

## 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),  
 Institute of Software, School of EECS, Peking University, Beijing, China.  
 {guliang05, baigd08, yaoguo, cherry, meih}@sei.pku.edu.cn*

**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. 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. 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.

**Keywords**—Component-based systems; remote attestation; trusted computing; security model; security policy;

### I. INTRODUCTION

Thanks to the achievements from both academic and industrial organizations in the past few years, Component-Based Systems (CBS) has been widely applied in various applications, including 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 whether 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 [1–3]. 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[1];
- 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 mechanisms based on cryptographic protocols [4, 5] are not adequate to establish trust between clients and the dynamic CBS;
- Furthermore, a dynamic reconfigurable CBS [6] can evolve because of component updates, runtime environment changes or users' 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)[7] 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 [8–14]. We use TCG attestation [7] 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.

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 records, 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.

This paper makes the following main contributions:

- To the best of our knowledge, this is the first work for applying the TCG-based attestation techniques specifically on dynamically reconfigurable component-based systems to attest the security model of CBS execution.
- 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.
- The implementation and evaluation of the prototype are studied in the case study and it demonstrates our scheme in practical usage.

The rest of the paper is organized as follows: Section II introduces the background, including two motivating scenarios, the TCG attestation and security model. Section III presents our solution for attesting CBS. Section IV introduces the case study and evaluation of our scheme: attestation on OSGi system. Section V introduces related work and Section VI concludes the paper.

## II. BACKGROUND

### A. 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.

### B. Remote Attestation

Existing remote attestation schemes mostly come into three categories: integrity attestation [8, 10, 15], property-based attestation[11] and semantic attestation[9, 12, 13]. 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.

### C. Security Model and Security Policy

A security model is a high level specification or an abstract machine description of what the system does [16]. 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 [17–19]. 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 [20].

### III. 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 specification 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,  $\mathcal{AS}$  records the execution of relevant parts in the CBS, as well as the security enforcement  $\mathcal{PE}$ . At verification time,  $\mathcal{CH}$  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.

#### A. Scheme Overview

1) *Scheme Architecture*: The architecture of our scheme is shown in Figure 1. Two parties are involved: the challenger  $\mathcal{CH}$  and the platform  $\mathcal{H}_r$  which hosts the CBS. The challenger can be a remote user of the CBS, a local system administrator, or even a process. The platform  $\mathcal{H}_r$  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  $\mathcal{PE}$  in its framework layer.  $\mathcal{PE}$  consists of a policy decision point, policy enforcement point and policy records. We introduce an attestation service  $\mathcal{AS}$  with corresponding attestation policy in the runtime environment layer for CBS. The  $\mathcal{AS}$  employs the TPM to dynamically monitor and record the runtime execution of CBS and the function of  $\mathcal{PE}$ . In order to support the secure domain for runtime environment layer, the platform layer may employ trusted virtual machine [15] to provide an isolated environment. The communications between  $\mathcal{CH}$  and  $\mathcal{AS}$  can be protected by cryptographic protocols.

2) *Attestation Service*: The attestation service  $\mathcal{AS}$  is introduced in the foundational layer in the runtime environment of CBS.  $\mathcal{AS}$  should be able to monitor the execution of CBS and the security policy enforcement mechanism  $\mathcal{PE}$ . According to the specific security model and other attestation requirements, all objects required to be attested are identified as an attestation objects list  $AOL$ .  $\mathcal{AS}$  is responsible for recording the states and behaviors of these attestation objects. After all required proofs for attestation are collected,  $\mathcal{AS}$  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 challengers' 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.

#### B. Security Model Oriented Attestation

At runtime, a security policy instance is enforced to restrict 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 [21–23]. 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 verification on the XACML based policy according to the specified security model.

*Trusted Policy Change Behaviors*: In a CBS, the security policy can be dynamically reconfigured at runtime, by updating components or reflective components. 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  $\mathcal{AS}$  monitors these security mechanisms and records their states immediately before they execute.

