Hypervisor-based Attestation of Virtual Environments

Hagen Lauer
Monash University
hagen.lauer@monash.edu

Nicolai Kuntze
nicolai.kuntze@gmail.com

***Abstract*—Several years ago, virtualization technologies and hypervisors were rediscovered and today virtualization is used in a variety of applications. Network operators have discovered the cost-effectiveness, flexibility, and scalability of virtualizing network functions (NFV). However, in light of current events and security breaches related to platform software manipulation the use of Trusted Computing technologies has become not only more popular but increasingly viewed as mandatory for adequate system protection. While Trusted Computing hardware for physical platforms is currently available and widely used, analogous support for virtualized environments and virtualized platforms is rare and not suitable for larger scale virtualization scenarios. Current remote and deep attestation protocols for virtual machines can support a limited amount of virtual machines before the inefficient use of the TPM device becomes a crucial bottle neck. We propose a scalable remote attestation scheme suitable for private cloud and NFV use cases supporting large amounts of VM attestations by efficient use of the physical TPM device.**

I. Introduction

Virtualization-based technologies and product offerings are gaining widespread acceptance. Recent trends like Network Functions Virtualization (NFV)[16][17][18] strive to maximize server resource utilization by an optimizing load distribution. Various cloud offerings allow customer to efficiently outsource computational tasks.

All virtualization scenarios levy requirements for the integrity of the virtual environments within Virtual Machines. At the moment, the user of a virtualized environment has to trust the service provider to ensure that Virtual Machines are not compromised. The service provider in turn has limited ability to verify the state of all the servers in a server farm. Attacks on VMs or servers might not be detected.

Typical security technologies like Virus scanners or isolation technologies like firewalls are in many cases not suitable due to high performance impacts, slow detection, or are just not applicable. Trusted Computing offers integrity verification based on trusted and verifiable measurement. Using a hardware-based root of trust, Trusted Computing lets a management system decide on the trustworthiness of a particular system.

This paper proposes a novel scheme for efficiently applying of Trusted Computing to virtual environments. The existing scheme of remote attestation for individual devices as done by Trusted Computing is extended for virtual environments as found in hypervisor-based systems supporting large numbers of VMs.

In Section II, Trusted Computing technology is presented. The requirements for using Trusted Computing in virtualized environments are defined in Section III-B, related work is discussed in Section IV, and prior is presented in Section V. The proposed solution for attesting in virtual environments is presented in Section VI, and Section VII describes variations of this solution. A conclusion and further research directions are provided in Section VIII.

II. Trusted Computing

Trusted Computing [1] is a set of technologies that provide hardware and software support for secure storage and software integrity protection. Trusted Computing allows the integrity of a computing system to be verified remotely. The system's integrity is determined based on trusted measurements of its software (e.g., the operating system, device drivers, etc.) that are taken before the software is instantiated as a process. Integrating Trusted Computing into virtualized computing systems enables hardware-based protection of private (sensitive) information and detection of corrupted software.

Understanding Trusted Computing requires the introduction of some underlying concepts. In Section II-A, the physical Trusted Platform Module (TPM) is introduced: the TPM provides the Root of Trust for the computing system. The virtual analogue of the physical TPM is presented in Section II-B.

A. Trusted Platform Module (TPM)

The TPM is a specification defined by the Trusted Computing Group [1][2]. It is implemented as a chip that is physically attached to a platforms motherboard and accessed by system software using a restricted set of well-defined commands[1]. Using the TPM to create measurements of system software is described in Sections II-C and II-D. The reporting process that uses these measurements is described in Section II-E. Section II-F describes a specific problem of this reporting in the context of virtual machines.

The current implementation of the TPM is a tamper-resistant hardware chip. Other major components of the TCG specification are pre-BIOS (Basic I/O System) hardware and software components called the Core Root of Trust for Measurement (CRTM) which is supposed to ensure that platform states are recorded in the TPM.

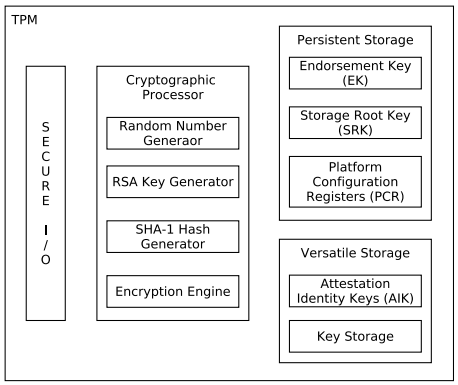

Fig. 1. Internal components of a Trusted Platform Module.

B. Virtual TPM (vTPM)

A vTPM is a system-level software module that emulates a TPM that is associated with a particular VM. In order to be trusted, a vTPM requires hardware support (e.g. from the hardware TPM) but the details of this relationship are not included in our scenario. To processes in its VM, the vTPM's visible command set and behavior is indistinguishable from that of a standard TPM (e.g. as described in II-A).

