A Protocol for Preventing Insider Attacks in Untrusted IAAS Clouds

LORD JEGANNATH COLLEGE OF ENGINERRING AND TECHNOLOGY

RAMANATHICHANPUTUR.

T. Sandhiya

dhiyasana1996@gmail.com

S.Siva Sankarai

sivasankarisiva30@gmail.com

Abstract—Recent technical advances in utility computing have allowed small and medium sized businesses to move their application store the cloud, to benefit from features such as auto-scaling and pay-as-you-go facilities .Before clouds are widely adopted, there is a need to address privacy concerns of customer data out sourced to these platforms. In this paper, we present a practical approach for protect in the confidentiality and integrity of client data and computation from insider attacks such as cloud clients as well as from the Infrastructure-as-a-Service(IaaS) based cloud system administrator himself .We demonstrate a scenario of how the origin integrity and authenticity of health-care multimedia content processed on the cloud can be verify educing digital watermarking in an isolated environment without revealing the watermark details to the cloud administrator. Finally to verify that our protocol does not compromise confidentiality and integrity of the client data and computation or degrade performance. Formal verification using ProVerif tool hows that cryptographic operations and protocol communication cannot be compromise during are alistic attacker model. Performance analysis of our implementation demonstrates that it adds negligible overhead.

IndexTerms—CloudComputing,TrustedComputing,Protocol,LateLaunch,DigitalWatermarking

1 INTRODUCTION

CLOUD Computing is an exciting and promising new paradigm that allows clients to outsource storage and computational resources on demand. While cloud computing bases on current technologies such as virtualization and service oriented architecture, the major driving factors of this technology are the advancement in machine architecture, the requirement to process and/or maintain large data sets and high bandwidth network channels. Additionally, features such as multi-tenancy, auto-scaling and low cost enables cloud computing to flourish more successfully than its predecessor- the Grid.

Companies are adopting cloud based IT solutions as public clouds become the source of a rich and novel range of IT solutions ranging from massive online collaborative content storage to health-care work- flow management systems. At the converse, the wide adoption of cloud based services is badly suffering due to confidentiality and security concerns especially from insider attacks [1]. One way to ensure confi- dentiality in the cloud environment is to constantly store customer data in encrypted form and decrypt it on the cloud platform on the fly when being retrieved or being operated on. However this approach is not practical due to its high computational cost [43][44] and in case of a untrusted cloud platform the confidentiality of the data can be compromised at the point the data is decrypted for computation. Researchers have proposed homomorphic encryption schemes [2], that allow computations to be carried out on encrypted content, producing an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. However so far only primitive operations are supported and there is a large amount of overhead. Moreover there is a strong requirement to make the operations of the IaaS based cloud transparent to clients. That means that clients be able to verify the underlying cloud platform

In this paper, we propose an approach to ensure the confidentiality and integrity of client data and computation on the cloud platform. This is to ensure that private data is not exposed to internal parties such as the cloud administrator and other cloud clients. Our

approach makes use of remote attestation [4], and a late launch based technique, called Flicker [29], to verify the integrity of the cloud platform. This technique secures the virtual machine (VM) launch operation and further allows the launched VM to perform operations on sensitive data in full isolation. To test our approach, we have implemented a prototype by extending a popular open source cloud computing solution known as Eucalyptus [15]. The extra integrity verification processing overhead of our approach is found to be minimal. To illustrate the practicality of our proposed protocol, we have demonstrated how it can be used to verify presence of a hidden watermark in a health-care multimedia context. This is done in a manner that preserves the confidentiality of the watermark contents and the integrity of the verification process. The contribution of our work is as follows: • We propose a protocol for secure launch of a client VM on a trusted cloud node. Other than secure launch, our second proposed protocol enables a client to protect the confidentiality and integrity of its data and computation from other client applications in the cloud and from the cloud system administrator. • In our proposed protocol architecture, the Trusted Computing Base (TCB) is reduced to the size requirement of the Flicker based code executed and its input and output. The software stack from the BIOS up to the virtual machine monitor (VMM) level is thus removed from the suggested TCB of client sensitive code executed on the cloud platform. • In a virtualized cloud environment, past system configuration cannot guarantee current or future trustworthiness of a system. We have shown how to provide assurance to clients in such an environment.

