Assuring Virtual Network Function Image Integrity and Host Sealing in Telco Cloud

Shankar Lal *†, Sowmya Ravidas*†, Ian Oliver* and Tarik Taleb†
*Nokia Bell Labs, *Espoo, Finland*
†Aalto University, *Espoo, Finland*
Email: *first.last@nokia-bell-labs.com, †first.last@aalto.fi

***Abstract*—In Telco cloud environment, virtual network functions (VNFs) can be shipped in the form of virtual machine images and hosted over commodity hardware. It is likely that these VNF images will contain highly sensitive data and mission critical network operations. For this reason, these VNF images are prone to malicious tampering during shipping and even after uploaded to the cloud image database. Furthermore, due to various applications, there is a requirement from mobile network operators to seal VNFs on specific platforms which satisfy certain hardware and software configurations. This requires cloud service providers to introduce some mechanisms to verify VNF image integrity and host sealing before the instantiation of VNFs. In this paper, we present a proof of concept demonstrated with the help of an experimental setup to solve the above-mentioned problems. We also evaluate the performance of the envisioned setup and present some insights on its usability.**

I. Introduction

Network function virtualization infrastructure (NFVI) provides numerous benefits for the transition of Telco hardware functions into software-based functions leveraging popular virtualization technologies. Several research works have been conducted to explore the potentials and challenges in Telco cloud [1]. Researchers have also manifested the virtualization possibilities of existing physical network component such as the evolved packet core (EPC) to unveil the potential of NFVI [2].

Along with the advancements in the area of cloud and NFV, mobile network operators (MNOs) are considering to deploy their services in the cloud. One of the major challenges they are facing, is how to verify the integrity of cloud infrastructure in order to establish trust in the cloud. The notion of trust is an important factor to consider, especially when mission critical telecommunication functions are placed in the cloud.

In cloud environment, virtual machine (VM) images are used to launch VM instances, which act as virtual servers to deliver the specified services. In NFVI, VNFs can be deployed over a single VM or multiple VMs. In the latter case, each VM will host one component of the VNF [3]. As a matter of fact, there are many threats present in the existing environment to compromise the VNF image integrity [4]. It is easy to tamper a VNF image by inserting malicious code or software into it while it is in transit or in rest on cloud image database. If a VNF image needs to be transfered via internet and there are no appropriate security methods in use, then there exists possibility of injecting malware into the VNF software image during the transit. One similar study is performed in [5], where a researcher found anonymity network TOR (The Onion Router) exit node adding malware to the software binary files while they are being downloaded from the internet. There is also a possibility that cloud image database can become compromised when it is located in a network with limited security protection [6]. In addition to that, cloud administrators can have enough privileges to access and copy or steal sensitive data in a VNF image as demonstrated by Rocha et, al and Khan et, al in [7], [8]. These threats expose the risks of violation of VNF data integrity, privacy and confidentiality.

Besides integrity verification of VNF software images, MNOs also require their VNFs to be placed on the hosts with certain hardware and software configurations such as the choice of the operating system (OS), hypervisor and also geographical location of the underlying physical host as depicted in Figure 1. This process can be termed as VNF-Host sealing. VNF-Host sealing can be defined as the process which uses trusted platform module (TPM) to bind some VNF instance to a specific compute host which satisfies the given set of system configuration policies. There is another concept, which sounds similar to VNF-Host sealing, known as virtual machine sealing as introduced by Red Hat. Contrary to VNF-HOST sealing, virtual machine sealing is actually a process of removing all system-specific details from a virtual machine instance before creating an image template from it [9].

The rest of this paper is structured as follows. Sections II presents some related work. Section III discusses the components needed to setup a trusted cloud. In Section IV, we present the testbed experimental setup along with its performance evaluation. In Section V, we provide the use cases of VNF image integrity verification and VNF-Host sealing and discuss the different relevant challenges, defining new directions for future research work. Section VI concludes the paper recapping the main findings of the envisioned experimental setup.