C. Trusted Boot

As anti-malware (AM) software has become better at detecting runtime malware, attackers are also getting better at creating rootkits that can avoid detection. Detecting malware that starts early in the boot cycle is a challenge that most AM vendors address diligently. Currently, AM vendors create system protection solutions that are not supported by the host operating system and can potentially place the computer in an unstable state. Trusted Boot [3][4] measures each component from firmware up through the boot start drivers, stores those measurements in the Trusted Platform Module (TPM) on the machine, and then makes a log available that can be examined remotely as part of the operation of attesting the state of the platform (Figure 2).

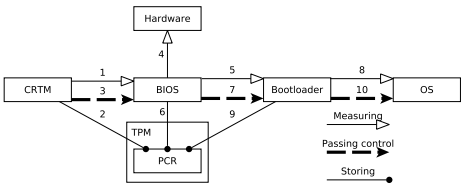

Fig. 2. Trusted Boot scheme from CRTM to OS describing measurement and control flow.

D. Integrity Measurement Architecture (IMA)

IMA [5] is an open source trusted computing component. IMA maintains a runtime measurement list Stored Measurement Log (SML) and, if anchored in a TPM using Platform Configuration Register (PCR) 10, an aggregate integrity value over this list (Template Hash or aggregate integrity value[5]). The benefit of anchoring the aggregate integrity value in the TPM is that the measurement list cannot be compromised by any software attack, without being detectable. Hence, on a Trusted Boot system, IMA can be used to extend the chain in Figure 2 from OS to Applications by performing runtime measurements.

E. Remote Attestation (RA)

Remote Attestation (RA) is the activity of making a claim about properties/state of a target by supplying evidence which supports the claim to an external appraiser.[8] In this paper this evidence is based on the contents of a TPMs PCRs which, at the point of an attestation, have been filled with the measurements performed by Trusted Boot and IMA. An appraiser is an entity, generally residing on a computer on a network, making a trust decision about some other entity. In this paper, targeted entities are Host (e.g. Virtual Machine Monitor) and Guest operating systems. Remote Attestation also involves cryptographic protocols to supply sensitive evidence data to an appraiser. Remote Attestation Protocols [6] make use of public-key encryption using a TPM's Quote[1][2] command along with its Attestation Identity Key (AIK), which is an asymmetric key pair whose public portion is known to the appraiser. The use of Remote Attestation will allow an appraiser to detect potentially malicious changes in the software configuration of a target platform.

F. Deep Attestation

Deep Attestation is a procedure for attesting entities in multiple layers of single a virtualized system.[12] A deep attestation could be necessary if an appraiser needs to assess the trustworthiness of a VM, the underlying hypervisor, and potentially attestation evidence all the way down to the underlying hardware roots of trust (e.g., physical TPM).[12] Schemes allowing for deep attestation might include quotes from both virtual and physical TPMs. As stated by the TCG,

> ... one of the largest challenges with performing a deep attestation of a virtualized system is how to know that the integrity evidence being presented is actually coming from the same physical system.

The relationship between different layers of a single virtualized system must be shown for each individual attestation of a VM.[12] It is to be noted in this context, that the binding of the layers must be based on attestable knowledge provided by the device. Knowledge stemming from 3rd parties like network management is susceptible to tampering or other attacks and therefore cannot be used to verify the layer bindings.

Deep attestation is considered the strongest form of a VM attestation and it is arguably the only meaningful way to attest a VM. It must be considered that the attestation of a VM all by itself is meaningless if its underlying hypervisor is potentially malicious. A simple example for this might be the attestation of a virtualized network function which holds certain secrets to identify itself in a network. The attestation of

this VM might lead to the conclusion that the VM is running fine and the software configuration is operating as defined and therefore it is considered fully operational. However, this conclusion can easily be spoiled by a malicious hypervisor which might eavesdrop on I/O and leak credentials or even tamper with I/O. Therefore, a hypervisor attestation should be supplied alongside a VM attestation in a way that allows an appraiser to link them. Currently, existing concepts either provide this attestation link between layers or they scale in large virtualization environments [12].

III. Use Cases and Security Requirements in Virtual Environments

One upcoming use case for virtualization is Network Functions Virtualization (NFV) as specified by ETSI [14] [16]. Currently, NFV grows to a large field of application for virtualization technology. Using this use case as an underlying example, NFV is introduced in Section III-A. As virtualized environments are different in their security requirements in contrast to physical systems, as a first step relevant attack scenarios are presented in Section III-B. The use of Trusted Computing necessitates the definition of requirements towards the Remote Attestation guiding the development and evaluation of an attestation scheme. These requirements are discussed in Section III-C.

A. Network Functions Virtualization (NFV)

ETSI's Industry Specification Work Group for NFV defines a general NFV architecture in their specification document NFV002 [15]. However, in this paper the ETSI architecture is simplified to elements relevant for the proposed attestation method (Figure 3).