2 PROBLEM BACKGROUND

2.1 Trusted Computing

Major hardware vendors, including Intel, Dell and HP, have founded a consortium called Trusted Computing Group (TCG). The objective of this group is to build trust in computing devices such as PDAs, mobile devices and PCs and to provide a transparent view of the platform software stack to its owner. According to the TCG specifications [20], all electronic devices complying with TCG standards should be equipped with a hardware chip called Trusted Platform Module (TPM) [20]. A TPM is a secure storage area where cryptographic keys and other secure data can be stored. The key and data stored inside the TPM is protected from malicious alteration. The data stored inside the TPM normally includes platform configuration status. The platform status stored inside TPM can then be provided to external entities, through a process called Remote Attestation, to convey platform trustworthiness. The Trusted Computing Base (TCB) of a system is the collection of all hardware, firmware, and/or software modules that are vital for the security of the overall system. Any vulnerabilities occurring inside the TCB can compromise the security of the entire system.

2.2 Remote Attestation

In remote attestation, the platform (firmware and software) configuration is captured and stored in a tamper resistant and cost effective chip called a TPM. Confidential information is held inside the TPM and is then signed and reported to a remote entity for verification and attestation purposes.

2.3 Virtualization

According to Sempolinski et al. [28], there are six fundamental components of a generic cloud computing stack; (1) hardware and OS, (2) VMM, (3) VM disk image archive, (4) front-end, (5) network and (6) cloud framework. Among these, virtualization is the key enabling technology of an IaaS based cloud. Virtualization, cost effectively abstracts system resources and supports multiple and heterogeneous operating systems simultaneously on a single hardware platform

2.4 Late Launch

Late launch [20], commonly refers to technologies that allow the execution of a secure kernel or secure VM on a system after running un-trusted software. This means that the chain of trust is not started from system boot but is rather initiated dynamically at a later stage. Certain family of processors from both Intel and AMD provides an implementation of this technology. Intel named its implementation 'Trusted eXecution Technology (TXT)' [30] while AMD calls their technology 'Secure Virtual Machine (SVM)' [3]

2.5 Sealed Storage

One of the key features provided by the TPM for securing sensitive code and data is sealed storage. A TPM contains a special 2048-bit key called Storage Root Key (SRK). The private part of the SRK never leaves the TPM in plaintext. The storage key is used to seal other data and sensitive information. The seal operation takes a set of PCRs as input, and then encrypts the given data using the SRK. The seal operation outputs cipher text C along with the list of PCRs provided and its corresponding values. The corresponding unseal operation takes cipher text C and the PCRs list as input. It then compares the PCRs list against their current values. It decrypts C only if a match occurs and then decrypts the suggested data. All these operations take place inside the TPM

2.6 Flicker

Flicker [29], is an infrastructure based on late launch technology for secure execution of a small piece of security sensitive code, called Piece of Application Logic (PAL), on systems where BIOS, OS and DMA

devices are not trusted. A PAL is a piece of application logic that performs a well defined task. Flicker executes the PAL in full isolation on the system from all other software and hardware (including OS and VMM). This isolation is possible due to hardware enhancements in modern commodity platforms from both Intel and AMD (AMD's secure virtual machine Technology and Intel's TXT). On Intel platforms, invoking a Flicker session suspends the current execution environment (OS and VMM) and then executes the SENTER instruction for setting up the secure environment for PAL execution. At the end of Flicker session, the previous execution environment is resumed.

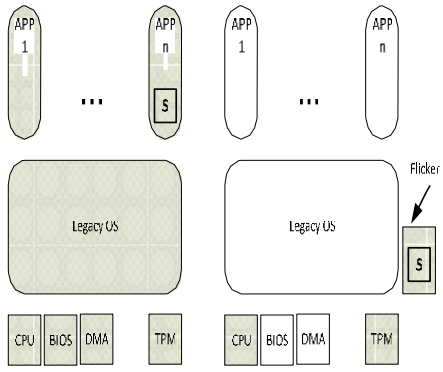

Fig.1.TCB of applications running with and without Flicker protection[29]

2.7 Protocol Verification

We will later define security protocols for secure VM launch and secure computation. As the adversary is actively looking for weaknesses, correctness of security protocols is very important. However, significant flaws have been found in widely used protocols, often years after the protocol has been defined. Examples are the Needham-Schroeder public-key protocol [33] which was found to have a serious flaw 17 years later [34], the SAML-based Single Sign-On for Google Apps [35], or the still re-occurring flaws in widely used protocols such as openssl [36] and openssh [37]. Formal models have been developed for reasoning precisely about security protocols in order to detect such flaws. Models based on process calculi (e.g. applied pi-calculus [38]) model the participants in a security protocol as processes and the messages as communication between processes.