II. Related Work

In [10], a survey on NFV security was performed. This survey lists some of the relevant security challenges. Some of these pertain to the need-to-use standard security mechanisms in NFVI, defining the standard interfaces, challenges associated with Management and Orchestration (MANO) and also the elasticity of VNFs. While this survey paper does not deal with integrity verification as a need, it mentions trust management as one of the future challenges.

According to [11], integrity verification of some software can be achieved by setting up a signing authority. Signing

authority produces the cryptographic signature of the original software and validation authority verifies the signature. In practice, cryptographic hash digest of software image is calculated in the beginning and signed using the private or root key of the signing authority to produce the signature. For validation, fresh hash digest of the software image should be re-calculated. The validation authority uses this fresh hash digest along with the associated signature to check the software integrity. In order to detect the tampering of VNF images, signing and verification authorities can be setup in two different ways as below:

1) Performing signing and verification locally on the same system.
2) Involving trusted third party to perform signing and verification.

OpenStack cloud infrastructure is introducing VM image signature verification feature in its future release [12]. In this feature, signing and verification entities are setup locally on the controller node. Cloud users can generate the image signature and public key certificate and store it on the controller node. The issue with this technique is that the signature verification is performed locally. In case, when cloud image database is compromised, it would be easy for an attacker to add his own signature and public key certificate in the image metadata. This will not be detected since image signature would still be valid. Also, there is no protection from malicious administrator who can have enough privileges to compromise the VNF image integrity. Performing signature verification externally and preferably by involving a trusted third party is the wise approach.

VNF sealing to a certain host is relatively a new problem. In traditional computers, an analogous to this problem is addressed by using a technique called CPU pinning, whereby the processes are bound to specific CPUs and are allowed to be executed only on them. In a cloud scenario, CPU pinning refers to the pinning of virtual CPUs (vCPUs) of VNF instances to the physical CPUs of the host [13]. This is useful in scenarios whereby two guest vCPUs compete for CPU time of the host, which might lead to high latency of the work load running on the VNFs. CPU pinning avoids this latency by allocating vCPUs to specific threads in the host, thereby, efficiently balancing the workload execution on the vCPU [14]. While this solution can guarantee the SLA of running VNFs on certain physical CPUs, it does not state the platform configurations of the host required to VNF instance.

III. Components of Trusted Cloud

In this section, we present the techniques to set up a trusted cloud environment in which cloud nodes are able to verify their boot time integrity measurements. TPM module, trusted boot (tboot) tool and a remote attestation service are used to accomplish this setup. The components of trusted cloud are discussed hereunder.

A. Trusted Platform Module (TPM)

Trusted computing group (TCG) has provided TPM specifications and recommended to use TPM module to store passwords, cryptographic keys, certificates and other sensitive information. TPM contains platform configuration registers (PCRs) which can be used to store cryptographic hash measurements of the system's critical components [15], [16]. There are in total 24 PCRs in most TPM modules starting from 0 till 23. Figure 2 depicts these PCR registers and their association with the system's components.