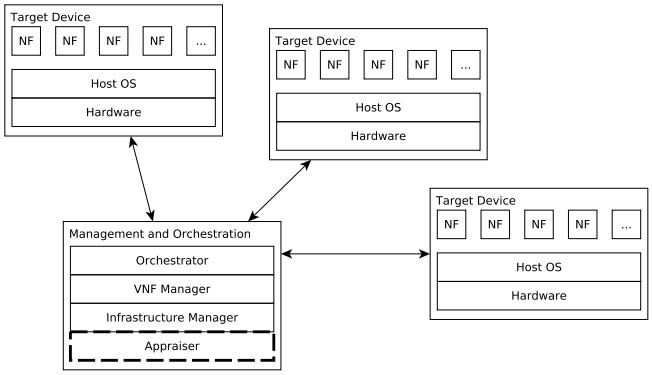

Fig. 3. Simplified NFV Architecture in accordance with [15]. The Appraiser could be part of the Management and Orchestration Node or it could be part of the Infrastructure Manager (Virtualization Infrastructure Manager). The appraiser needs means of communications to target devices, e.g. hypervisors hosting network functions in VMs.

Figure 3 presents an Appraiser as part of the Management and Orchestration node, however, if all nodes must be verifiable/attestable, the appraiser would reside in a separate node. The reference architecture could also be translated to OpenStack[1], hence, the proposed attestation method could be applied to public cloud scenarios as well.

B. Attack Scenarios

One of the most common threats for any system whether physical or virtual is the execution of malicious applications. Execution of malicious applications might be the result of an user installing malicious software unintentionally or intentional modification of installed binaries. However, these applications do not interfere with the OS or other processes. A practical example would be an application that deletes all user files. These attacks might not be critical to the OS itself but in a lot of cases a single malicious user space application might be enough to compromise secrets both in the host OS and in the virtual machine. This threat increases if the malware is aimed at a lower system level, e.g the kernel and processes in kernel space. If a malware is able to successfully implant itself into kernel processes virtually all secrets would potentially be accessible. An even more dangerous and probably the hardest detectable type of system infection can be created by injecting malware into code that is executed well before any kernel code is executed[19][20]. This type of malware is commonly referred to as a rootkit. Rootkits are especially problematic since common system tools and anti virus programs fail to provide means of detection or protection against them. At this point it must be obvious that all named and unnamed threats to the hardware and software configuration underlying a virtual environment also apply to a virtual environment itself. However, the biggest and commonly undetected threat for a virtual environment and a virtualized network function in particular is a malicious hardware/software configuration in its host OS. Although not mentioned frequently, the hypervisor should also be able to determine a VM's health in terms of platform and software configuration. A prominent attack against hypervisors and their host OS is called *VM escape*. VM escape attacks try to execute code on the hypervisors behalf, thereby bypassing any security mechanisms of the system and gaining full control of the physical platform. This kind of attack could be performed by creating malicious VM images or infecting running VM images.

C. Attestation Requirements

The remote attestation process itself has been defined by [12], however, MITRE[8] has proposed five principles or requirements for remote attestation and remote attestation architectures.

MITRE[2] suggests using information as fresh as possible. The supplied information should contain measurements of a running system rather than just disk images. We would like to add to this principle the use of a concrete technologies or measurement architectures like trusted boot and IMA to provide fresh information from boot time up until launch of user applications.

[1]https://www.openstack.org/

[2]http://www.mitre.org

Principle I is fresh information, although the wording might suggest the usage of a Nonce as something the appraiser uses to challenge a target in order to mitigate replay attacks, this requirement refers to the type of information supplied to the appraiser.

Principle II Comprehensive information. An attestation mechanism should be capable of delivering meaningful information about the target. An appraiser should be capable to evaluate a targets internal state based upon the supplied information. This requirement can be partially fulfilled by integrating IMA's log file, i.e. Stored Measurement Log (SML), along with a TPM Quote over the requested PCR. By providing the PCR as well as the SML an appraiser is able to verify the integrity of this log as well as the resulting platform state. As suggested by [8][9] this also raises privacy issues which we declare as out of scope in a private cloud and NFV scenario.

Principle III - Constrained disclosure. A target should only reveal information on its internal state to parties which can be identified as valid appraisers. This can be achieved by forcing the appraiser to reveal information about itself[8], by the use of cryptography, or by using exclusive channels over which attestation information is exchanged.

Principle IV - Semantic explicitness of attestation contents should be guaranteed and contents should be provided in a logical form. The semantics used in the attestation should not only allow the identification of the target but also allow an appraiser to correlate multiple attestations. Consequently, an appraiser should be able to infer consequences from several attestations, e.g. when different measurements of the target jointly imply a prediction about its behavior. This requirement can easily be fulfilled, again, by using IMA's SML as the crucial part of a reporting mechanism.

Principle V - Using a trustworthy mechanism by which information/evidence is delivered to an appraiser. This helps to guarantee that the information/evidence itself can be trusted. This requirement is typically met by the use of a TPM Quote, which is equivalent to a digital signature thereby ensuring the integrity and authenticity of the provided information.

These principles are sufficient for an individual platform, e.g., a PC or IoT device. To support virtualization, we add two more principles in accordance with the ongoing standardization work[12] considering the nature of virtual environments:

Principle VI - Layer Linking. Layer linking requires for a VM attestation to be tied to an attestation of its underlying components which might be the hypervisor or host OS. Attesting a virtual platform without an inspection of its underlying physical platform is possible using the same remote attestation protocols, however, conclusions based solely on these attestations are vague.