3 DESIGN

In the first part of this section, we will introduce the two definitions "Level I security" and "Level II security" that are frequently used in the rest of the paper and are very significant to our protocol design

3.1 Level I vs Level II security

Trusted Computing and remote attestation enable a remote party to challenge a given platform and verify its security properties remotely. If the current state of a system is successfully verified, the remote party can trust this system for future operations. Here we introduce two definitions related to Trusted Computing.

Definition 1 Level I security: Platform Integrity Attestation, where before transferring computation and data, a remote party verifies through remote attestation, that the target platform belongs to the actual cloud hosting provider as well as executes trustworthy hardware, firmware and software. The client can then trust the challenged platform after verifying its current state through remote attestation for future operation.

Definition 2 Level II security: Integrity and confidentiality, where a remote party not only verifies the integrity of the target platforms hosting provider, hardware, firmware and software but also requires additional security measures to ascertain that the confidentiality and integrity of sensitive operations executed on the target platform will not be compromised. Level II security assurance can be provided with Intel TXT technology based mechanism called Flicker and will be detailed in the coming sections.,

3.2 Secure VM Launch

The client VM is stored in an encrypted form on the CS, so that it can only be launched on trusted NCs. The purpose of the secure VM launch protocol is to get the VM decryption key DkV M securely from the client, decrypt the VM and then launch it on a trusted node. The protocol proceeds in two phases. In the first phase, we certify the public keys of the client (pkc) and Flicker (pkf). This is performed by using the TPMs of the client and the NC to establish a secure channel between the client and the Flicker sessionusing the secure channel protocol of Flicker [29]. In the first step, the client sends a request to the NC for its VM launch. When the NC receives this request, it initiates a Flicker session and executes PAL P and extends PCR 18 with the measurement of PAL P and with its input and output (Flicker session con- figurations). P is an application code used to generate Flicker asymmetric keys and support the VM decryption process. Here we represent the private and public portion of the Flicker asymmetric key by Pkf and pkf respectively. The private part of the asymmetric key Pkf generated inside Flicker is then sealed for the subsequent invocation of the same Flicker session. The purpose of this asymmetric key pair is to get the VM decryption key DkV M securely from the client. In the next step, NC then sends the Flicker public key pkf and the TPM Quote of the system to the client. When NC generates a Quote, it includes the value of PCR 18 in the Quote operation and hence Quote reflects the Flicker session configurations signed with the TPM

private key. Attestation from the Flicker session can convince the client that the PAL P executed inside Flicker protections and that the public key pkf was a valid output of the session. After verifying the TPM Quote the client then establishes a secure channel to the Flicker session.

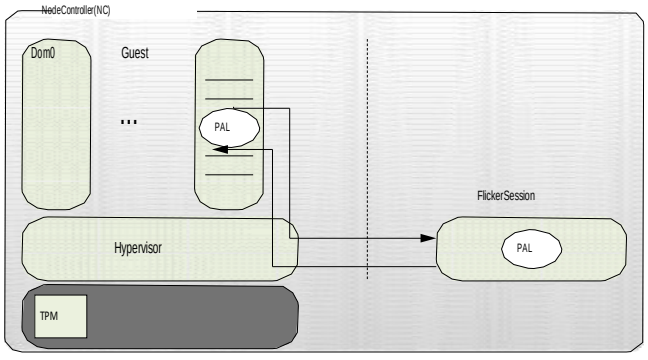

Fig.2.Generalized architecture of a cloud environment

3.3 Confidentiality Sensitive Computation

After secure VM launch through the protocol proposed in section 4.2, the next step is to ensure the confidentiality of client sensitive data and computation. Here we present a protocol for confidentiality sensitive computation based on Level II security. The computation inside the security sensitive client VM is divided into two categories, normal computation and security sensitive computation. The normal computation takes place as usual on the virtualized platform with the system administrator running in dom0. The same procedure however, cannot be followed for security sensitive computation as an insider such as the system administrator can run arbitrary processes in dom0 and can compromise the confidentiality of client data and computation. The integrity information exchanged during VM launch cannot guarantee this protection.