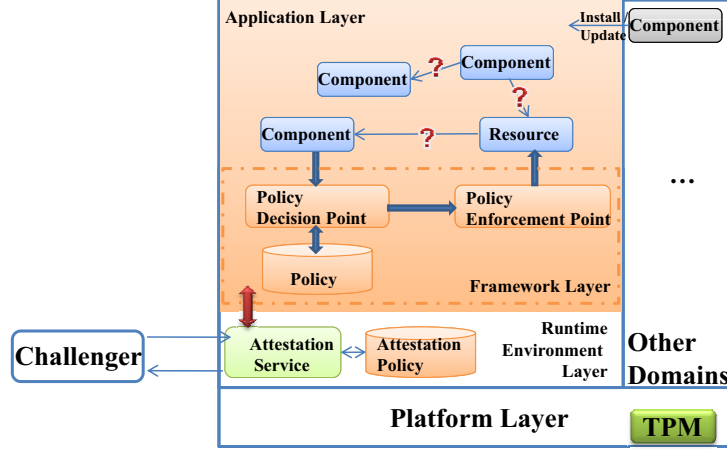


Figure 1. The architecture for attesting Component-Based Systems

#### 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.

#### C. Identify Attestation Objects

A straight way for attesting the whole CBS is to monitor all executed objects. However, as the restrictions 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 scale requirements. The task of identifying attestation objects can be carried out by different parties, such as the challengers, the system administrators and the program developers. We assume that the preparation phase is carried out in a trusted domain or by a trusted party.

1) *Identify 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  $\mathcal{AS}$  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,  $\mathcal{AS}$  does not have to attest all components in the system.

2) *Update Attestation Object List:* As dynamically re-configurable 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 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.

#### D. 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.  $\mathcal{AS}$  employs TPM to record the states of the 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,  $\mathcal{AS}$  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 run-time 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.

#### IV. CASE STUDY: ATTESTATION ON OSGI SYSTEMS

As a demonstration, we applied our scheme to support trust establishment for OSGi [24] systems. 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 [25] to provide the OSGi framework.

An attestation service is introduced to monitor these security enforcement mechanisms. We implement the attestation service based on the Java Virtual Machine Tool Interface (JVMTI)[26]. 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.

*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.

*Performance Evaluation:* We implemented the prototype of our attestation service based on (JVMTI)[26], Felix and TrouSerS-0.3.1. We evaluated the performance of our attestation service by monitoring the execution of Felix. The experiment was carried out on a Lenovo ThinkCenter M8000t desktop with Intel Core 2 Quad E8400 @ 3GHz and 2G Memory. The host system is Ubuntu with kernel linux-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. As shown in Table I,

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 I 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.

#### V. RELATED WORK

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

Recent studies on component security [1–3, 27] 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[28]. Some studies [3, 29] concern the policy validation. For the security evaluation perspective, Muskens et al. [30] introduce an integrity measurement mechanism in CBS, however, the mechanism itself is fragile because of software attacks. Certifying security of software components [31] 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.

#### VI. CONCLUSION

In this paper, we propose a new attestation scheme to support trust establishment between clients and a dynamic CBS in an open environment. 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. We implemented a prototype of attestation service based on JVMTI for Felix. The evaluation results show that our scheme is effective and practical.

#### VII. ACKNOWLEDGEMENTS

This work is supported by the National Basic Research Program of China (973) under Grant No. 2009CB320703, the Science Fund for Creative Research Groups of China under Grant No. 60821003, National Key S & T Special Projects under Grant No. 2009ZX01039-001-001 and the National High-Tech Research and Development Plan of China under Grant No. 2007AA010304 and No.2009AA01Z139, and National Natural Science Foundation of China under Grant No. 60903178.

#### REFERENCES

- [1] U. Lindqvist and E. Jonsson, "A map of security risks associated with using cots," *Computer*, vol. 31, no. 6, pp. 60–66, 1998.
- [2] K. M. Khan and J. Han, "Composing security-aware software," *IEEE Software*, vol. 19, no. 1, pp. 34–41, 2002.

Table I  
THE INITIALIZATION TIME FOR FELIX ( MILLISECOND, *ms*)