Principle VII - Scalability. Based on Principle VI, an attestation of a physical platform should be provided alongside an attestation of a virtual platform to support the claims made in the information received by an appraiser. The TPM Quote command is a time consuming operation with delays up to 500ms and above. Attestation schemes must treat this as a bottle neck and refrain from frequent usage or provide means to fulfill VI with the physical limitations.

IV. Related Work

HP presented an approach for Trusted Computing in NFV[21] infrastructure to IRTF[3] and IETF 93[4][22]. Unlike our proposal, it does not consider VM attestations and mimics concepts of Trusted Compute Pools[23] and TClouds[24].

V. Prior Work

This section describes *Separate Hypervisor and VM attestation* (V-A) and *Deep Attestation* (V-B) as prior work.

After the topic of vTPMs came up in 2006 [7] multiple papers were published on popular topics such as attestation and migration in all sorts of environments (private clouds, public clouds, cloud mash-up services, and federal networks)[10]. The solutions proposed, no matter what the target infrastructure was, were independent but with a common similarity: the vTPM is used as if it were a real TPM - with almost all of its benefits and almost all of its physical limitations. We will not discuss concepts for VM migration; instead, we summarize two common approaches for VM attestation.

A. Separate Hypervisor and VM Attestations

This approach attests virtual and physical environments using the same attestation mechanism. Unfortunately, it is not resource-conserving and the physical TPM becomes a bottle-neck for attestations.

Approaches treating VM and hypervisor attestations equally have benefits: They are easy to integrate into existing attestation architectures, they require few modifications to the attestation implementation on a target system, and VM's and physical devices can be treated equally from the perspective of an appraiser. (Figure 4)

However, approaches based on separate attestations can only work under specific assumptions: VMs and associated vTPMs are strongly isolated from other VMs, and from the hypervisor itself. Meaning, that through mechanisms a hypervisor must be prevented manipulating a VMs execution. Furthermore, a vTPM process once launched must be isolated from the hypervisor as well. Under no circumstances must information contained in the vTPM be exposed to any other party than the associated VM. If sufficient mechanisms were in place to validate these assumptions a VM could be attested using the same remote attestation protocol as a physical platform while creating equally meaningful evidence.

An appraiser can now believe that these isolation mechanisms are in place and operational when attesting a VM or has to assess the state of the underlying platform. This means that scalability will suffer as for every VM attestation a separate hypervisor attestation is required. Strong isolation also precludes Layer Linking (Section - III-C) between VM,

[3]https://irtf.org/

[4]https://www.ietf.org/meeting/93/

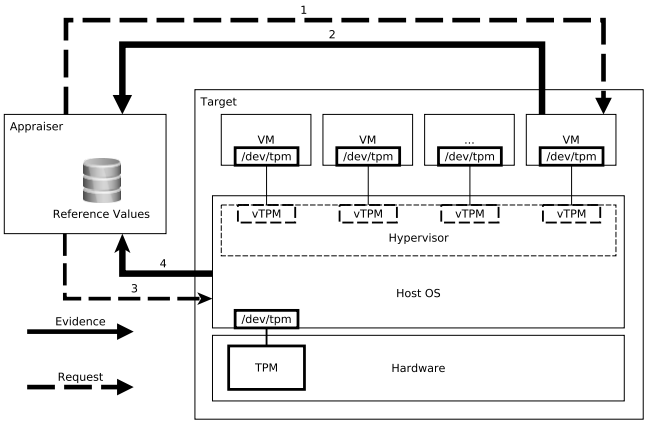

Fig. 4. Separate hypervisor and VM attestation. The appraiser attests the VM first and continues attesting the hypervisor using the same attestation mechanisms and protocols.

hypervisor, and physical platform. When attesting a VM, an appraiser would need to receive reliable information about the underlying platform. This could be achieved by maintaining a database containing VMs and their localities. No VM launch or migration must be allowed without updating the database. This requirement leverages this approach to a new level of complexity, because mechanisms have to be in place that either (i) prevent the hypervisor from changing a VM's locality without authorization or (ii) force the hypervisor to report changes to a VM's locality.

Alternatively, a VM and a hypervisors attestation could be linked using a secret that is only known to the VM and its hypervisor. This could be achieved by sending a nonce *N* unknown to a hypervisor to a VM. The VM can now be attested using *N*. Subsequently, the expected hypervisor is attested. Unlike traditional approaches, the hypervisor does not receive a nonce. The appraiser expects the VM to contact its hypervisor through an exclusive channel and supply him with *N*. Unfortunately, this approach also violates principle vii (Scalability) and no longer treats VMs and hypervisors equally.

As final remark on separate VM and hypervisor attestations, we emphasize the order of attestations: A VM attestation must be followed by an attestation of its hypervisor. This principle is based on the theory of causal ordering [11] and the technicality that a hypervisor could influence a VM. This implies that whenever a VM is attested the supporting evidence remains questionable until its underlying hypervisor has also been attested, proving that at the time of the VM attestation the hypervisor was also in a good state. Consequently, an appraiser might be able to use the same protocols for individual attestations it will not be able to treat the created evidence equally. Ultimately, separate hypervisor and VM attestations work under an extensive set of assumptions and preconditions which leave room for attackers.

