Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-13

Enhancing Software Piracy And Integrity Protection In Cloud Computing With TPM

Ahmed Bentajer, AbouElMehdi Karim, Hedabou Mustapha, EL Amrani Fatima zohra, EL Fezazi Said

***Abstract*—Software as a Service (SaaS) is a kind of *aaS distribution and deployment model in which applications are provided to customers as a service. The applications can run on the user's computing systems or the CSP's Web servers. The solution may belong to a software company, which host it among a CSP and offer it as instances to customers. As mentioned [1] software piracy is the major concern of software's publisher; lot of solution has been set up, but editors still suffer and lot of copies was made or the solution was cracked. Our proposed design demonstrate how software protection can be protected from redistribution and tampering. The aim is to secure instance access through the new specification of TPM 1.2[13] that is time stamping to have the exclusive access to an instance or result of software computation, so only one person can have the access. Besides the new design, combined to the TCCP design [12] protects side channel attacks and/or attacks from the sysadmin.**

***Keywords*—SaaS, IaaS, TPM, Trust Computing**

I. Introduction

Cloud computing provides the capability to use computing and storage resources on a metered basis and reduce the investments in an organization's computing IT. Computing is a combined of many existing technologies (virtualization, storage….) and the adoption is growing greatly because of its evolving architecture, and Infrastructure as a Service is a kind of service provided through the cloud.

Over recent years, lot of effort has been done to secure software solution through serials or securing access to code [1, 2] to avoid tampering and fraud, or even in the case of tampering this would be limited to some individuals cases. Thus, the software protection is one of the most important issues concerning computer science. In the era of the cloud, software protection become easier to manage because, one, CSP cannot afford to sell illegal instances, two, software protection become easier to manage in cloud through TPM, because this wind of change has allowed a licenses software acquisition with a lower price as banking instance or health applications services.

For all this applications and others, the only person who must have access to them are legitimate and untampered client. Hence, software publisher wants to be able to verify that its solution run on a trusted platform and used by a trusted client.

A verification entity (ETC) or design is able to assure execution of software, using attestations.

Before going any further, we distinguish between two problems: Protection against software redistribution and protection against software illegitimate duplication. Our proposed design protect software's from redistribution, meaning that the software cannot be moved and installed in another platform and then guard the authenticity of the application; and blinding read access to code running on RAM from be read and protect the client from running software on an untrusted platforms.

In this paper, we address the problem of laaS threats, and how a SaaS can be placed on a set of trusted node and then protect the SaaS from redistribution. Since recent, many enabled TPM computers are sold. Therefore, we are going to take advantage from this solution without the use of heavily solution based on software and/or hardware.

II. *aaS Deployment Model

In this section we focus only on IaaS and SaaS delivery and deployment model, since the trusted Node (N) reside in the IaaS perimeter and the deployed solution is delivered as a SaaS. Because we believe that understanding, cloud computing security risks cannot be achieved without understanding the relationship and dependencies between cloud computing models

A. *IaaS Architecture and risk issues*

Infrastructure as a Service (IaaS) as defined by CSA [3] is delivering computer virtualized infrastructure as a service along with raw storage and networking, rather than purchasing servers, software, data-center space or network equipment.

In addition, to the risks and threats inherent in traditional IT computing, IaaS presents an organization with its own set of security issues. In this section, we will try to highlight the most significant risks to the IaaS architecture.

Side-channel attacks: Traditional attacks on cryptographic algorithms use only the input and output of the algorithm,

Bentajer Ahmed, EL Fezazi Said, EL Amrani Fatima zohra
High School of Technology Safi / Cadi Ayyad University
Morocco

Hedabou Mustapha
National School of Applied Science / Cadi Ayyad University
Morocco

AbouElMehdi Karim
LAMAPI, Chouaib doukali University Eljadida Morocco

SEEK DIGITAL LIBRARY

Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-13

treating it like a monolithic black box. However, this does not reflect reality. Algorithms must be implemented in software and run on hardware, which have various properties [4, 5 and 6](a physical quantity such as time, power consumption, electromagnetic radiation or sound...) that change as a result of the cryptographic algorithm's execution. Side-channel attacks try to extract secret information based on some side channel. In addition, an attacker may use multitenancy in cloud computing to gain access to data through side channel attacks by placing its VM in the same physical machine as another client [8].

