Formal Foundations for SCONE attestation and Intel SGX Data Center Attestation Primitives

Extended Abstract

Muhammad Usama Sardar
TU Dresden
Dresden, Germany
muhammad_usama.sardar@mailbox.tu-dresden.de

Christof Fetzer
TU Dresden
Dresden, Germany
christof.fetzer@tu-dresden.de

ABSTRACT

One of the essential features of confidential computing is the ability to attest to an application remotely. Remote attestation ensures that the right code is running in the correct environment. We need to ensure that all components that an adversary might use to impact the integrity, confidentiality, and consistency of an application are attested. Which components need to be attested is defined with the help of a policy. Verification of the policy is performed with the help of an attestation engine. Since remote attestation bootstraps the trust in remote applications, any vulnerability in the attestation mechanism can therefore impact the security of an application. Moreover, mistakes in the attestation policy can result in data, code, and secrets being vulnerable. Our work focuses on 1) how we can verify the attestation mechanisms and 2) how to verify the policy to ensure that data, code, and secrets are always protected.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**; **Logic and verification**.

KEYWORDS

Confidential Computing, Attestation

ACM Reference Format:
Muhammad Usama Sardar and Christof Fetzer. 2021. Formal Foundations for SCONE attestation and Intel SGX Data Center Attestation Primitives : Extended Abstract. In *Proceedings of ACM PaveTrust Workshop (PaveTrust'21).* ACM, New York, NY, USA, 3 pages.

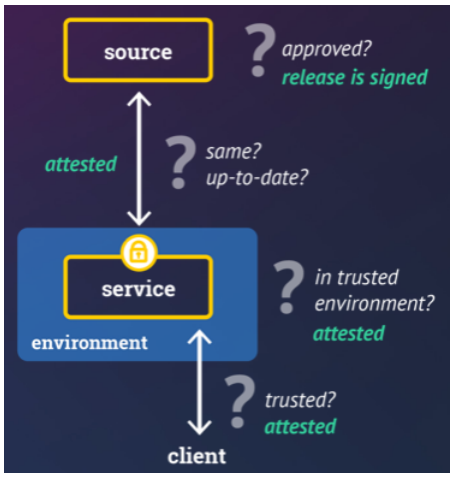

Figure 1: To ensure that integrity, confidentiality and consistency in hostile environments, we need establish trust into various components using attestation.

1 INTRODUCTION

Traditionally, we establish trust in an application bottom up. We ensure that the computers executing the application are hosted in a data center with sufficient physical security. The hypervisor and the operating system are maintained correctly by trusted operation staff. These days, we want to host applications in environments that the application owner does not control. The application owner wants to outsource the hardware management, hypervisor, operating systems, and container infrastructure to external entities like a cloud provider. The application owner might even want to outsource the management of the application to an external provider. The provider might be under different jurisdictions, and hence, we might need to ensure that the provider can read or modify the data, code, and secrets of the application.

One of the leading technical difficulties is that a hypervisor or an operating system can read an application's memory. In other words, it is quite a challenge to protect an application from a provider in control of the hypervisor or the operating system or the application that wants to access an application. Confidential computing enables application owners to protect applications and establish trust in their applications top-down.

An application owner outsources the computing infrastructure management to a provider that she cannot legally trust. Hence, we need to consider a new threat model: the adversary might have root access to all layers of our computing infrastructure, i.e., management access to the servers and all system software.

Confidential computing protects applications by permitting an application to run inside of encrypted memory regions. We call such a region an enclave. Only the application code running inside of the enclave can see the code, data, and secrets stored in this encrypted memory region in the clear. All other software and hardware can only see encrypted data. Any modification of this region by other code or hardware will either be prevented or at least detected.

PaveTrust'21, December 2021, Online

When creating an enclave, we have a problem on how to bootstrap trust in the enclave. A loader creates this enclave. How can we establish trust in the loader or establish trust in the loaded enclave? Some systems use a trusted loader with a known public key: each computer has a unique public key. The application owner could encrypt the content of an enclave with the help of this public key. The loader can decrypt the content inside its enclave, create a new enclave, and transfer it via an encrypted channel.

A trusted loader approach raises some technical challenges. First, we would need to provide an encryption key for all trusted loaders. This requirement would limit the use of generic container images. Moreover, any loader executing on a server affected by a security advisory could compromise the encrypted content.

Our general approach is to establish trust in the loaded enclave instead. An untrusted loader creates the enclave. We establish trust in the created enclave with the help of attestation. Attestation is a testimony as to the truth of a matter. In the context of confidential computing, we use attestation to establish trust in an application started inside of an enclave.

