Improving System Trustworthyness by Combining Remote Attestation and Root Cause Analysis

Ian Oliver, Gabriela Limonta, Borger Vigmostad *
*Nokia Bell Labs, *Espoo, Finland*
Email: *ian.oliver@nokia-bell-labs.com

One particular concern in Telecommunications networks is the provisioning of quality of service despite system failures. Said failure invariably encompass many parts of the system. For example, a simple networking misconfiguration can result in the loss of telecommunications services for an entire country; though most failures are much less mundane if still critical. As system security is hardened through various techniques, the complexity of tracing faults through the myriad of interconnected systems becomes more complex.

In this paper, we present an integrity monitoring architecture based around remote attestation (RA) of systems using the Trusted Processor Module (TPM), describe the kinds of faults that can be detected and present the basis of root cause analysis (RCA) procedures for understanding these faults.

Telecommunications systems are moving to cloud based infrastructures using the NFV architecture defined by the Europian Telecommunications Standards Institute (ETSI). This has many benefits, including the ability to effectively optimise and scale the provisioning of telecommunications [1], [2].

With a large number of interacting elements, operators need to keep track of the integrity of the system in terms of what system software and hardware configurations are running and detect changes in these. Such changes imply updates, patching, hacking, failures and misconfigurations. This is achieved using a combination of a trusted computing base utilising the TPM (specifically TPM 2.0) and remote attestation over time along with an understanding of the coupling between systems.

The TPM acts as the root of trust for a system providing secure storage mechanism, cryptographic functions, public-private key generation with a measurement storage mechanism. The measurements of system components are made at pre-boot, kernel load and potentially at run-time.

During boot, kernel load and run-time, different components are hashed, and those measurements stored in platform configuration registers (PCRs) in the TPM. These components include the core root of trust, BIOS, operating system and filesystem.

These registers are then accessed through the TPM_Quote mechanism which returns a data structure and signature corresponding to the selected set of measurements stored in the PCRs. The quote is generated internally inside the TPM itself and signed by either the chip's attestation key or a given, suitable signing key.

The decision whether a given machine is trusted is to check a quote's attested value against a known good value and ensure that the quote itself has been signed by the TPM belonging to the machine providing the quote.

$$\text{trusted}(q, s, ev) = \begin{cases} \mathbf{1}, & q.attested = ev \wedge signed(q, s) \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{1}$$

One of the problems with the TPM architecture is that it was designed originally to protect a single machine a boot-time only through a mechanism known as launch control policies. This has been extended for the distributed and virtual environments by the quote mechanism described earlier. To facilitate this, the use of an RA component is used to interrogate a machine of its status.

The RA contains data of known machines and expected values for those machines. It has the responsibility for connecting to those machines at an appropriate time (eg: after boot), requesting a quote, ensuring the quote is valid and signed and then comparing it against known good values.

Additionally, the RA has a role in virtual workload placement, since the physical hardware should satisfy the trust requirements. For example, in OpenStack, when a virtual machine is being placed a set of requirements, such as available CPU and memory are considered for the underlying machine; RA adds a trust constraint into this.

Finally, the RA should report trust failures to other NFV components and not just exist as a static component. This may trigger machines to be sandboxed, removed from the system or shutdown as appropriate. Run-time failures should cause virtual machine migration/evacuation from the affected hardware - this topic however is unexplored in the literature.

A trusted machine allows us more certainty regarding the configuration of the firmware or BIOS, operating system configuration and loading mechanism and loaded software. This is utilised in virtualisation environment to present a known environment for orchestration and workload placement algorithms. Many telecommunications workloads are sensitive in nature; for example, the lawful intercept functionality has very strict requirements upon placement. Furthermore, the attestation and integrity status of the machines in an NFV infrastructure provide an additional reference point for the overall security of system and can be cross-referenced during updates and during attack events.

The TPM quote contains a number of fields of information which need to be taken together, in order to decide whether a machine is trusted *at a given point in time* or not. A few select, individual rules are shown below:

$$\text{expectedValue}(q, ev) = \begin{cases} \mathbf{1}, & q.attested = ev \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{2}$$

$$\text{signed}(m, q, s) = \begin{cases} \mathbf{1}, & validSignature(q, s, m.akpub) \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{3}$$

$$\text{resetinc}(q, q') = \begin{cases} \mathbf{1}, & q'.reset \geq q.reset \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{4}$$

One of the fields included in the quote, reset count, gives information about the number of reboots that the machine has experienced. Similarly, the restart count field identifies the number of suspend operations, power saving and certain TPM start/clear commands.

Next, we describe how rule sets are constructed, applied and notions of strictness. We can define a minimal idea of trust for a given machine, in the form of a rule set, as:

$$trusted_{min}(m,q,s,p,V) = expectedValue(q,p) \wedge signed(m,q,s) \quad (5)$$

This minimal definition can be tightened by extending existing rule sets with new rules. Therefore, creating a hierarchy of strictness for the different definitions of trust.

