Anarchy by Design
Preventing invisible RNG attacks on TLS using SGX

Jeremy Erickson *Timothy Trippel*
{jericks, trippel}@umich.edu
EECS 582, Fall 2016
University of Michigan, Ann Arbor

1 Introduction

In the computer and network security domain, nation state actors (NSAs), are typically characterized as having a significant amount of computing resources, intelligent researchers and engineers, and legal authority that private sector organizations do not possess. With this kind of power, NSAs have many documented cases of using their strategic advantage to subvert modern computing and communication systems to gather intelligence [2, 4, 5].

In previous work [1], we investigated how an NSA could leverage its resources to gain widespread access to encrypted communications while avoiding public criticism through secrecy and stealth. Specifically, we focused on a threat model in which the NSA could, through exploitation, coercion, or another method, acquire superuser privileges on some fraction of the host machines of a cloud services provider.

Despite its power, we make one critical assumption about the NSA that we believe is reasonable given the existing political and legal climate:

> An NSA must perform offensive actions in a manner of utmost stealth. Detection of offensive actions by any non-NSA personnel will lead to a full investigation and removal of superuser privileges on the cloud provider, as well as damage to public opinion.

Under this constraint, we propose that an NSA may decide to focus on subverting the Random Number Generator (RNG) of virtual machines through control of the hypervisor. This has several attractive qualities. First, it is possible to achieve control of the RNG without having to modify the virtual machine itself, meaning a cloud tenant, even one that inspects checksums of critical system components such as the kernel, is unable to detect any subversion without specifically searching for artifacts of the used technique. Second, with control over the RNG, cryptographic keys become predictable, and so there is no need to exfiltrate encryption keys out of the cloud infrastructure. Encrypted communications can be monitored from outside the cloud infrastructure with no extraneous, suspicious key leakage shadowing each new encrypted message. Third, we can control the RNG with full precision, so we may generate the output random numbers using a previously-defined shared secret. Thus, the output appears random to all observers, but we can predict it and therefore decrypt any encrypted communications.

```
# dd if=/dev/urandom count=1 bs=10 2>/dev/null | xxd
 362b 3f69 cdb8 fce9 64f1                6+?i....d.
# dd if=/dev/urandom count=1 bs=10 2>/dev/null | xxd
 6666 6666 6666 6666 6666                ffffffffff
```

Figure 1: Generating random bytes first without, and then with, the hypervisor attack. A static value of all `0x66` bytes is chosen for visibility.

In our previous work, we were able to demonstrate consistent, practical attacks against both the Linux kernel RNG and the OpenSSL library RNG used by the Apache2 web server. In this project, we aim to build a robust defense against this threat using Intel's new Software Guard Extensions (SGX) [3].

2 Background

Intel SGX is a platform in which applications may ask the processor to reserve a special section of encrypted memory and special execution space for secure functionality. In this paradigm, the enclave application cannot be tampered with by the operating system or hypervisor, and its running code can be *remotely attested*.

As shown in Figure 2, the unprivileged portion of the application must pass its instructions through the OS and hypervisor layers. However, once the enclave is running in privileged memory and verified, the unprivileged ap-

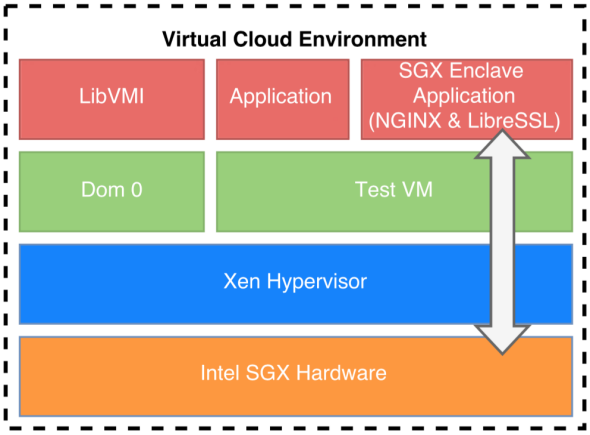

Figure 2: Diagram of a virtualized environment on single host machine using Intel SGX hardware, the `Xen` hypervisor, and `libvmi` inspection tool. Portions of security-critical applications, i.e. NGINX and LibreSSL, can execute in a secure SGX enclave to prevent an adversary with access to the hypervisor, i.e. through `libvmi`, or OS from subverting security critical tasks, i.e. random number generation. We note that Xen does not currently support SGX, but we expect it to soon.