Malicious Insiders [4, 7]**:** The threat of a malicious insider is well known to most organizations. This threat is amplified for consumers of cloud services by the convergence of IT services and customers under a single management domain, combined with a general lack of transparency into provider process and procedure.

B. *SaaS architecture and risk issues*

Software as a Service (SaaS) [3, 9] is solutions deliver software applications over the Web. A SaaS provider deploys software to the user on demand, commonly through a licensing model. The provider may host the application on its own server infrastructure or use another vendor's hardware. The application may be licensed directly to an organization, a user or group of users, or through a third party that manages multiple licenses.

Based on the definition of SaaS, security of a SaaS solution include security up to the software layer including hypervisor (IaaS and PaaS) security of data and the applications.

SaaS require more attention from all stakeholders, because the delivery method that provides access to an application and its functions remotely as a web-based service, which is splited into a client side (browser) and server side. Conceptually this architecture needs that information (web flows) go through network, so they can be exposed to side channel attack despite using an HTTPS protection or if they are encrypted [10]. Besides the account or service hijacking [3] such as phishing, fraud or exploitation of software vulnerabilities on which an attacker may have an admin access to the SaaS.

For more risk issues we refer the reader to [1, 3, 5, 8 and 9].

C. *Analyse of *aaS risk issues*

By analyzing this risk issues we notice that despite of implemented solution to protect software from illegal redistribution in traditional IT infrastructure, SaaS makes matter worse, even if the client take advantages of SaaS, but software publisher is not. Moreover, its solution still vulnerable to illegal redistribution because of side channel attacks [4, 5, 6]. Since any software (no matter how encrypted) is just a binary sequence which a pirate can copy (bit by bit) and run on his own machine. Hence, to protect against duplication, some hardware measures must be used.

D. *Software Protection*

Since our work is based on SaaS, and before the era of cloud, there was much research that have been done to secure the software against redistribution and protect publisher intellectual solution.

Software piracy occurs when people copy, sell, share, or distribute software illegally. It can vary from a limited case of installation of a single-user license on multiple computers to a more chronic problem of widespread online distribution. Regardless of the rationale or delivery method.

According to the BSA (Business Software Alliance) and IDC 6th Annual Global Software Piracy Study[21], the retail value of unlicensed software — representing revenue "losses" to software companies — broke the $50 billion level for the first time in 2008. Worldwide losses grew by 11 percent to $53 billion. Excluding the effect of exchange rates, losses grew by 5 percent to $50.2 billion.

In [18, 19] many solutions has been proposed to protect software redistribution, those solutions are software-based encryption to protect the access to code running in RAM. In [1] they show that software based encryption solution is not enough to protect the software redistribution; but the problem have to be theoretically studied before the suggestion of any solutions, because software based solution cannot grant total protection. Although researchers and engineer should focus on Software and hardware based solution as SH-Package [1].

Likewise, organization moved to management solutions and Framework [15, 22, 23] to protect software against redistribution and they proposed frameworks and method to enhance software protection, which include risk assessments, control activities, monitoring….

III. Trusting Computing Groupe

The Trusted Computing Group proposed a set software technologies to enable the construction of trusted computing platforms (TCP). The TCG proposed a number of technologies and standards to provide security for systems, network ….

These technologies include Trusted Platform presents the component of TPM (figure 1 present the component of TPM)(TPM) that starts used to address some security issues in cloud computing as establishing trust in the provider of IT services and enabling transparency to the physical location of data in the cloud.

The TPM is a computer chip authenticate a platform (passwords, encryption keys …). The TPM includes capabilities such as machine authentication, hardware encryption, secure key storage, and attestation.

Encryption and signing are well known makes them stronger by storing keys in protected hardware storage space. A TPM can be used to store platform measurements that help ensure that trustworthy and prove that it is what it claims to be.