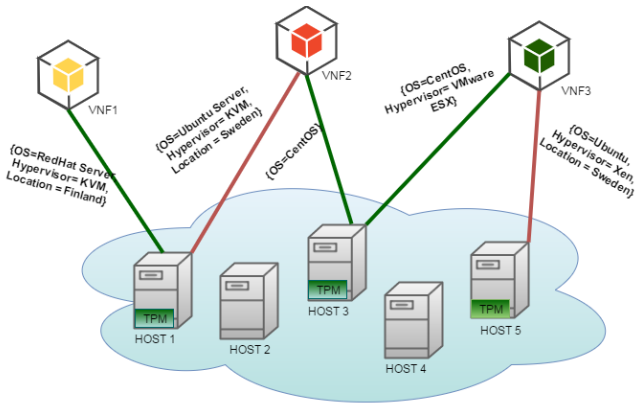


Fig. 1. General idea of VNF-Host sealing

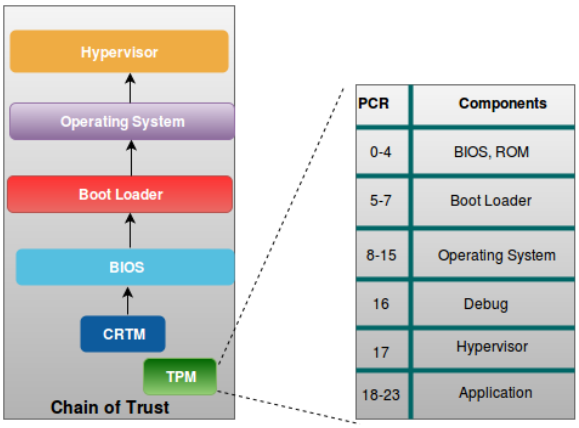


Fig. 2. TPM platform configuration registers (PCRs)

B. Trusted Boot

Trusted boot (tboot) is an open source tool which uses Intel's trusted execution technology (TXT) to perform the measured boot of the system. Trusted boot process starts when tboot is launched as an executable and measures all the binaries of the system components (i.e., firmware code, BIOS, OS kernel and hypervisor code). Tboot then writes these hash measurements in TPM's secure storage [17]. It should be noted that trusted boot and UEFI (Unified Extensible Firmware Interface) secure boot are two different technologies used for different purposes. Trusted boot is used to measure the boot loader and other system components, whereas UEFI secure boot is used to authenticate if the system has booted up using the digitally signed boot loader and other system components. Thus, UEFI can only be used to restrict unauthenticated software to boot but it can not provide the measurement of

system components to be used for attestation. The interested reader may refer to this link [18] for more detailed explanation.

C. Remote Attestation Service

Remote attestation is the process of verifying the boot time integrity of the remote hosts. It is a software mechanism integrated with TPM, to securely attest the trust state of the remote hosts. It uses boot time measurements of the system components such as BIOS, OS, and hypervisor, and stores the known good configuration of the host machine in its white list database. It then queries the remote host's TPM module to fetch its current PCR measurements. After receiving the current PCR values, it compares them against its white list values to derive the final trust state of the remote host [19]. This process is depicted in Figure 3. Practical implementation of remote attestation service is known as open cloud integrity technology (OpenCIT). OpenCIT is an open source tool and is hosted on GitHub [20].

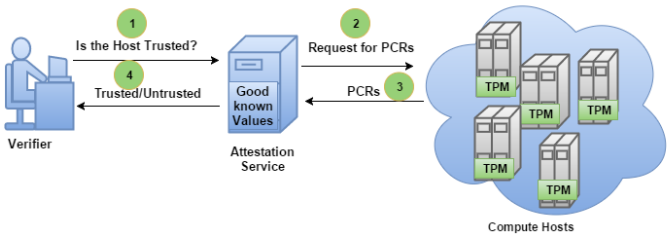


Fig. 3. Remote attestation process

D. OpenStack Resource Selection Filters

In OpenStack, when a VNF is launched, the nova scheduler filters pass through each host and select the number of hosts that satisfy the given criteria. Each filter passes the list of selected hosts to the proceeding filter. When the last filter is processed, OpenStack's default filter scheduler performs a weighing mechanism. It assigs weight to each of the selected hosts depending on the RAM, CPU and any other custom criteria to select a host which is most suitable to launch the VM instance [21][22]. Additionally, it is also possible to create own custom filters. One example of a custom filter is the trust filter, whose function is to find the hosts which are booted in trusted manner and their boot time trust state is attested by a remote attestation server.

IV. Testbed Experiment

In this section, we attempt to solve the problems of detecting VNF image tampering and VNF-Host sealing in an OpenStack-managed cloud environment. In this proof of concept demonstration, we made some modifications to the VNF startup procedure of OpenStack as explained below.

A. Implementation Setup

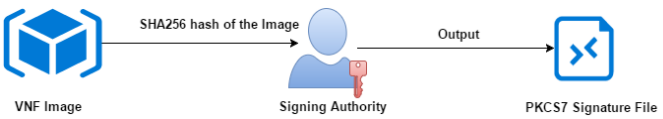
Our testbed experiment consists of four phyiscal server machines and two virtual machines, all running Debian Linux Ubuntu 14.04. This configuration setup contained two Intel Xeon Servers E5-2600 v3 @2.20 GHz with 72 GB RAM and TPM version 1.2, two HP ProLiant servers DL360 G5 having Intel Xeon CPU 5160 @3.00GHz and 24GB RAM. Both virtual machines have 2 GB RAM and 1 vCPU allocated. The OpenStack Kilo version is used as the software for managing the cloud infrastructure.