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

- [3] L. Sun, G. Huang, Y. Sun, H. Song, and H. Mei, "An approach for generation of j2ee access control configurations from requirements specification," in *The Eighth International Conference on Quality Software*, pp. 87–96, Aug. 2008.
- [4] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *In Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pp. 164–173, IEEE Computer Society Press, 1996.
- [5] N. Li, J. C. Mitchell, and W. H. Winsborough, "Design of a role-based trust-management framework," in *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, (Washington, DC, USA), p. 114, IEEE Computer Society, 2002.
- [6] P. Hnetynka and F. Plasil, "Dynamic reconfiguration and access to services in hierarchical component models," in *CBSE* (I. Gorton, G. T. Heineman, I. Crnkovic, H. W. Schmidt, J. A. Stafford, C. A. Szyperski, and K. C. Wallnau, eds.), vol. 4063 of *Lecture Notes in Computer Science*, pp. 352–359, Springer, 2006.
- [7] Trusted Computing Group, "TPM main specification," Main Specification Version 1.2 rev. 85, Trusted Computing Group, Feb. 2005.
- [8] R. Sailer, X. Zhang, T. Jaeger, and L. v. Doorn, "Design and implementation of a tcb-based integrity measurement architecture," in *Proceedings of the 13th USENIX Security Symposium*, (San Diego, CA, USA), August, 2004.
- [9] V. Haldar, D. Chandra, and M. Franz, "Semantic remote attestation—a virtual machine directed approach to trusted computing," in *the Third virtual Machine Research and Technology Symposium (VM '04). USENIX.*, 2004.
- [10] T. Jaeger, R. Sailer, and U. Shankar, "PRIMA: policy-reduced integrity measurement architecture," in *SACMAT '06*, (New York, NY, USA), pp. 19–28, ACM Press, 2006.
- [11] L. Chen, R. Landfermann, H. Löhr, M. Rohe, A.-R. Sadeghi, and C. Stübke, "A protocol for property-based attestation," in *STC '06*, (New York, NY, USA), pp. 7–16, ACM Press, 2006.
- [12] L. Gu, X. Ding, R. H. Deng, B. Xie, and H. Mei, "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*, pp. 11–20, ACM, 2008.
- [13] L. Gu, X. Ding, R. H. Deng, Y. Zou, B. Xie, W. Shao, and H. Mei, "Model-driven remote attestation: Attesting remote system from behavioral aspect," in *International Symposium on Trusted Computing*, pp. 2347–2353, IEEE Computer Society, 2008.
- [14] L. Gu, Y. Cheng, X. Ding, R. H. Deng, Y. Guo, and W. Shao, "Remote attestation on function execution," in *INTRUST* (L. Chen and M. Yung, eds.), vol. 6163 of *Lecture Notes in Computer Science*, pp. 60–72, Springer, 2009.
- [15] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra a virtual machine-based platform for trusted computing," in *SOSP 2003*, (Bolton Landing, New York, USA), October, 2003.
- [16] J. A. Goguen and J. Meseguer, "Security policies and security models," in *Proc. IEEE Symposium on Security and Privacy*, pp. 11–20, 1982.
- [17] D. E. Bell and L. J. L. Padula, "Secure computer systems: Mathematical foundations," Tech. Rep. ESD-TR-73-278, MITRE Corporation, 1973.
- [18] K. J. Biba, "Integrity considerations for secure computer systems," MTR-3153, Rev. 1, The Mitre Corporation, 1977.
- [19] D. C. Clark and D. R. Wilson, "A comparison of commercial and military security policies," in *Proc. IEEE Symp.on Security and Privacy, Washington DC*, 1987.
- [20] R. Boutaba and I. Aib, "Policy-based management: A historical perspective," *J. Network Syst. Manage*, vol. 15, no. 4, pp. 447–480, 2007.
- [21] V. Kolovski, J. A. Hendler, and B. Parsia, "Analyzing web access control policies," in *WWW* (C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, eds.), pp. 677–686, ACM, 2007.
- [22] Zhang, Ryan, and Guelev, "Evaluating access control policies through model checking," in *ICISC: International Conference on Information Security and Cryptology*, LNCS, 2005.
- [23] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz, "Verification and change-impact analysis of access-control policies," in *ICSE*, pp. 196–205, ACM, 2005.
- [24] OSGi, "About the OSGi service platform." <http://www.osgi.org/documents/collateral/TechnicalWhitePaper2005osgi-sp-overview.pdf>.
- [25] "Apache Felix." <http://felix.apache.org/site/index.html>.
- [26] "Sun Microsystems, Inc. JVM Tool Interface (JVMTI)." <http://java.sun.com/j2se/1.5.0/docs/guide/jvmti/>.
- [27] D. Clarke, M. Richmond, and J. Noble, "Saving the world from bad beans: deployment-time confinement checking," *SIGPLAN Not.*, vol. 38, no. 11, pp. 374–387, 2003.
- [28] N. Goeminne, G. D. Jans, F. D. Turck, B. Dhoedt, and F. Gielen, "Service policy enhancements for the OSGi service platform," in *CBSE*, vol. 4063 of *Lecture Notes in Computer Science*, pp. 238–253, Springer, 2006.
- [29] M. Pistoia, S. J. Fink, R. J. Flynn, and E. Yahav, "When role models have flaws: Static validation of enterprise security policies," in *ICSE*, pp. 478–488, IEEE Computer Society, 2007.
- [30] J. Muskens and M. Chaudron, "Integrity management in component based systems," *EUROMICRO Conference*, vol. 0, pp. 611–619, 2004.
- [31] A. K. Ghosh and G. McGraw, "An approach for certifying security in software components," in *Proc. 21st NIST-NCSC National Information Systems Security Conference*, pp. 42–48, 1998.