SEEK DIGITAL LIBRARY

Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-13

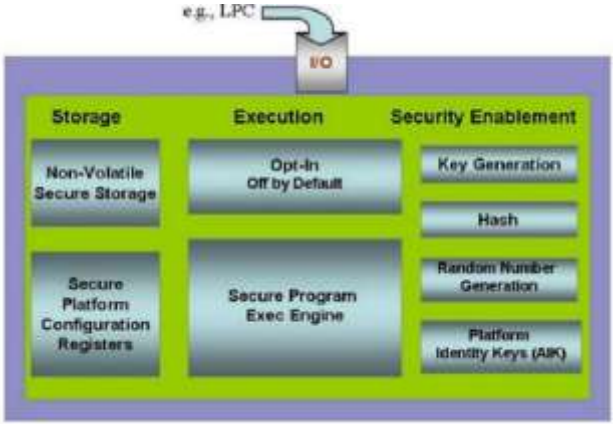

Figure 1 : Components of TPM

The TGC Software Stack (TSS) architecture's is neither platform nor OS dependent, the interaction and relationship between all modules will be the same regardless of the OS or platform (figure 2).

A. *Time Stamping*

The TGC 1.2 main specification define new TPM with new capabilities. Among these new features, there is Secure Timing, which use a tick counter with an external time stamping. The Time Stamp (TS) does not include an actual universal time clock value it is up to the caller to associate it with the ticks ta an actual UTC time, because of the price, but rather the number of timer ticks the TPM has counted since start-up of the platform.

The Tick Session Nonce (TSN) define the counted ticks from the start of timing session. At the start of a tick session, the tick session is reset to 0 and TSN is randomly generated by the TPM.

Command definition [13]:

TSS_RESULT Tcsip_TickStampBlob

(

TCS_CONTEXT_HANDLE hContext, // in

TSS_HKEY hKey, // in

TPM_NONCE antiReplay, //

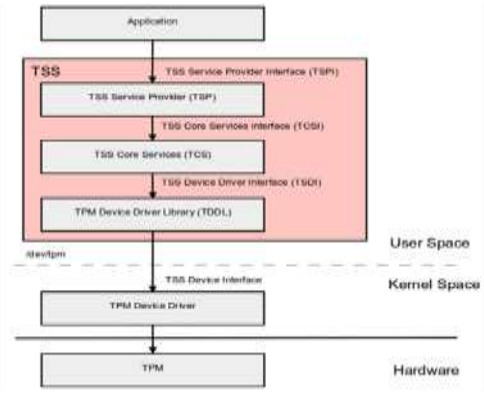

Figure 2 : Simplified architecture of TGC Software Stack[14]

TPM_DIGEST digestToStamp, // in

TPM_AUTH* privAuth, // in, out

UINT32* pulSignatureLength, // out

BYTE** prgbSignature, // out

UINT32* pulTickCountLength, // out

BYTE** prgbTickCount // out

);

Parameters

hContext The handle of 20 byte hash of blob to be tickstamped

hKey The key used to perform the signature operation.

antiReplay An application-supplied nonce to ensure freshness of the signature.

digestToStamp The value being signed

privAuth The authorization digests that authorizes the use of hKey.

pulSignatureLength Length of resultant signed tickstamp

prgbSignature On successful completion this parameter points to the signature data which makes up the tickstamp.

pulTickCountLength Length of the resulting tick count prgbTickCount.

B. *Integrity measurement*

The initial platform state is measured by computing cryptographic hashes of all software components loaded during the boot process. The task of the CRTM is to measure the code and parameters of the BIOS and extend the first PCR register with this measurement. Next, the BIOS will measure the binary image of the bootloader before transferring control to the bootloader, which in its turn measures the operating system. In this way, a chain of trust is established from the CRTM to the operating system and potentially even to individual applications. Changes in the executing code can be detected by comparing measurement of executing code against recorded value. The measurements themselves must be protected from undetected manipulation. (Figure 3: Integrity measurement by TPM).

TCG attestation is designed to provide remote verification of the complete platform configuration since start-up of the platform.

SEEK DIGITAL LIBRARY

Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-13

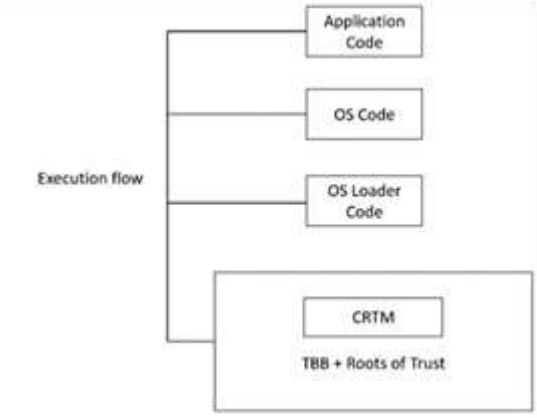

Figure 3: Integrity measurement by TPM

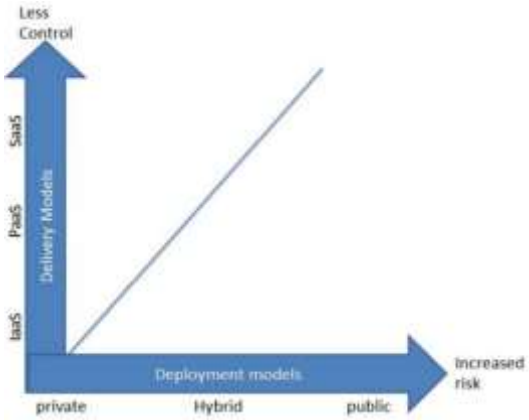

Figure 4 : Risk Relationship With Deployments Model

IV. Remote Attestation and Protection of Software Solution

It has been suggested [1], to protect software against redistribution, to use a physically shielded central processing Unit with an encrypted program SH-package (Software Hardware package). However, the solution is a little heavy to implement, because keeping the content of the RAM encrypted need more computation.

A. *Secure Deployment of a VMi at a CSP*

A client does not have any mean to verify the confidentiality and integrity of their sensitive data and computation. As we discussed in section 2.2 and what have outlined in [15], deploying a public SaaS or IaaS is not safe, as CSPs would have us to believe (figure 4).

Before that the SP offer its solutions as a service, he have to configure it on its own VM, we insist on the fact that he have to configure the solution on its own VM because: 1) it will help us to calculate the execution time of an instance which will be used to determine if there is a third party who try to inject malicious code or tamper the application. 2) The SP will use the TCCP design to deploy its preconfigured VMi in a secure Node then protect the solution from redistribution. Since the SP will inject its public key in the software and at every launch the software will verify through TPM that it still remain at the same Node where it have been deployed at the first time; if the public key correspond to the endorsement key (EK) of the TPM then the instance will be decrypted else the TPM won't be able to decrypt the instance (figure 3).

B. *The TCCP Design*

In [12] Krishna et al. proposed TCCP design to provide a closed box execution environment that guarantees confidential execution of guest virtual machines. Moreover, it allows users to attest to the IaaS provider and determine whether or not the service is secure before they launch their virtual machines.

The TCCP is able to guarantee that the VM is launched on a trusted Node, and the Sysadmin is unable to inspect or tamper with the initial VM state.

For more information about secure Node registration and secure VM launch we refer the reader to [12, 16 and 17].

C. *Improved TCCP Design for Software*

The TCCP design was in the first case designed for IaaS platform [11, 12], the design can be improved so the software publisher when configuring its VM, by injecting the public key of TPM and at each time launch for an instance the system will try to decrypt it using the private key of TPM, if the VMi has been moved to another place the new private key won't be able to decrypt the instance because it's encrypted by public key of another TPM (figure 3).

To protect the software from tampering and blinding access to data exchanged between the client and server we will use the time stamping functionality of TSS 1.2 specification.

For the first time that a client will ask for the software's instance (INST.)

- The client will use the TPM to create a tick on this nonce : TS1 □ SingSK(INST||t1||TSN1);
- The software publisher will use TS1 to create a fingerprint : SP□CKSUM(TS1,V);
- The calculated CKSUM get time stamped by TPM : TS2□SingSK(ETCC||T2||TSN2)
- The client can verify the software integrity :

- Check TS1=TS2 and SP correspond with the value that the software publisher has calculated to verify if TPM has been reset or if they were a hardware attack.

SEEK DIGITAL LIBRARY

Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-13

- Extract T2-T1 to check whether it corresponds with the expected execution time of the checksum function to verify if software take more time to open and detect any malicious changes

- Use the nonce T1 to have the exclusivity of the use of the instance, so no third party can have access even if in read mode to the instance.