B. Deep Attestation - Attestation in Standards

Deep Attestation mechanisms attest hypervisors indirectly through VMs using VM attestation credentials (eAIK) bound to physical platform PCRs representing the platform configuration, unfortunately, this mechanism provides insufficient evidence freshness.

TCG's Virtualized Platform Work Group (VPWG) has defined an attestation scheme that minimizes physical TPM use by employing the concept of ephemeral Attestation Identity Keys (eAIK)[12] in a VM. The concept is based on the fact that every VM has a set of credentials in its own vTPM, consisting of at least an Endorsement Key[1]. Once a VM is associated with a vTPM an eAIK is created. This eAIK is now used whenever the VM is attested. (Figure 5)

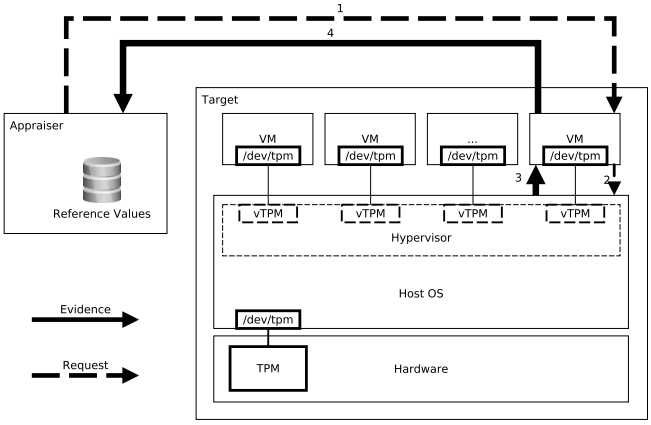

Fig. 5. Deep Attestation Method as proposed by TGC. The concept is based on indirect attestation of the underlying hypervisor through a VM using credentials in the vTPM bound to PCR values in the platform TPM.

The Work Group requires that the eAIK to reliably indicate the state of the underlying platform: in particular, whenever the platform state changes, then so must the eAIK values. This synchrony could be achieved by binding the eAIK to a set of PCR values of the hypervisor/host OS. Unfortunately, PCRs are either predictable or they reflect the platform state in detail. The lower-numbered ($<$10) PCRs in a TPM contain measurements of the system's boot sequence. They will, for instance, contain measurements of the BIOS or the boot record. A main problem of binding information such as keys to a TPM's PCRs is the consistency of PCR values over time. Keys are bound to PCR values by policies that define the PCR value(s) required before the TPM is allowed to use the key. Up to a point, e.g., loading the kernel, which may include measurements of individual modules, a set of expected PCR values can be defined and used as a policy or key restriction. Beyond this point, however, bindings to PCR values can become fragile because the loading process and the order in which measurements are performed and extended into PCRs

can become randomized by runtime-specific optimizations. This limits the span of freshness of information that can be supplied using an eAIK bound to PCRs. Unfortunately, binding key availability to PCR values also complicates the usage of advanced measurement architectures like IMA.

VI. Hypervisor-based Attestation

Hypervisor-based attestation, in contrast to other solutions, does not contact and attest VMs directly; instead, the appraiser contacts the hypervisor and receives a collection of VM attestations attached to a single physical TPM attestation.

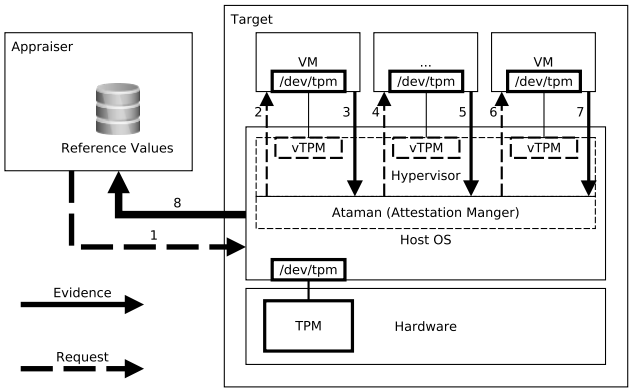

Fig. 6. Hypervisor-based Attestation. The Appraiser attests the Target including *n*-VMs. Ataman (Attestation Manager), a process in the Host OS or hypervisor, collects individual evidence from the VMs. Subsequently, the Target sends a response containing its own and each VMs evidence.

Key points of Hypervisor-based Attestation:

- **Bottom-Up Attestation:** We propose an attestation process that is initiated in the hypervisor. An appraiser sends a request to a target-platform, the target then collects VM evidence through mechanisms (*Ataman*) and attaches it to its own attestation data in a response to the appraiser.
- **Bulk VM Attestations:** This is done by the Attestation Manager (*Ataman*). Ataman is responsible for collecting individual vPCRs and vSMLs of VMs and creating the data structure that is finally attached to a single hypervisor attestation.
- **On-Block evaluation:** The appraiser receives the platform attestation with the VM attestation data attached to it. It can now verify the platform using stored reference values. Later, the same verification process can be used for the attached VM evidence. However, a mechanism is required to validate the integrity, freshness, and authenticity of the attached VM data.