B. Virtualization Technologies Used

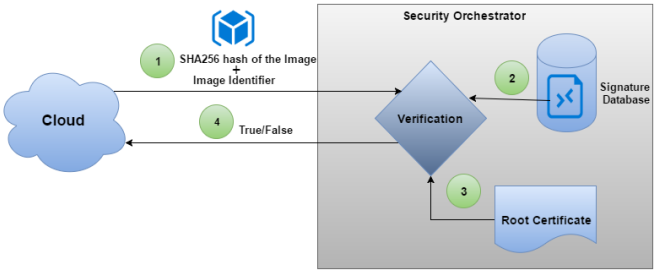
We carried out our experiment using two popular choices of virtualization technologies, namely KVM hypervisor and Docker container. OpenStack supports both virtualization technologies. For detailed functioning and performance comparison studies of these technologies, the interested reader may refer to the work of [23], [24].

C. Verification and Sealing process

In order to sign and verify a VNF image, we setup signing and verification authorities using OpenSSL. Both use asymmetric cryptographic key pair (i.e., private keys and public keys) to generate and verify signatures. In our setup, signing authority requires 32 bytes long SHA256 hash digest of the VNF image to be signed and produces PKCS7 (Public Key Cryptography Standards 7) formatted signature file as an output. Each signed VNF image has separate signature file which is used in verification process for its integrity checking. In order to verify the signature from signing authority, the fresh SHA256 digest of the image is calculated and is used as an input to the signature verify tool along with the associated PKCS7 signature file. The verification tool uses root certificate containing the public key of signing authority to verify if the image's signature are valid. The output is the decision if the image verification either succeeded or failed. The complete signing and verification process is shown in Figure 4.



(a) Signature generation



(b) Signature verification

Fig. 4. Process of VNF signature generation and verification

In this research work, we implemented a Security Orchestrator (SecO), a server written on Node.js v0.10.25 with express framework. SecO listens to the signature verification and VNF-Host sealing requests from OpenStack and provides the result back to it. SecO also hosts signature verification tool and acts as a verification authority. MongoDB database is used in SecO to store signature files and sealing policies.

We used OpenCIT as a remote attestation server. OpenCIT requires trust agent to be installed on the target host to

enable the communication with its TPM module. In our setup, OpenCIT provides the PCR measurements of the remote host whenever requested by SecO.

In order to enable the communication between SecO and OpenStack, we built two custom filters. First is the verification filter, written in Python, for OpenStack nova scheduler. The primary purpose of this verification filter is to calculate the fresh SHA256 hash digest of the VNF image and send it to SecO along with its unique identifier for signature verification. The verification filter first extract the VNF image ID from the OpenStack's base filter properties and then uses image ID to locate the VNF image location on image server in order to calculate the hash digest. In OpenStack image server (known as glance), the VNF images are named by their ID and typically stored in the directory */var/lib/glance/images/* on image server.

The second filter is the VNF sealing filter and is used to seal VNF instances to a specific host. The process of sealing is performed by associating the VNF image with one or more sets of policies in its metadata. In this experiment, VNF image policy is defined by using the PCR measurements of the remote host which possesses the desired system configuration. In our defined policy, the OS parameter is set as Ubuntu server with Linux kernel version 3.19.0-39 and hypervisor as KVM. The purpose of this sealing filter is to select the host which satisfy the conditions defined in the sealing policy. These polices can also be defined for a pool of hosts by listing the PCR measurements of all hosts in the subset. The sealing filter works by first extracting the VNF image ID and the list of available hosts from OpenStack base filter properties. It then sends the list of all the hosts to SecO for matching them with the stored policies. Upon receiving the sealing request, SecO fetches the current PCR measurements of each host from OpenCIT server. It then compares the retrieved PCR measurements against the associated policy of the VNF image. If the measurements of some host match with the policy, the VNF sealing is considered to be successful on that particular host. Figure 5 depicts this process. In figure 6, we give the complete overview of the signature verification and VNF sealing process, showing the flow of messages among all the components.