plication may only interact with it through a defined and approved API, limiting the unprivileged application or another application from tampering with it.

3 Approach

We aim to demonstrate how developers can provide security guarantees at the application layer, using Intel's SGX technology, for HTTPS servers that prevent RNG subversion attacks resulting from a compromised OS or hypervisor. Transport Layer Security (TLS) is a protocol that provides an authenticated and confidential network channel of communication. To do so it relies on implementing cryptographic algorithms to generate secret keys. The generation of secret cryptographic keys in turn relies on the successful generation of random numbers.

Random number generators, included in many cryptographic software libraries, gather sources of entropy to feed a deterministic psuedo-random number generator. With knowledge of the entropy pool and pseudo-random number generation algorithm, all random numbers become easily predictable. If the random number generator used by a TLS server running on VM in a virtual cloud environment is compromised by the hypervisor or OS, all TLS connections emanating from it are not secure. We note that despite the OS providing randomness directly from the kernel's RNG, applications using the OpenSSL and LibreSSL libraries will maintain their own entropy pool from which to draw randomness.

We plan to demonstrate how this threat can be mitigated by embedding the entropy pool and random number generator components of the LibreSSL cryptographic software library in an SGX enclave. This involves substantial modification to the flow of execution in the NGINX and LibreSSL applications, as they will need to accommodate the structural overhead enforced by SGX.

4 Scope

We are attempting to build a working implementation of NGINX with support for our enclave. To do this, we will need to support the creation of the web server's long-term private key and the corresponding Certificate Signing Request, and keep this private key in encrypted memory. Our enclave will also need to support the web server operations of generating random numbers for ephemeral session keys and signing TLS Server Exchange messages with the long-term private key after verifying their integrity. Depending on whether or not we wish to prevent the untrusted code from ever coming in contact with the ephemeral keys, we may additionally need to provide encryption and decryption routines for multiple simultaneous flows.

Evaluation of this system's effectiveness will be challenging, as Xen does not currently pass hardware support of SGX to it's virtual machines (nor does KVM). This precludes our ability to test our existing attack on the new system. However, we do plan to evaluate the performance overhead of this approach, as web servers are often heavily loaded and a severe degradation of performance when creating new connections would be a notable downside.

References

[1] Jeremy Erickson, Timothy Trippel, and Andrew Quinn. *Cloaking Order in Chaos: Subverting the random number generator via the hypervisor*. 2016. URL: `https://jeremy-erickson.com/static_docs/EECS588/paper.pdf`.

[2] April Glaser. *After NSA Backdoors, Security Experts Leave RSA for a Conference They Can Trust*. Jan. 30, 2014. URL: `https://www.eff.org/deeplinks/2014/01/after-nsa-backdoors-security-experts-leave-rsa-conference-they-can-trust`.

[3] Ittai Anati et al. *Innovative Technology for CPU Based Attestation and Sealing*. Tech. rep. Intel, 2013. URL: https://software.intel.com/en-us/articles/innovative-technology-for-cpu-based-attestation-and-sealing.

[4] Olga Khazan. *The Creepy, Long-Standing Practice of Undersea Cable Tapping*. July 13, 2013. URL: http://www.theatlantic.com/international/archive/2013/07/the-creepy-long-standing-practice-of-undersea-cable-tapping/277855/.

[5] Mandient. *APT1. Exposing One of China's Cyber Espionage Units*. Feb. 19, 2013. URL: http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf.