schunter M, Van Herreweghen E, Waidner M. Property attestation—scalable and privacy-friendly security assessment of peer computers. Technical Report RZ 3548, IBM Research; May 2004.

Sadeghi A-R, Stble C. Property-based attestation for computing platforms: caring about properties, not mechanisms. New security paradigms; 2004.

Sailer R, Zhang X, Jaeger T, Doorn Lv. Design and implementation of a tcg-based integrity measurement architecture. In: Proceedings of the 13th USENIX security symposium. San Diego, CA, USA; 2004. p. 223–38.

Sandhu RS. Role-based access control. Advances in computers 1998;46:238–87.

Shi E, Perrig A, Doorn LV. Bind: a fine-grained attestation service for secure distributed systems. In: 2005 IEEE symposium on security and privacy; 2005. p. 154–68.

Sun L, Huang G, Sun Y, Song H, Mei H. An approach for generation of j2ee access control configurations from requirements specification. In: The eighth international conference on quality software; August 2008. p. 87–96.

The OSGi Alliance. OSGi service platform—core specification. Release 4. URL 〈http://osgi.org/osgi_technology/download_specs.asp〉; August 2005.

Trusted Computing Group. TPM main specification. Main Specification Version 1.2 rev. 85, Trusted Computing Group; February 2005.

Yan Z, MacLaverty R. Autonomic trust management in a component based software system. In: Yang LT, Jin H, Ma J, Ungerer T, editors. Proceedings of the third international conference on autonomic and trusted computing, ATC 2006, Wuhan, China, September 3–6, 2006, Lecture notes in computer science, vol. 4158. Springer; 2006. p. 279–92.

Yan Z, Prehofer C. Autonomic trust management for a component based software system. IEEE Trans Depend Secure Comput 2010;PP(99):1.

Zhang N, Ryan M, Guelev DP. Evaluating access control policies through model checking. In: Zhou J, Lopez J, Deng RH, Bao F, editors. Information security, Lecture notes in computer science, vol. 3650; 2005. p. 446–60.

Please cite this article as: Gu L, et al. Security model oriented attestation on dynamically reconfigurable component-based systems. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.014