Our proposed attestation method is not only scalable for large cloud environments but is also significantly less expensive for an appraiser in terms of protocol overhead for attesting individual VMs and evaluation of each vTPM quote and the associated TPM quote.

Hypervisor-based attestation is based on the requirement that for each VM attestation there has to be a hypervisor attestation as well. Contrary to existing approaches, we propose a bottom-up instead of a top-down attestation process, meaning that our attestation scheme attests VMs through their hypervisor instead of attesting hypervisors indirectly or directly through VMs. Our bottom-up design makes full use of the private cloud and NFV scenarios which typically have only one owner attesting VMs and hypervisors.

A. Attestation Scheme & Protocol

Our proposed remote attestation protocol is illustrated in Figure 7. It consists of the two parties *Appraiser* and *Target* with its internal component *Ataman*, although the architecture could be extended by adding a Trusted Third Party (TTP). The TTP could be used to manage key exchanges between Appraiser and Target Platform to fulfill privacy, confidentiality, and authenticity requirements. Our attestation scheme requires that a Target's identity credential, i.e. the public portion of its AIK, is known to the Appraiser.

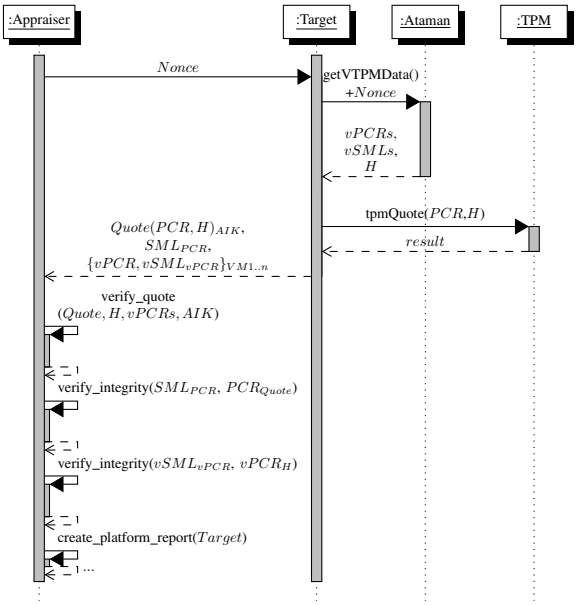

Fig. 7. Interaction and protocol steps for the main components in Hypervisor-based Attestation.

The protocol consists of the following steps:

1) The Appraiser sends a $Nonce$ to the party it needs to attest - the nonce could be part of a defined attestation request. The request format is specific to the deployment scenario and out of scope.
2) Upon reception of the nonce, the Target contacts the local attestation manager (Ataman) which is responsible for collecting data from VMs running on the platform. This process is depicted in Figure 8 and variations of how data is accessed and collected is shown in Section VII.

3) Once Ataman has collected all PCRs and SMLs of the VMs (vPCR, vSML), individual vTPM vPCRs are concatenated using the following mechanism

$$vPcr_{m_i} := SHA1(vPcr_{m_{i-1}}||m_i) \quad (1a)$$

$$vPcr_{m_{i+1}} := SHA1(vPcr_{m_i}||m_{i+1}) \quad (1b)$$

and thus creating an integrity value over all collected vPCRs. The supplied nonce serves as the initial value, however, it could be placed at any position within the hash chain. The concatenation of Nonce and vTPM PCRs values will be referenced as H.

4) Ataman passes the following data back to the Target: $ExternalData = H$ as well as the tuple $\{vPCR, vSML_{vPCR}\}$ for each VM.
5) Target now gathers data for its own attestation using the TPM_Quote command with $ExternalData$ supplied by Ataman.
6) The Appraiser receives the following answer: TPM_Quote, including the requested PCR using H as external data + the Log, e.g. SML, as well as the tuple $\{vPCR, vSML_{vPCR}\}$ for each VM.
7) Subsequently, the Appraiser verifies the authenticity of the supplied TPM_Quote using the Targets AIK_{pub}. If successful, it evaluates the supplied external data, e.g. H. Once the Appraiser has successfully recalculated H using his Nonce and the supplied vPCRs, he can evaluate the target platform using PCR and log (SML).
8) The Appraiser uses the supplied logs, e.g., SML and vSMLs, to recalculate PCR and vPCR values while checking the contents of the log. Hypervisor-based Attestation is not bound to any specific logging mechanism, however, IMA as a measurement architecture along with SML as the log is a very common implementation.

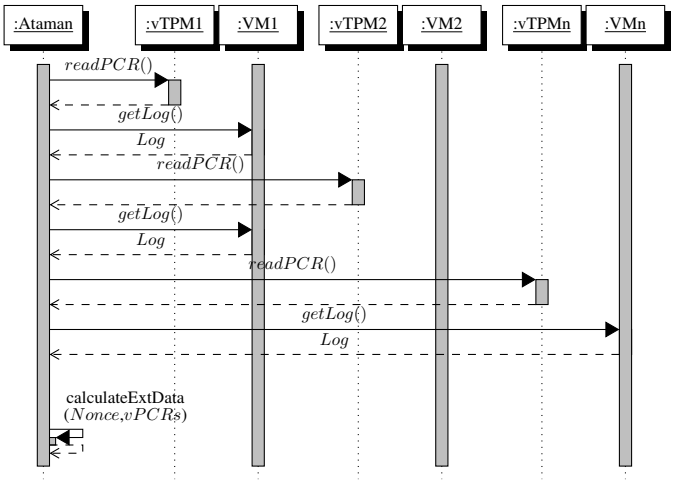

Fig. 8. Interaction between Attestation Manager, vTPM, and VM's. This is the direct access approach, Ataman doesn't have to contact VMs to gather PCRs since they can be pulled from the vTPM process running in the host OS.