With the help of attestation, we want to ensure the integrity, confidentiality, and consistency of an application's data, code, and secrets. In other words, we want to ensure that we can trust the output of an application (integrity and consistency) and that the data was not leaked (confidentiality). Since our threat model assumes that adversaries might have root access, we need to verify various conditions (see Fig 1):

- the code and data initially loaded in the enclave is up-to-date and not modified,
- the enclave is executed on a CPU with up-to-date hardware and firmware,
- the state of the container image or the volumes is correct has not been modified,
- the enclave runs on a specific set of hosts only (optionally),
- the enclave is not cloned (optionally)

In the future, we expect that we will attest to additional components. For example, that the operating system and the hypervisor are correctly booted.

2 ATTESTATION AND VERIFICATION

An attestor signs a report as part of the attestation. A report associates a measurement with some other data. For example, the measurement could be a secure hash value of the initial enclave state. Additionally, the report might contain the public key of an enclave. An application owner creates a policy that defines the expected measurements. The policy defines the initial state of the enclave. A policy could also require the state of the operating system, i.e., it could limit what operating systems can be loaded and what applications can be executed [3]. The policy can also specify a set of hosts on which the enclave can run. It can define how many of the instances of the enclave can run at any point in time.

SCONE [1] is a platform that enables the transformation of native services into confidential services. The policy can be generated automatically as part of transforming a native application into a confidential computation. A confidential service is automatically attested and verified by SCONE. Only if a service satisfies its security policy, it receives its secrets (see Fig 2).

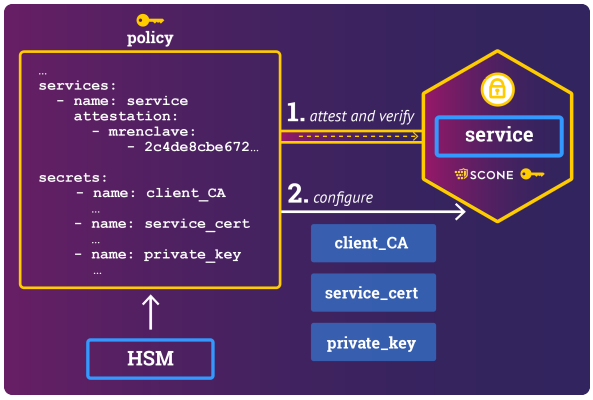

Figure 2: Services are automatically attested during startup. After verification against the service policy, a service gets access to its secrets.

For any complex software and hardware system, we have to expect security vulnerabilities. Attestation enables us to deal with security vulnerabilities effectively. If there is any known security vulnerability on a certain host, we need to mitigate these before our application is permitted to execute. For example, we have experienced that in some Kubernetes clusters, not all servers were running the newest firmware. The confidential applications cannot run on these servers since they do not pass the verification. Kubernetes will detect that the startup failed and will eventually restart the service on a different node.

For Intel SGX, a large variety of security vulnerabilities have been published, e.g., [2, 6, 8]. Some of the vulnerabilities, like side-channel attacks, are outside of the original threat model of Intel SGX and need to be mitigated in different ways. For all known vulnerabilities of Intel SGX, Intel issues security advisories and issues firmware patches. However, in some cases, like Load-Value Injection [7] only new CPU hardware can efficiently fix the issue. We verify during attestation of an enclave if the underlying platform is affected by a security advisory. If software mitigations are in place, we can specify in its policy that we can tolerate this security advisory.

There are various approaches to mitigate side-channel attacks. We can mitigate these attacks using oblivious memory and/or fine-grain randomization. However, this requires a recompilation of the binaries. Moreover, oblivious memory introduces a very high runtime overhead. We support an attestation-based approach [3] with minimal runtime overhead. The idea is that we perform a load time and runtime attestation of the operating system. Any side-channel attack requires the execution of a program to perform this attack. Such an attack would be detected using runtime attestation. Only known benign programs are permitted to be started. The runtime attestation will prevent the start of any unknown program. Also, this will result in the stop of the enclaves running on this host.