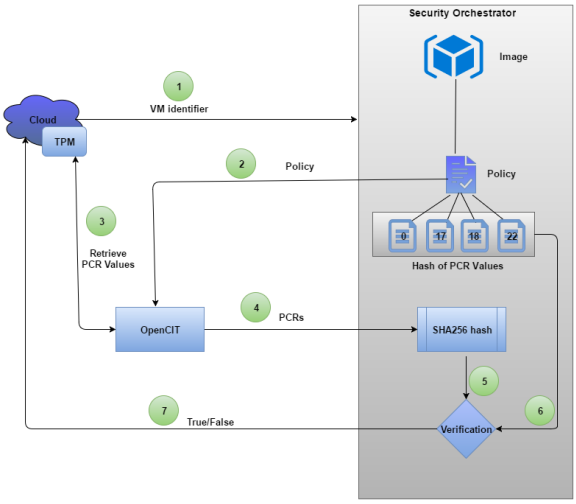


Fig. 5. A mechanism for VNF-Host sealing

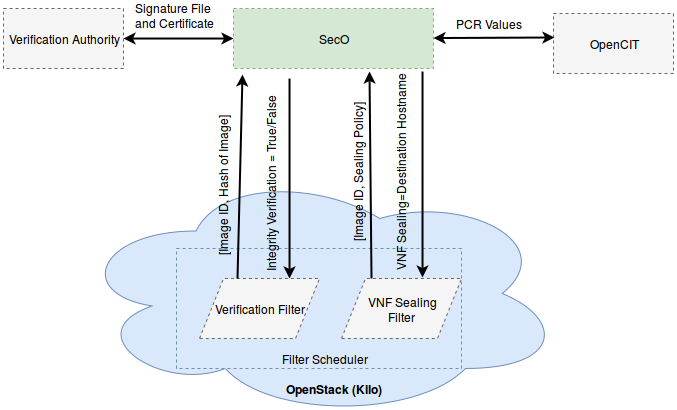


Fig. 6. Flow diagram of verification and sealing process

D. Performance Evaluation

The metrics used to evaluate the performance of this testbed setup are time overhead and response time of SecO under different loads. We first examined the correlation between various VNF image sizes and time taken to perform signature verification and host sealing over them. We performed this experiment on 5 different VNF image sizes such as 13 MB, 289.8 MB, 951.1 MB, 2.5 GB and 5 GB. The normal launch time of the VNF images versus the launch time with the signature verification and host sealing is depicted in Figure 7. The blue area in figure 7 shows the normal VNF launch time while the red bars show the launch time with the signature verification and host sealing. This time is calculated from the moment when a VNF image is launched and until it becomes available to use. From figure 7, it can be seen that the time overhead introduced by the verification and VNF sealing filter is only a few seconds for small-size VNF images. This is due to the fact that less time is needed to calculate hash digest over them. However, the time overhead increases for bigger VNF images since the hash digest calculation takes longer time. Additionally, the overhead time becomes significant only when the VNF image size exceeds 2 GB. This result looks satisfactory since we assume that 2 GB image size should be sufficient for compiling most of the VNF software images.

Next, we measured the effect of using different hashing functions to calculate the hash digest of a VNF image. This analysis helps in selecting the best hash function in terms of performance and security. Hashing algorithms tested in this experiment are MD5, SHA1 and SHA256. Although using MD5 algorithm is not a feasible option as it has been cracked down long ago, we used it in this study since it is still widely used in software integrity verification [25]. The time it takes to calculate the hash digest by these three hashing functions versus VNF image size is plotted in figure 8. It can be seen that MD5 and SHA1 algorithms take shorter time in calculating the hash digest but they are considered insecure. On the other hand, SHA256 is the most secure hashing function but it incurs