Based on the supplied information, the Appraiser can now verify the the Target using the $Nonce$ as evidence of freshness and SML_{PCR} along with PCR from the Quote to verify its platform configuration. Through our addition of both vTPM PCRs and individual VM SMLs, the Appraiser can apply the same process for VM verifications. Furthermore, the Appraiser can verify the integrity and freshness of supplied vTPM PCR's and SML's using the accumulated hash value H and $Nonce$.

VII. Variants and Alternative Solutions

Variations of this scheme become necessary as soon as the accessibility of the vTPM changes. In a simple scenario Ataman can access vTPM data and thus PCRs directly via a process interface or file access to the stored measurements. However, hardening and increased isolation of vTPMs enhance data protection but lead to changes the access to vTPM data.

A variation of our proposed protocol assumes means of communication, called as channels, which exist exclusively between a hypervisor and a VM. These channels implicitly guarantee that only a VM's underlying platform can communicate through it, which makes authentication between the two communicating parties unnecessary. If Ataman can no longer directly access vTPM data, it has to contact the VMs.

The basic attestation protocol from Appraiser to the Target in Figure 7 remains unchanged, however, the protocol between Ataman and VMs changes significantly:

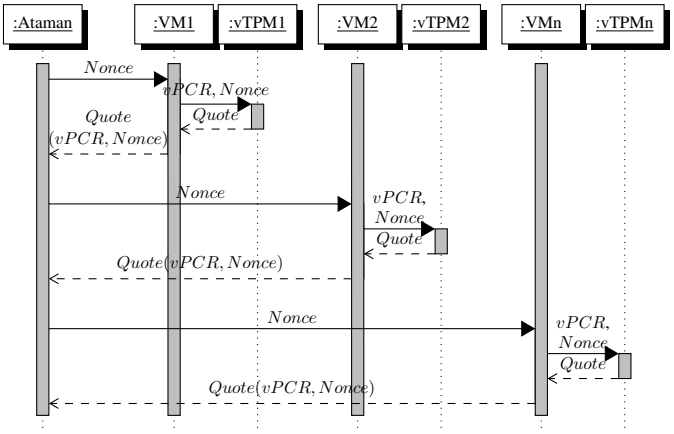

Fig. 9. Interaction between Ataman, VMs, and vTPMs. Ataman contacts VMs requesting SMLs and vTPM Quotes instead of accessing the vTPM directly.

This changes the set of information that Ataman holds after receiving responses from each VM to

$$VmEvidence := Nonce, \{\{Quote_{vTPM1}, vPCR_{vTPM1} SML_{vTPM1}\}, ..., \{Quote_{vTPMn}, vPCR_{vTPMn}, SML_{vTPMn}\}\}$$

which means that Ataman collects individual attestations from each VM in the same way the Appraiser does so for the platform.

There are two strategies for handling this information. One approach would be to create

$H := SHA1(VmEvidence||Nonce)$, using H as external data in the TPM_Quote, and attach the collected $VmEvidence$ as proposed above. Please note that SHA-1 was chosen to ensure TPM1.2[1] algorithm compatibility, improved hash functions used in [2] are recommended. This approach would mean that the Appraiser has to verify the TPM_Quote as well as the individual vTPM_Quotes in addition to their vPCRs and vSMLs. Consequently, an Appraiser would be verifying individual VM attestations using the same process as for the hypervisor attestation to which they are attached. Consequently, this means that AIKs used by vTPMs have to be known to an Appraiser which creates overhead.

An alternative approach would require Ataman to verify whether supplied vPCRs are both fresh and authentic, i.e. Ataman verifies the vTPM_Quotes over vPCRs and the supplied $Nonce$. This process is depicted in Figure 10.

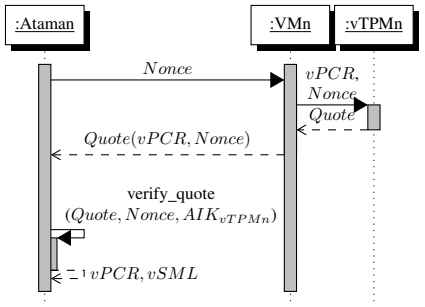

Fig. 10. Interaction between Ataman, VMs, and vTPMs. Ataman contacts VMs requesting SMLs and vTPM Quotes, verifies them, and collects vPCRs and vSMLs

As a result, the data structure and information sent from the Target Platform to the Appraiser remains compatible with the original scheme. Thus, an Appraiser does not need to have knowledge about vTPM AIKs while still being able to attest the Target Platform and VMs running on it.