A modern cloud-native application consists of a set of services. These services communicate via the network with each other. One can ensure that a service is part of the same application with the help of implicit attestation. Each service is provisioned with service

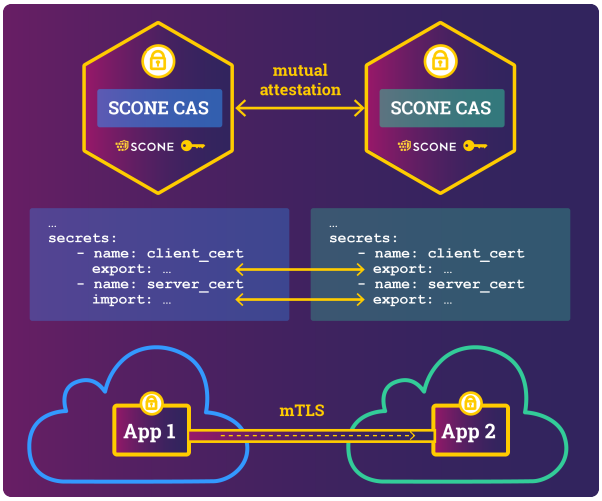

Figure 3: Implicit attestation of communication partners using mTLS.

and client certificates as part of the attestation process. The client and service certificates are issued by a certificate authority that is unique for each application instance (see Fig 3). This ensures that only services belonging to the same application instance can communicate with each other via mTLS (mutual TLS).

3 FORMAL METHODS

We investigate the use of formal methods to verify the attestation mechanism as well as the policies.

SCONE supports both Intel Attestation Service as well as Intel DCAP [5]. The Intel Data Center Attestation Primitives (DCAP) is a third-party attestation service to enable data centers to create their own attestation infrastructures. These services address the availability concerns and improve the performance compared to the remote attestation based on Enhanced Privacy ID (EPID).

Since remote attestation bootstraps the trust in remote applications, any vulnerability in the attestation mechanism can therefore impact the security of an application. The lack of formal proof for DCAP might cause security concerns. To fill this gap, we propose an automated, rigorous, and sound formal approach to specify and verify the remote attestation based on Intel SGX DCAP under the assumption that there are no side-channel attacks and no vulnerabilities inside the enclave [4]. In this approach, the data center configuration and operational policies are specified to generate the symbolic model, and security goals are specified as security properties to produce verification results. The evaluation of non-Quoting Verification Enclave-based DCAP indicates that the confidentiality of secrets and data integrity is preserved against a Dolev-Yao adversary in this technology.

An application consists of a set of services. Each service is associated with a policy that defines how to attest and verify the service and which secrets we want to provision to the service. Having a large number of policies, it can become quite complex to ensure that all components of an application are indeed sufficiently protected. Our research focuses on how to verify automatically that the policies of an application protect all code, data, and secrets:

- Can only services - except the ingress service - of the application communicate only with each other?
- Data at rest is always protected and only accessible by the services of the application?
- Administrators of the service cannot see any secrets nor any data?
- Any change of a policy requires approval by the application owner?
- Can we perform secure software updates?

Our general approach is to be able to answer these questions automatically using formal methods.

4 CONCLUSION

Many security attacks exploit known vulnerabilities. Attestation enables the enforcement of advanced cyber-hygiene. We can ensure that only up-to-date application code runs inside enclaves on top of up-to-date firmware and hardware that mitigates all known security advisories. Moreover, we can verify with the help of attestation that services execute only on known hosts with an up-to-date operating system and a limited number of applications.

Attestation is an essential component to ensure the security of applications. Hence, we investigate the use of formal methods 1) to ensure the correctness of the attestation mechanisms and 2) to verify that the policies of an application sufficiently protect the services, the data, the code, and the secrets of an application.

REFERENCES

[1] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Dan O'keeffe, Mark L Stillwell, et al. 2016. {SCONE}: Secure linux containers with intel {SGX}. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 689–703.

[2] Kit Murdock, David Oswald, Flavio D Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. 2020. Plundervolt: Software-based fault injection attacks against Intel SGX. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1466–1482.

[3] Wojciech Ozga, Do Le Quoc, and Christof Fetzer. 2020. A practical approach for updating an integrity-enforced operating system. In *Proceedings of the 21st International Middleware Conference*. 311–325.

[4] Muhammad Usama Sardar, Rasha Faqeh, and Christof Fetzer. 2020. Formal Foundations for Intel SGX Data Center Attestation Primitives. In *International Conference on Formal Engineering Methods*. Springer, 268–283.

[5] Vinnie Scarlata, Simon Johnson, James Beaney, and Piotr Zmijewski. 2018. Supporting third party attestation for Intel SGX with Intel data center attestation primitives. *White paper* (2018).

[6] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 991–1008.

[7] Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lippi, Marina Minkin, Daniel Genkin, Yuval Yarom, Berk Sunar, Daniel Gruss, and Frank Piessens. 2020. LVI: Hijacking transient execution through microarchitectural load value injection. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 54–72.

[8] Nico Weichbrodt, Anil Kurmus, Peter Pietzuch, and Rüdiger Kapitza. 2016. AsyncShock: Exploiting synchronisation bugs in Intel SGX enclaves. In *European Symposium on Research in Computer Security*. Springer, 440–457.