Safeguarding VNF Credentials with Intel SGX

Nicolae Paladi
RISE SICS
nicolae.paladi@ri.se

Linus Karlsson
RISE SICS
linus.karlsson@ri.se

ACM Reference Format:

Nicolae Paladi and Linus Karlsson. 2017. Safeguarding VNF Credentials with Intel SGX. In *Proceedings of SIGCOMM Posters and Demos '17, Los Angeles, CA, USA, August 22–24, 2017*, 3 pages.
<https://doi.org/10.1145/3123878.3123930>

1 INTRODUCTION

Operators use containers – enabled by operating system (OS) level virtualization – to deploy virtual network functions (VNFs) that access the centralized network controller in software-defined networking (SDN) deployments. While SDN allows flexible network configuration, it also increases the attack surface on the network deployment [8]. For example, insecure communication channels may be tapped to extract or inject sensitive data transferred on the north-bound interface, between the network controller and VNFs; furthermore, to protect the network controller from malicious VNF instances, the integrity and authenticity of VNFs must be verified prior to deployment.

Scott et al. described in [8] threats to the security of VNFs, such as unauthorized access, data modification, data leakage, and malicious or compromised applications. Some of the enumerated threats are mitigated by protecting north-bound communication using standard network communication security protocols such as Transport Layer Security (TLS) with server- or mutual authentication. However, while the use of TLS prevents certain classes of attacks – e.g. topology spoofing, traffic eavesdropping – it shifts the focus to the protection of authentication credentials.

VNFs may contain exploitable vulnerabilities allowing attackers to obtain their authentication credentials. For VNFs deployed in containers, vulnerabilities in the OS isolation layer may render the container host – and *a fortiori* the neighboring containers – vulnerable to attacks on data integrity and confidentiality [4].

Integrity monitoring and integrity verification are used to detect the compromise of the OS virtualization layer and of VNFs deployed in containers. For example, commodity isolated execution environments have been used in earlier work to strengthen the security guarantees in SDN deployments. Kim et al. explored the design space for SGX-enabled software-defined inter-domain routing, peer-to-peer anonymity networks and middleboxes [6]; Shih et al. described an approach for protecting internal network function

virtualization states from tampering by a malicious datacenter administrator [9]; Coughlin et al. showed that performance-sensitive applications such as packet processing can be performed in isolated execution environments with an acceptable overhead [5]. However, ensuring and verifying the integrity of VNFs, as well as ensuring the confidentiality of VNF authentication credentials have not been addressed so far. We build upon previous work [7] to provide security guarantees regarding the integrity of VNFs deployed in containers prior to their deployment.

To mitigate the risks described above, we implemented a prototype that leverages hardware-based mechanisms for isolated execution implemented by Intel SGX in combination with a run-time integrity measurement subsystem, namely Linux Integrity Measurement Architecture (IMA)¹. This prototype is a first step towards providing to tenants and end-users integrity guarantees regarding the network components in SDN deployments.

2 ARCHITECTURE

A high-level architecture of an SDN deployment is illustrated in Figure 1. The figure also highlights additional security components introduced for integrity verification and monitoring.

Figure 1: High-level architecture of an SDN deployment with additional security components.

In our implementation we leverage the Representational State Transfer (REST) application programming interface (API) for communication between VNFs and network controller. We assume that the container host is equipped with a hardware-based isolated execution environment, such as AMD Secure Encrypted Virtualization [1], ARM TrustZone [3], SecureBlue++ [10], Intel Software Guard Extensions (SGX) [2], or similar. We introduce a *Verification Manager* module that has a central position in our proposed architecture: it obtains integrity measurements of VNFs through an attestation protocol and appraises the trustworthiness of the platform. Furthermore, it handles the communication with third-party

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions.acm.org.

SIGCOMM Posters and Demos '17, August 22–24, 2017, Los Angeles, CA, USA

© 2017 Association for Computing Machinery.

ACM ISBN 978-1-4503-5057-0/17/08...\$15.00

<https://doi.org/10.1145/3123878.3123930>

¹Linux IMA project page: <https://sourceforge.net/p/linux-ima/wiki/Home/>

attestation services, generates the HMAC key and nonces, as well as the certificates for the client authentication. To create trusted isolated execution environments (TEEs), we have chosen to use Intel SGX due to its popularity and availability on commodity platforms; however, other hardware-based isolated execution environment can be used to implement the described architecture. The Verification Manager communicates with special-purpose TEE *enclaves* to (1) extract integrity measurements of the software running on the container host to assess its trustworthiness and (2) provision or revoke authentication keys that can be used by VNFs as long as the container host is trustworthy.

We assume that the controller is connected via a secure channel to the container host and network applications running inside the containers (we implement this using TLS). The workflow (illustrated in Figure 1) starts with the Verification Manager ❶ which initiates the remote attestation of the container host. The Verification Manager contacts the *Intel Attestation Service* (IAS) – using the protocol provided by the SGX implementation² – to both verify the validity of the enclave key against the revocation list and the validity of the integrity quote ❷. We use the attestation functionality of SGX enclaves to measure the code loaded into the enclave prior to initialization (after that the enclaves becomes immutable [2]).

Upon a successful completion of the remote attestation protocol, the enclave sends to the Verification Manager a quote containing the integrity measurement list with measurements of the software on the container host. The integrity measurement list is produced by the Linux Integrity Measurement subsystem, which allows to collect measurements of certain files (the measurement targets are configured by the administrator in a policy file). The Verification Manager appraises the trustworthiness of the container host based on the obtained quote. The protocol continues only if the host is considered trustworthy following the appraisal.