A practical realization of exclusive channels are Virt I/O[5] interfaces provided by KVM, sockets, or authenticated TCP/IP interfaces.

VIII. Conclusion and Future Work

By using a mechanism in the hypervisor collecting measurements, i.e., *Ataman*, an implicit layer linking is created. Moreover, the correct deployment and operation of the linking mechanism can be verified using the hypervisor attestation to which VM measurements are bound.

Furthermore, our attestation approach makes efficient use of TPM Quotes while providing full support for advanced measurement architectures like IMA which represents a significant enhancement to a deep attestation method as proposed in the current specification (Section V-B), VPWG [12].

Further efforts must go into exploration and evaluation of possibilities for integrating the proposed approach into existing hypervisors like QEMU or Xen, and cloud platforms such as OpenStack. Finally, this type of attestation can be adopted by TCG's specification for virtualized environments.

[5]http://www.linux-kvm.org/page/Virtio

Acknowledgments

We would like to thank Stefan Berger and Kenneth Goldman of IBM for providing the TPM Software Stacks and vTPM implementations used while developing our concepts. Furthermore, we thank TCG's Virtualized Platform Work Group for their encouraging support and feedback. Finally, we would like to thank Bede McCall for his thorough reviews and comments.

References

[1] Trusted Computing Group, *TPM Main Specification Version 1.2*, Revision 116 TCG, 2011.
[2] Trusted Computing Group, *Trusted Platform Module Library Specification, Family 2.0*, Revision 01.16 - 2014 TCG, 2014.
[3] Trusted GRUB. http://http://trousers.sourceforge.net/grub.html
[4] Trusted Boot. https://software.intel.com/en-us/articles/intel-trusted-execution-technology-intel-txt-enabling-guide
[5] Sailer, Reiner and Zhang, Xiaolan and Jaeger, Trent and van Doorn, Leendert, *Design and Implementation of a TCG-based Integrity Measurement Architecture* Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13, San Diego, 2004.
[6] Sachiko Yoshihama,Tim Ebringer, Megumi Nakamura, Seiji Munetoh, Hiroshi Maruyama. *Efficeient and Fine-Grained Remote Attestation on Web Services* IBM Research Report, Tokyo, 2005
[7] Berger, Stefan and Cáceres, Ramón and Goldman, Kenneth A. and Perez, Ronald and Sailer, Reiner and van Doorn, Leendert, *vTPM: Virtualizing the Trusted Platform Module* Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15, Vancouver, B.C., Canada, 2006
[8] George Coker, Joshua Guttman, Peter Loscocco, Amy Herzog, Jonathan Millen, Brian O'Hanlon, John Ramsdell, *Principles of Remote Attestation* The MITRE Corporation, National Security Agency, 2011.
[9] Ahmad-Reza Sadeghi, Christian Stble, Marcel Winandy, *Property-Based TPM Virtualization* Ruhr-University Bochum, Bochum, 2008.
[10] Antonio Celesti, Maria Fazio, Massimo Villari, Antonio Puliafito, Davide Mulfari, *Remote and Deep Attestation To Mitigate Threats in Cloud Mash-up Services* 2013 World Congress on Computer and Information Technology (WCCIT), Sousse, 2013.
[11] Lamport, Leslie, *Time, Clocks, and the Ordering of Events in a Distributed System* Commun. ACM, New York, 1978.
[12] Trusted Computing Group, *Virtualized Trusted Platform Architecture Specification*, Version 1.0 TCG, 2011.
[13] J. Rutkowska, *Introducing Stealth Malware Taxonomy*, Version 1.01 COSEINC Advanced Malware Labs, 2006.
[14] European Telecommunications Standards Institute (ETSI), *Network Functions Virtualisation (NFV); Use Cases*, Version1.1.1 ETSI GS NFV 001, 2013.
[15] European Telecommunications Standards Institute (ETSI), *Network Functions Virtualisation (NFV); Architectural Framework*, Version1.1.1 ETSI GS NFV 001, 2013.
[16] European Telecommunications Standards Institute (ETSI), *Network Functions Virtualisation - Introductory White Paper*, Issue 1 SDN and OpenFlow World Congress, Darmstadt-Germany, 2012.
[17] HP, *Network Functions Virtualisation - Technical White Paper* http://www.hp.com/.
[18] Cisco, *NFV - Network Functions Virtualisation* http://www.cisco.com/.
[19] Arstechnica, *"Unauthorized code" in Juniper firewalls decrypts encrypted VPN traffic* Arstechnica, 2015.
[20] Threatpost, *Attackers Replacing Firmware on Cisco Routers* Threatpost, 2015.
[21] IRTF NFVRG, *Building Blocks Towards a Trustworthy NFV Infrastructure* Hewlett-Packard Laboratories, July 22nd, 2015
[22] IETF 93 - SDNRG meeting, *A SDN Attestation Approach* IETF 93, July 22nd, 2015
[23] Intel Trusted Compute Pools, *Trusted Compute Pools with Intel Trusted Execution Technology* www.intel.com
[24] Trusted Clouds, *Trustworthy cloud computing whitepaper* https://www.zurich.ibm.com/ cca/papers/tclouds-white.pdf