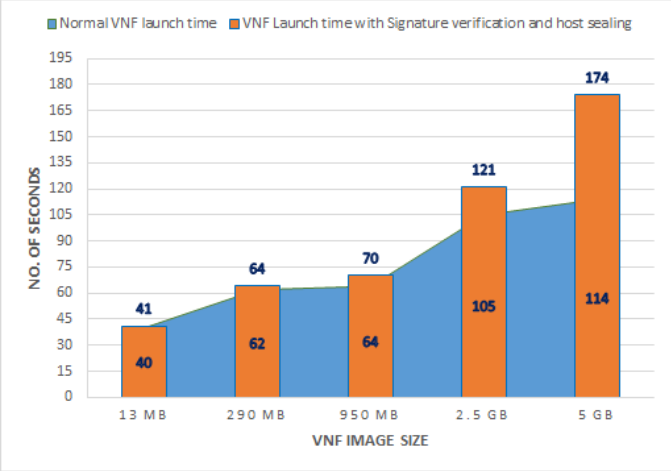


Fig. 7. Comparison of VNF normal launch time versus time with signature verification and host sealing

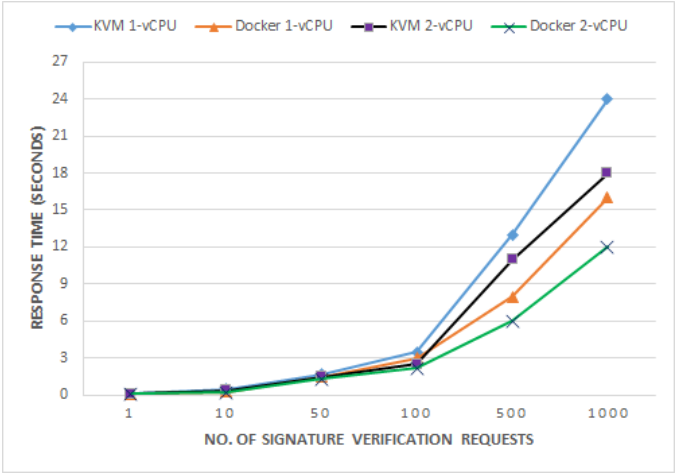


Fig. 9. SecO performance using KVM and Docker

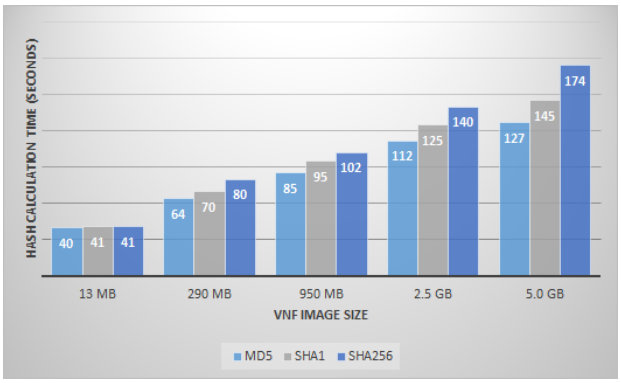


Fig. 8. Comparison of hash calculation time of VNF images

higher time overhead. It can be seen in figure 8 that the hash digest calculation time is almost the same for all three hashing functions for VNF images with size less than 950 MB. For VNF images exceeding 2.5 GB, the time gap seems to broaden. Nevertheless, we consider SHA256 hashing function as the best choice since the time overhead is not very significant as compared to other hashing functions.

In order to test the robustness of the signature verification and VNF-Host sealing method, we configured SecO on two different platforms: first in a virtual machine using KVM hypervisor and second in a Docker container. We tested the response time of SecO using two different configurations (i.e., with 1-vCPU and 2-vCPU). The performance of SecO is measured in terms of signature verification and sealing requests versus its response time to process them. Figure 9 depicts the response time of SecO. It becomes apparent that the time taken by SecO to respond to 100 simultaneous signature verification and sealing requests is mostly the same in all four configurations. It gets significant when 500 or more simultaneous requests arrive at SecO. SecO container running on Docker is the fastest among all in responding to the verification and sealing requests. That is thanks to better latency and computing efficiency of the application containers as compared to VMs.