When the instance has been time stamped, only on user will be able to use it or have the access to the computation result. Besides, even if a third party try to use the time stamping generated by the user to access the instance, the request will be rejected according to the TSS 1.2 specification [13].

v. Conclusion

In this work, we have proposed improvements to the TCCP design and we take advantage of new TSS 1.2 functionality. The aim is that the software publisher configure the SaaS in a VMi and deploy it at a CSP to take full advantage of TPM 1.2 and TCCP by calculating the time that the instance take to launch and then compare it with one of the software publisher to be sure that the application has not been tampered. Also this design will enhance software piracy and improve its integrity by preventing access to code running into RAM or results in webflow, used by a client, from be read or executed by a SysAdmin or an attacker

References

[1] Rafail Ostrovsky. Software Protection and Simulation on Oblivious RAMs. May 17, 1992.

[2] Kennell, R. and L. H. Jamieson, Establishing the Genuinity of Remote Computer Systems, in: Proceedings of the 12th USENIX Security Symposium, August 4-8, 2003, Washington, DC, USA (2003), pp. 295–308.

[3] Security guidance for critical areas of focus in cloud computing v3.0.

[4] Sebastian Schinzel. Time is NOT on Your Side: Mitigating Timing Side Channels on the Web. Friedrich-Alexander Universität Erlangen-Nürnberg Lehrstuhl für Informatik 1 IT-Sicherheitsinfrastrukturen.

[5] Paul Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Proc. CRYPTO ’96, volume 1109 of LNCS, pages 104–113. Springer, 1996.

[6] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Proc. CRYPTO ’99, volume 1666 of LNCS, pages 388–397. Springer, 1999.

[7] Top Threats to Cloud Computing V1.0 Prepared by the Cloud Security Alliance March 2010–6.

[8] Thomas Ristenpart, Eran Tromer† Hovav Shacham, Stefan Savagepan. Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds. CCS’09, November 9–13, 2009, Chicago, Illinois, USA.

[9] Ronald L. Krutz and Russel Dean Vines. Cloud security A comprehensive guide to secure cloud computing. Wiley Publishing Inc. ISBN: 978-0-470-58987-8

[10] Shuo Chen, Rui Wang, XiaoFeng Wang, Kehuan Zhang. Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow. Proceedings of the IEEE Symposium on Security and Privacy (Oakland)

[11] Ahmed Bentajer, Abou El Mehdi Karim, El Fezazi Said, Hedabou Mustapha. Protection Of Virtual Machine Deployment At A Cloud Service Provider Through TPM. International Journal of Current Research Vol. 7, Issue, 06, pp.16832-16834, June, 2015

[12] Nuno Santos Krishna P. Gummadi Rodrigo Rodrigues. Towards Trusted Cloud Computing. Proceeding HotCloud'09 Proceedings of the 2009 conference on Hot topics in cloud computing Article No. 3 USENIX Association Berkeley, CA, USA ©2009

[13] TGC Inc. TCG Software Stack (TSS) Specification version 1 .2 Level 1 Errata A Part1 : Commands and Structures March 7, 2007.

[14] http://bsssd.sourceforge.net/architecture.html

[15] Bentajer Ahmed, AbouElMehdi Karim, Dali Loubna, EL-Fezazi Said, Hedabou Mustapha, El Amrani FatimaEzzahra. An Assessing Approach Based On Fmeca Methodology To Evaluate Security Of A Third Party Cloud Provider. Journal of Theoretical and Applied Information Technology 30th April 2015 -- Vol. 74. No. 3 – 2015

[16] S. Berger, R. C´aceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn. vTPM: virtualizing the trusted platform module. In Proc. of USENIX-SS’06, Berkeley, CA, USA, 2006.

[17] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In Proc. of NSDI’05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[18] Christian S. Collberg and Clark Thomborson. Watermarking, Tamper-Proofing, and Obfuscation Tools for Software Protection. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 28, NO. 8, AUGUST 2002

[19] Georg T. Becker, Wayne Burleson, Christof Paar. Side-Channel Watermarks for Embedded Software.

[20] Singh N, Singh, S., Agarwal, S. An Efficient Approach for Software Protection in Cloud Computing. Fourth International Conference on Communication Systems and Network Technologies (CSNT), 2014.p. 550 – 554. Publisher : IEEE. DOI:10.1109/CSNT.2014.116.

[21] IDC Study : http://globalstudy.bsa.org/2008/studies/globalpiracy2008.pdf

[22] COSO Framework : http://www.coso.org/IC.htm

[23] ISO 27001 : http://www.iso.org

SEEK DIGITAL LIBRARY