SGX Project

Team Awesome

July 29, 2016

Contents

1 Introduction

With the recent surge in privacy concerns, employing SSL/TLS to secure communications in the middle of the network has become common place. SSL/TLS offers guarantees of confidentiality and integrity provided that a private key's secrecy is maintained. Yet, SSL/TLS was designed assuming that its user trusts the hardware and OS of the machine on which the key is held.

While this assumption is perfectly valid in the case where a person is running an SSL/TLS enabled service on their own machines, many web applications are now hosted by third party cloud service providers such as Amazon Web Services, Heroku, Digital Ocean &c. Moreover, to offer SSL/TLS, the private key must also be stored with the web application on these service providers' machines. This implies that a server administrator using the aforementioned services is trusting the cloud-provider, including any personnel with physical/administrative access to the machines, and the underlying OS to maintain the secrecy of the sensitive key material. Such a wide trust surface makes it difficult to maintain the privacy of critical secrets.

Consider a case where the cloud provider is not malicious; a vulnerability within their platform could lead to leaking the private key if exploited by an adversary. Moreover, if the cloud provider is indeed malicious they could simply read your private key from the hard disk if stored unencrypted, or mount some form of memory sniffing attack to read the key from the web server's memory since data in RAM is unencrypted. A compromised private key allows an adversary to do the following:

- Decrypt past, stored, communication between the web server and a client (assuming a cipher that does not provide perfect forward secrecy is in use)
- Decrypt any ongoing communication between the web server and a client
- Masquerade as the server and fool a client into disclosing sensitive information such as passwords

In all cases, a compromised key voids the confidentially and integrity guarantees of SSL/TLS.

2 Background

2.1 Provisioning the enclave with the long-term private key

Provisioning a web server with a long-term private key for the purposes of SSL/TLS is currently done storing the private-key along with the executable on the remote machine. This scheme, however, assumes a trusted cloud-provider/OS. However, under our threat model this scheme is not viable. Alternatively, we require a method by which we can verify the identity and integrity of the server application and then, upon successful verification, we can send the long term private key to the server in a secure fashion. Loosely, the requirements are as follows:

- The mechanism allows the verification of the identity and integrity of the server application and the underlying TCB. This is so that we can be sure the private-key is being sent to the same server we placed on the remote machine, and the software is being executed by trustworthy hardware.
- The mechanism allows us to setup a secure channel, ensuring that the only entities privy to the private key are the server application and the server administrator.

The first and second requirement are met by a process called inter-platform attestation.

2.2 Inter-platform attestation

Inter-platform attestation is a mechanism that can be invoked by an entity, referred to as the challenger, running on one platform to verify an enclave running on another, remote, platform. This process is enables the challenger to verify the following about the remote enclave:

1. The contents of the enclave's pages (code, data, stack and heap) upon creation (after the ECREATE instruction completes)
2. The identity of the entity that signed the enclave
3. The trustworthiness of the underlying hardware
4. Authenticity and integrity of any data generated by the enclave and sent as part of the attestation process. This allows us to satisfy the second requirement by generating an ephemeral key pair and binding it to the remote attestation process. This, therefore, allows the challenger to verify the integrity of the ephemeral public key and verify that it was generated by the server application.

The steps involved in the attestation process are as follows (illustrated in Figure 1):

1. The challenger invokes the remote attestation mechanism to verify the identity and integrity of the remote enclave

2. The non-trusted part of the web server receives the challenge, passes it along to the trusted portion of the web server along with the identity of the quoting enclave. The quoting enclave is a special enclave provided by Intel as part of the SGX platform to enable remote attestation by verifying the integrity of the underlying hardware.

3. The enclave invokes EREPORT which is an SGX instruction that generates a REPORT structure to be provided to a *local* enclave, the quoting enclave in this case. This structure contains a hash of the contents of the enclave's pages upon ECREATE's termination, a hash of the identity of the enclave's signer, a hash of any user-data, the ephemeral key in our case, generated by the enclave. The REPORT is signed by a MAC-key that can only be accessed by the CPU and the quoting enclave. The REPORT along with the ephemeral key is then sent to the non-trusted part of the application.

4. The REPORT is sent to the quoting enclave where its integrity is verified by calculating the MAC across its contents.

5. Assuming the REPORT is verified successfully, the quoting enclave generates a QUOTE structure that includes the REPORT structure and a signature across the quote generated using a key known as the EPID key. The EPID key is a private key unique to the CPU that is part of the platform and verifies the firmware of the processor and its SGX capabilities.

6. The QUOTE is sent along with the ephemeral key to the challenger

7. The challenger verifies the QUOTE structure by using an EPID public certificate. If this is successful then the challenger is sure that this QUOTE came from a valid SGX CPU and can trust its authenticity. The challenger can then check the contents of the REPORT contained within the QUOTE to verify the identity of the remote enclave, and the integrity of the ephemeral key received along with the QUOTE. The ephemeral key, if proven to be valid, can now be used to communicate with the remote enclave in a secure manner.

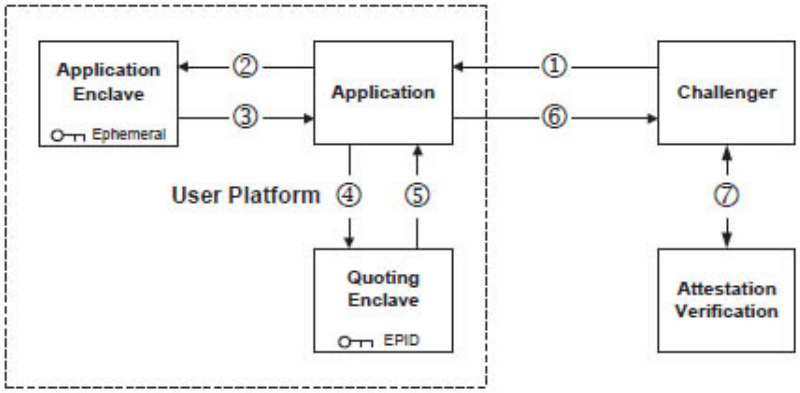

Figure 1: Remote Attestation and Secret Provisioning

s

3 Design

Hello world!

Hello, here is some text without a meaning. This...

4 Project Management

Hello world!

Hello, here is some text without a meaning. This...

5 Conclusion

Hello world!

Hello, here is some text without a meaning. This...