After the attestation with the container host has been completed successfully, the Verification Manager starts the remote attestation of the VNF enclaves ❸ – e.g. VNF 1 and VNF 2 in Figure 1. Next, the Verification Manager interacts with the IAS ❹ to verify the validity of the integrity quote produced by the enclaves storing the VNF credentials, namely TEE 1 and TEE 2 in Figure 1.

Upon successful verification of the integrity quotes, the Verification Manager generates the certificate and private key and provisions them to the corresponding VNFs enclaves ❺. Once the authentication credentials are successfully provisioned, the VNFs can communicate with the network controller ❻. The credentials *do not* leave at any point the security context of the enclaves. Thus, to communicate with the network controller a VNF invokes its corresponding enclave, which then establishes a TLS session with the network controller. In our implementation, the security context established for each TLS session (including the session key) *does not* leave the enclave. An investigation of alternative implementations (and their performance impact) is left for future work.

3 USE CASES AND DEMONSTRATION

We implemented a prototype using *Ubuntu 16.04 LTS* with kernel version 4.4.0-51-generic for both the container host and network controller host. We used Docker version 1.12.2 to deploy VNFs

inside containers, which communicate with a *Floodlight* network controller version 1.2. Finally, we used *mbedtls-SGX*³ TLS protocol suite to implement a secure channel between the enclaves on the container host and the Verification Manager. Floodlight supports three different security modes for the REST API, non-secure (plain HTTP), HTTPS and trusted HTTPS (with client authentication). Floodlight performs client certificate validation by adding client certificates to its keystore, which introduces the challenge of maintaining the keystore updated with newly created keys. We solve this by provisioning the controller with a trusted certificate authority, rather than all client certificates. The Verification Manager acts as a certificate authority, and signs all newly created client certificates. The Floodlight controller must only validate that the client certificate has a valid signature from the trusted certificate authority.

The approach is based on two use cases. The first use case is the integrity attestation of a VNF. This is done by requesting a quote from the application attestation enclave (step ❸ to ❹ of Figure 1), that is then verified and matched against the expected values by the Verification Manager. This use case is demonstrated by the attestation protocol, communication with IAS, and matching the *actual* and *expected* measurements.

The second use case is enrolling the VNF into the SDN deployment. A prerequisite for this is that the VNF has been attested as above. The Verification Manager then generates a key and certificate, signs the certificate with its certificate authority, and next provisions the VNF's enclave with the key material. This corresponds to step ❺ of Figure 1. The provisioned key can then be used to establish a secure communication session with the SDN controller. This use case addresses key provisioning and ensures that entities without correct credentials cannot enroll in the SDN deployment.

4 FUTURE WORK

The integrity measurements of the container host are not currently protected by a hardware root of trust, such as a Trusted Platform Module (TPM). Integrity measurements are thus vulnerable to tampering by an adversary having root access to the container host. In future work we intend to implement a communication protocol to enable the integrity attestation enclave to retrieve authenticated integrity measurements from a TPM deployed on the platform.

5 ACKNOWLEDGEMENTS

This research has been performed within the 5G-ENSURE project (www.5Gensure.eu) and received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 671562.

REFERENCES

- [1] Advanced Micro Devices. 2016. *AMD64 Architecture Programmer's Manual Volume 2: System Programming*. Technical Report 24593-3.28. AMD. <http://support.amd.com/TechDocs/24593.pdf>
- [2] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. 2013. Innovative technology for CPU based attestation and sealing (*HASP*). ACM.
- [3] ARM. 2009. *Building a Secure System using TrustZone Technology*. Technical Report PRD29-GENC-009492C. ARM. <http://infocenter.arm.com/help/topic/>

²Intel SGX Software Development Kit page: <https://software.intel.com/en-us/sgx-sdk>

³mbedtls-SGX source: <https://github.com/bl4ck5un/mbedtls-SGX>

com.arm.doc.pr29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf

- [4] T. Combe, A. Martin, and R. Di Pietro. 2016. To Docker or Not to Docker: A Security Perspective. *IEEE Cloud Computing* (2016).
- [5] Michael Coughlin, Eric Keller, and Eric Wustrow. 2017. Trusted Click: Overcoming Security Issues of NFV in the Cloud (*SDN-NFVSec '17*). IEEE.
- [6] Seongmin Kim, Youjung Shin, Jaehyung Ha, Taesoo Kim, and Dongsu Han. 2015. A First Step Towards Leveraging Commodity Trusted Execution Environments for Network Applications (*HotNets-XIV*). IEEE.
- [7] Nicolae Paladi and Christian Gehrman. 2016. TruSDN: Bootstrapping Trust in Cloud Network Infrastructure (*SecureComm'16*). Springer.
- [8] Sandra Scott-Hayward, Sriram Natarajan, and Sakir Sezer. 2015. A survey of security in software defined networks. *IEEE Comm. Surveys & Tutorials* (2015).
- [9] Ming-Wei Shih, Mohan Kumar, Taesoo Kim, and Ada Gavrilovska. 2016. S-NFV: Securing NFV States by Using SGX (*SDN-NFV Security '16*). IEEE.
- [10] Peter Williams and Rick Boivie. 2011. CPU Support for Secure Executables (*TRUST '11*). Springer.