V. Use Cases and Discussions

Using a VNF signing and verification method, VNF vendors can sign their VNF images and ship them to customers (e.g. mobile network operators). When a customer validates a VNF image's signature using vendor's public key, this provides a guarantee that images are integral and it also ensures the proof of ownership that VNF images come from the real vendor. VNF-Host sealing method can be useful in applications which require digital right management (DRM). A MNO can define custom policies that would enforce VNFs to start only on particular platforms and refuse to launch them if the platform is different.

The experimental work described in this paper has brought a number of challenges regarding the provisioning of trust. These challenges are discussed below:

1) Setting up a trusted cloud using the components discussed in this paper is very complex and highly technical process, specially when leveraging TPM module. TPM driver support varies in each OS and requires different tricks to get it working. In addition, TPM requires hardware and software provisioning by the help of scripts made by OEMs (original equipment manufacturers). This is compulsory step in order to use TPM with trusted boot.
2) In hybrid trusted cloud, whereby underlying infrastructure consists of both trusted and untrusted platforms (i.e. a platform with limited trust features), resource management is quite challenging. Indeed, it is possible that trusted resources are limited and the launch of mission-critical VNFs, required to run only on trusted platform, is denied. In order to solve this problem, extensive resource selection policies can be defined on trusted resources and related fault tolerance mechanisms can be developed.
3) A useful result, derived from this work, is the definition of run time trust in cloud, which is still very vague and ill-defined. There are many potential challenges

present with providing trust during cloud run time, especially challenges with measuring and verifying the critical system binary files when they are loaded into the memory. It should be noted that the concept of trusted compute pools [26] and trusted cloud at the moment only prevents from firmware root kit and boot kit attacks. Root/Boot kits are activated when the system is rebooted and they remain hidden since the OS security engines and anti-virus softwares are not activated at that level. Thus even after using the trusted boot technique, there is no guarantee that the cloud infrastructure would not be compromised during the run time. Also, using number of different components (such as SecO and remote attestation server) to build trusted cloud requires trust to be placed on each component. Securing all of these remote components is as critical as securing the cloud infrastructure itself. This is trade-off between setting up the security functions locally on cloud systems and remotely outside the cloud.

VI. Conclusion

In this paper, we discussed the need for VNF image integrity verification and VNF-Host sealing. In this vein, we devised mechanisms to address these problems. We implemented a signing and verification functionality in SecO (i.e., Security Orchestrator; external management entity) that can verify the integrity of VNF images during the launch time. This method detects if a VNF image is maliciously or accidentally tampered. It also proves the ownership of VNFs and protects them from internal attackers such as malicious administrators. We studied the Telco cloud requirements for launching VNFs on the specific hosts and accordingly presented adequate mechanisms to seal VNFs on certain hosts with given platform configurations.

Further, we evaluated the performance of our experiment setup using metrics such as response time and overhead time. In the performance evaluation of the experimental setup, we found that adding signature verification over the existing cloud infrastructure, such as OpenStack-based ones, does not add any major overhead time for VNF images of sizes smaller than 2 GB. The overhead time becomes significant only when VNF image size exceeds 2 GB. We also presented the SecO server's performance evaluation on both KVM hypervisor and Docker container. SecO running on Docker container appears to be faster in processing the verification and sealing requests.

Acknowledgments

This work was partially funded by Tekes, DIGILE Oy and the CyberTrust research programme. It was also partially supported by the ANASTACIA project, that has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N 731558 and from the Swiss State Secretariat for Education, Research and Innovation.

References

[1] T. Taleb, "Towards Carrier Cloud: Potential, Challenges, and Solutions," in IEEE Wireless Communications Magazine, Vol. 21, No. 3, Jun. 2014. pp. 80-91.

[2] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, and T. Magedanz, "EASE: EPC as a Service to Ease Mobile Core Network," in IEEE Network Magazine, Vol. 29, No. 2, Mar. 2015. pp.78-88.

[3] ETSI, GSNFV. "Network functions virtualisation (nfv): Architectural framework." ETSI GS NFV 2.2 (2013): V1.