4 IMPLEMENTATION

In this section, we present a brief description of our proof-of-concept implementation of the proposed protocol architecture. For the realization of remote attestation, we used the most popular and widely used approach called Integrity Measurement Architecture (IMA) [21]. IMA is based on binary attestation. When configured, IMA calculates and extends hashes of all software components loaded after the boot process into relevant PCRs. To preserve privacy of the NC, we have used the Attestation Identity Key (AIK) for signing PCRs values during the Quote generation. We have used a Trusted Java [6] based software stack known as Java Trusted Software Stack (jTSS) for communicating with the TPM through the TPM driver.

5 VERIFICATION

The suggested protocol ensures confidentiality and integrity of the clients' data. We use ProVerif [39] to automatically verify that the data is not revealed to the attacker and that the protocols proposed do not suffer from man-in-the-middle attacks. Checking these two properties is sufficient to ensure confi- dentiality because we show that only the client can access the data, and secondly no other principal can masquerade as the client. We make the following assumptions: • The attacker has access to all communication (except the communications on private channels). We assume that the cloud provider may be part of the attacker. • The attacker may modify, replay and re-arrange messages but not break cryptography. • Neither the Flicker session nor the TPM nor the client is compromised. However, the attacker can interact with the Flicker session. • The public keys of the client and Flicker can be used to provide a secure communication channel between the client and the Flicker session. For this purpose we use the TPM

6 EVALUATION AND DISCUSSION

Although we conducted our experimentation on Intel based machines but the work can easily be adapted to an AMD based architecture. On an AMD architecture, the SKINIT instruction is used to invoke a Flicker session whereas in our Intel based machine the Flicker session is started with GETSEC [SENTER]. However the security property and protection provided by a Flicker on both machines is similar. On an Intel machine, the attestation requires at least PCRs 17 and 18 to be sure that a given PAL is executed inside Flicker protection. While in case of an AMD architecture it is reduced to only PCR 17. The reason is that an AMD architecture does not use a chipset module for launching a Flicker session.

7 RELATED WORK

Our work benefits from related work in trusted computing, trusted virtual machine monitors and specialized encryption based approaches. Several approaches in the first two categories are based on reliance on a trusted hypervisor also known as trusted virtual machine monitor (TVMM) [17][18]. The suggested TVMM model prevents platform administrators and other super users from examining or modifying client VMs. TVMM has the property of being able to defend its own integrity on the deployed platform. Hypervisors are unfortunately complex software and as a result inflate the Trusted Computing Base (TCB). Due to hypervisors' complexities and large TCB there are many challenges that need to be addressed for the emergence of a practical TVMM. Approaches that rely on a Trusted Computing Base We further classify approaches that make use of a TCB into two generic categories, i.e. trusted virtual machine monitor and modified VMM. Approaches in the first category build their solution on top of an existing

TVMM assuming that it would provide the desired properties of confidentiality and integrity and a root of trust. Whereas modified VMM based approaches detail certain modifications to the design of existing hypervisor's for providing desired root of trust. Trusted virtual machine monitor based approaches Terra [17] is one of the initial and influential research works that aims to provide secure closed box execution environment and an open general purpose system side by side on a single platform. Terra is based on a TVMM to make the function of a single node communicating in a distributed environment, transparent. This approach protects client computation integrity and confidentiality by providing a closed-box execution environment to a user VM from the platform administrator interference and inspection. The TVMM provides an interface to a management VM for allowing the allocation of memory, storage and other resources to the client VM. TVMM allows remote parties to trust the client VM closed-box execution environment by providing the attestation of the environment to the remote parties. A technique for achieving client computational integrity and data confidentiality in an IaaS based cloud is presented in Khan et al. [22]. The technique is based on using remote attestation and a trusted virtual machine monitor (based on a TVMM). A trusted third party is used for the establishment of trust between cloud clients and cloud nodes. The cloud provider first attests and registers its nodes with a trusted third party which in turn verifies the cloud platform properties using remote attestation. Modified VMM based approaches Murray et al. [19], analyzed the Xen architecture and proposed architectural recommendations for the emergence of a TVMM based on the Xen hypervisor. In order to reduce the TCB and make the attestationmore meaningful, they presented a technique based on disaggregation property in a Xen based environment. The technique moves the domain building process of the administrative domain into a special domain called DomB which is removed from the TCB.

8 CONCLUSION AND FUTURE WORK

In the last few years, cloud computing has experienced very high growth rates and is showing great prospects. One of the biggest challenges to the wide adoption of cloud based services is client confidentiality and integrity concerns. In this paper, we have presented and formally verified a practical solution to address this problem. Our solution includes a protocol for secure VM launch which enables clients to verify cloud platform configuration before launching their VMs on the cloud. In addition, a protocol for performing sensitive computations in a cloud environment is presented. We have formally verified the security properties of our proposed protocols using ProVerif. Currently our implementation is for Intel based systems but it can easily be adapted to AMD. Evaluation results show that our solution is practical in terms of performance. In the future, we are planning to perform rigorous penetration testing of our protocol using an actual deployment.

ACKNOWLEDGMENTS

We wish to thank Chris Dalton from HP lab Bristol for his valuable advice and many enjoyable discussions.

REFERENCES

[1] N. Kroes. "Setting up the European Cloud Partnership". World Economic Forum, 2012.

[2] M. Naehrig, K. Lauter, and V. Vaikuntanathan. "Can homomorphic encryption be practical?". In Proceedings of the 3rd ACM workshop on Cloud computing security workshop, pp. 113-124, New York, USA, 2011.

[3] Advanced Micro Devices. AMD64 virtualization: Secure virtual machine architecture reference manual. AMD Publication no. 33047 rev. 3.01, May 2005.

[4] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. OHanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen. "Principles of remote attestation". International Journal of Information Security, Volume 10, Issue 2, pp. 63-81, 2011.

[5] X. Lei, X. Liao, T. Huang, H. Li and C. Hu. "Outsourcing Large Matrix Inversion Computation to A Public Cloud". IEEE Transactions on Cloud Computing, Volume 1, Issue 2, pp. 78-89, 2013.

[6] Trusted-Java: Jsr321: Trusted computing api for java(tm) (2009) Available at: http://jcp.org/en/jsr/detail?id=321. Accessed on 06/09/2013.

[7] P. Tysowski and M. Hasan. "Hybrid Attribute- and Re-EncryptionBased Key Management for Secure and Scalable Mobile Applications in Clouds". IEEE Transactions on Cloud Computing, Volume 1, Issue 2, pp. 172-186, 2013.

[8] S. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi and P. Samarati."Integrity for Join Queries in the Cloud". IEEE Transactions on Cloud Computing, Volume 1, Issue 2, pp. 187-200, 2013.

[9] X. Leroy. "Formal certification of a compiler back-end, or: programming a compiler with a proof assistant". In 33RD Proceedings of ACM Symposium on Principles of Programming Languages, 2006.

[10] F. Krautheim, D. Phatak, and A. Sherman. "Introducing the trusted virtual environment module: a new mechanism for rooting trust in cloud computing". In Proceedings of the 3rd international conference on Trust and trustworthy computing, 2010.

[11] A. Baldwin, C. Dalton, S. Shiu, K. Kostienko, and Q. Rajpoot. "Providing secure services for a virtual infrastructure". Operating Systems Review-ACM Special Interest Group on Operating Systems, Volume 43, Issue 1, PP. 44-51, 2009.

[12] P. Colp, M. Nanavati, J. Zhu, W. Aiello, G. Coker,

T. Deegan, P. Loscocco, and A. Warfield. “Breaking up is hard to do: security and functionality in a commodity hypervisor”. In Proceedings of the Symposium on Operating Systems Principles, PP. 189-202, 2011.

[13] Intel Corporation. “Intel Virtualization Technology for Directed I/O”. Intel Publication no. D51397-004 rev. 1.2, 2008.

[14] J. McCune, B. Parno, A. Perrig, M. Reiter, and A. Seshadri. “How low can you go? Recommendations for hardware-supported minimal TCB code execution”. In Proceedings of the ACM Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2008.

[15] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. “The Eucalyptus Open source Cloud computing System”. In Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, China