RCA can be used to understand the why a machine may fail its trust checks. Figure 1 shows a causal factor tree (CFT) for the scenario in which there are inconsistencies between the reset count reported by the TPM and what has appeared in reality. The TPM reset account can be considered to be accurate (unless the TPM has been replaced in which case we will see additional failures). In this case, it is necessary now to understand the interaction of other systems, eg: reboots without notification to the virtual infrastructure manager, loss of networking, machine self-test etc.

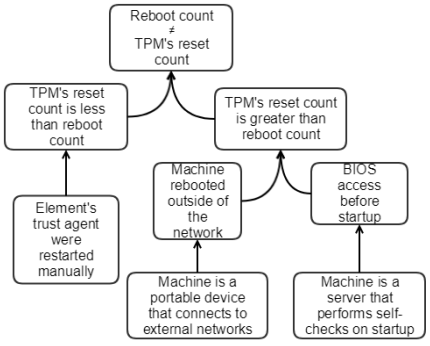

Fig. 1. TPM Reset Count Failure CFT

This kind of failure can be seen with server class hardware which often will perform multiple reboots during self-test procedures before the 1st stage boot loader is executed. For example, BIOS updates generally imply multiple TPM resets for a single 'boot' accompanied by a possible change in measurements, especially those regarding the BIOS state and configuration.

An Ishikawa diagram describing the above is shown in figure 2. Using this form of RCA analysis allows us to differentiate between constituent parts of the system, such as the machine or element in question, the attestation server and the network. Though, for constructing the mitigation database, the previous CFT mechanism has proven to be more workable.

Additionally, we are now able to identify situations where trust or trustworthiness can be recovered in a safe manner. Figure 2 can help us identify those elements which perform multiple reboots and under what conditions, which allows us to modify or introduce additional rules; for example, in eqn. 6.

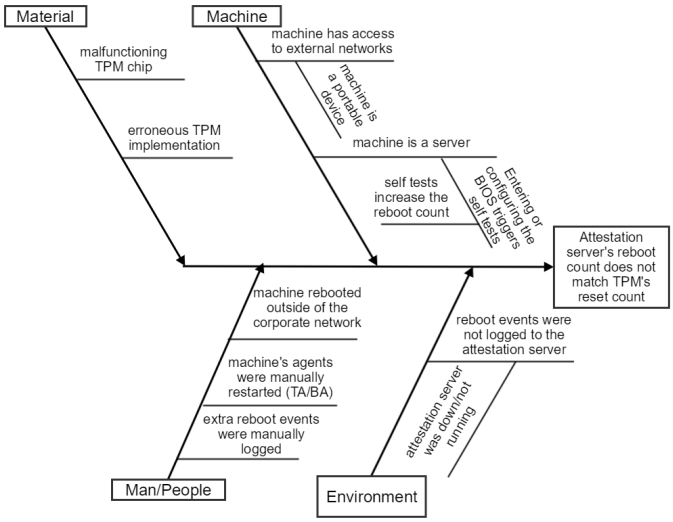

Fig. 2. Reboot Count Failure Ishikawa Diagram

$$\text{afbootinc}_1(m,q,q') = \begin{cases} \mathbf{1}, & q.pcr_1 \neq q'.pcr_1 \wedge \\ & q'.reset \geq q'reset + 2 \\ \mathbf{0}, & \text{otherwise} \end{cases} \quad (6)$$

Any new rules can be added to a rule set specific for that given element or class of elements and placed in the strictness hierarchy for the definition of trust.

In this paper, we have presented the use of a TPM and attestation server in a cloud environment extended by a more complex set of rules to decide the trustworthiness of any given element. The rules are then linked to CFTs as part of an RCA process to assist in both system diagnosis and forensics, and in the overall trustworthiness decision.

Future work includes the creation of a trust graph of which elements trust each other based on the information from trust decisions [3]. This graph can then be utilised by external mechanisms; for example, network routing provisioning can provide a more trusted environment for running sensitive and critical workload.

ACKNOWLEDGEMENT

This work has been partially funded by EU ECSEL Project SECREDAS (Grant Number: 783119) and EU Horizon 2020 Project SCOTT (Grant Number: 737422).

REFERENCES

[1] OLIVER, I., HOLTMANNS, S., MICHE, Y., KALLIOLA, A., LAL, S., , AND RAVIDAS, S. Experiences in trusted cloud computing. In *11th IEEE International Conference on Network and System Security, NSS 2017, Helsinki, Finland, August 2123* (2017), Springer.

[2] OLIVER, I., LAL, S., RAVIDAS, S., AND TALEB, T. Assuring virtual network function image integrity and host sealing in telco cloud. In *IEEE ICC 2017, Paris, France* (May 2017).

[3] OLIVER, I., LIMONTA, G., VIGMOSTAD, B., KALLIOLA, A., MICHE, Y., HOLTMANNS, S., AND MULLER, K. A testbed for trusted telecommunications systems in a safety critical environment. In *13th International ERCIM/EWICS/ARTEMIS Workshop on Dependable Smart Embedded and Cyber-physical Systems and Systems-of-Systems - DECSoS 2018, Västerås, Sweden* (September 2018).