[4] S. Lal, T. Taleb, and A. Dutta, "NFV: Security Threats and Best Practices" in IEEE Communications Magazine. (to appear)

[5] Researcher Finds Tor Exit Node Adding Malware to Binaries https://threatpost.com/researcher-finds-tor-exit-node-adding-malware-to-binaries/109008/ [accessed on: 10.02.2017]

[6] Wei, Jinpeng, et al. "Managing security of virtual machine images in a cloud environment." Proceedings of the 2009 ACM workshop on Cloud computing security. ACM, 2009.

[7] Rocha, Francisco, and Miguel Correia. "Lucy in the sky without diamonds: Stealing confidential data in the cloud." 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W). IEEE, 2011.

[8] Khan, Khaled M., and Qutaibah Malluhi. "Establishing trust in cloud computing." IT professional 12.5 (2010): 20-27.

[9] Sealing virtual machines in preparation for deployment as templates https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Virtualization/3.5/html/Administration_Guide/sect-Sealing_Virtual_Machines_in_Preparation_for_Deployment_as_Templates.html [accessed on: 10.02.2017]

[10] Yang, Wei, and Carol Fung. "A survey on security in network functions virtualization." NetSoft Conference and Workshops (NetSoft), 2016 IEEE. IEEE, 2016.

[11] Viega, John, Matt Messier, and Pravir Chandra. Network Security with OpenSSL: Cryptography for Secure Communications. "O'Reilly Media, Inc.", 2002.

[12] Nova Signature Verification https://specs.openstack.org/openstack/nova-specs/specs/mitaka/implemented/image-verification.html [accessed on: 10.02.2017]

[13] Virt driver pinning guest vCPUs to host pCPUs https://specs.openstack.org/openstack/nova-specs/specs/juno/approved/virt-driver-cpu-pinning.html [accessed on: 10.02.2017]

[14] Hoban, A., Czesnowicz, P., Mooney, S., Chapman, J., Shaula, I., Kinsella, R., and Buerger, C. A Path to Line-Rate-Capable NFV Deployments with Intel Architecture and the OpenStack Juno Release. Intel Corporation, March 2015.

[15] Jacquin, Ludovic, et al. "The trust problem in modern network infrastructures." Cyber Security and Privacy Forum. Springer International Publishing, 2015.

[16] Zhang, Qi, Lu Cheng, and Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges." Journal of internet services and applications 1.1 (2010): 7-18.

[17] https://sourceforge.net/projects/tboot/.

[18] Secure the Windows 8.1 boot process https://technet.microsoft.com/en-us/windows/dn168167.aspx [accessed on: 10.02.2017]

[19] Ruan, Anbang, and Andrew Martin. "Repcloud: achieving fine-grained cloud tcb attestation with reputation systems." Proceedings of the sixth ACM workshop on Scalable trusted computing. ACM, 2011.

[20] Open Cloud Integrity Tool https://github.com/opencit/opencit [accessed on: 10.02.2017]

[21] Filter Scheduler http://docs.openstack.org/developer/nova/filter_scheduler.html [accessed on: 10.02.2017]

[22] F.Z. Yousaf and T. Taleb, âĂIJFine Granular Resource-Aware Virtual Network Function Management for 5G Carrier Cloud,âĂİ in IEEE Network Magazine, Vol. 30, No. 2, Mar. 2016. pp. 110 âĂŞ 115.

[23] Dua, Rajdeep, A. Reddy Raja, and Dharmesh Kakadia. "Virtualization vs containerization to support paas." Cloud Engineering (IC2E), 2014 IEEE International Conference on. IEEE, 2014.

[24] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," IBM Research, Tech. Rep. RC25482, July 2014. [Online]. Available at: http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf [accessed on: 10.02.2017]

[25] Wang, Shiuh-Jeng, et al. "Concerns about Hash Cracking Aftereffect on Authentication Procedures in Applications of Cyberspace." IEEE Aerospace and Electronic Systems Magazine 22.1 (2007): 3-7.

[26] Security hardening http://docs.openstack.org/admin-guide/compute-security.html [accessed on: 10.02.2017]