Cloaking Order in Chaos

Invisibly subverting the Linux random number generator via the hypervisor

Jeremy Erickson (jericks) *Andrew Quinn (arquinn)* *Timothy Trippel (trippel)*
EECS 588, Winter 2016
University of Michigan, Ann Arbor

1 Problem

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[3][4][5].

In this project we propose to investigate how an NSA could leverage its resources to gain widespread access to encrypted communications while avoiding public criticism through secrecy and stealth. Specifically, we focus on a hypothetical scenario in which the NSA has, through exploitation, coercion, or another method, acquired 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:

> The NSA must act in a manner of utmost stealth. Detection of any actions by any non-NSA actor 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 likely 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, should be 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, if the NSA can control the RNG with high enough precision, it should be possible to cause the output random numbers to still appear truly random, and only predict them with knowledge of some NSA-controlled secret, thus maintaining the tenant's security against all non-NSA actors.

2 Context

Alt et al. did prior work in this area for a course project [1]. While their project looked at the ability of a hypervisor to control a guest OS's random number generator, they did not approach the problem from the standpoint of a NSA or focus on detection of such methods. The main difference between our proposed work and their prior work is our focus on stealth. Alt et al. demonstrated that the random number generation of a guest OS can be controlled by a hypervisor; we will extend this to identify whether this control can occur in a reasonably stealthy manner, or not, and what limitations to this approach exist. Further, from investigative work we have already done, we have identified one additional hurdle to accomplishing this task in a stealthy manner without guest OS cooperation. To monitor and modify the runtime memory of the virtual machine, we must be able to determine the precise memory location of a variety of data structures within the guest OS. The canonical way of accomplishing this task involves using a kernel module from within the guest to pull this information from runtime symbols and manually import it back to the introspection application. However, this approach will not work in our model, which cannot rely on guest cooperation.

Other work in the area of random number generation in virtual environments [2] has primarily focused on the unintentional side-effects of low boot-time entropy

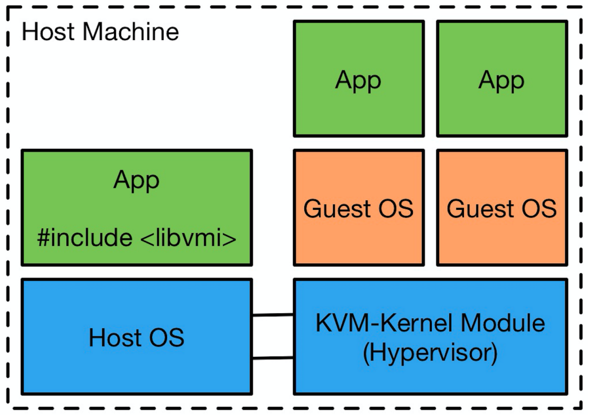

Figure 1: Diagram of virtualized environment on host machine using the `KVM` hypervisor and `libvmi` inspection tool

across virtual machines that are replicated from a common image. This is only slightly relevant to our proposal, as changes made to the Linux RNG to combat this low boot-time entropy need to be accounted for in any attack against the RNG, but can likely be controlled in the same manner as other sources of entropy can.

3 Approach

There are a number of ways that the state-level actors could alter the behavior of a hypervisor to control the random numbers that are generated by a particular guest OS. To control a guest OS's random number generator, a hypervisor has to either control the inputs to the entropy pool of that guest OS, which is later used to derive random numbers, or separately generate and replace the returned random numbers. Orthogonally, if the guest OS requests a random number via the `rdrand` instruction, the NSA could emulate the result of the instruction and return a controlled number.

Alt et al. showed that a hypervisor can intercept accesses to random data[1]. We first plan to reproduce their initial results, with a focus on the `KVM` hypervisor, and then plan to investigate whether a guest OS can detect that this entropy tampering is occurring. We note that even though typically defenses that detect a specific attack often turn into cat-and-mouse games between the attacker and defender, under our initial assumption, the detection of the NSA even a single time is sufficient to cause a significant disruption to the NSA's ability to maintain their capability.

We also plan on investigating the ability of a hypervisor to control the sources of entropy into an entropy pool. Alt et al. suggest that this approach may be infeasible, however we hope to be able to refute this claim. Since the APIs that feed entropy into the pool are well defined and modern Virtual Machine Introspection (VMI) tools allow us to introspect arbitrary memory addresses in a guest OS, we should be able to control all input into the entropy pool. However there are a number of data races within the Linux kernel's entropy pool logic which add non-determinism to the pool itself. Our current idea is to provide synchronization to the entropy pool through the VMI itself, thereby preventing these non-deterministic changes to the entropy pool.

To implement these attacks we plan to use the `libvmi` virtual machine introspection library to interact with the `KVM` hypervisor and control the guest VM. `KVM` runs as a module within the Linux kernel and emulates the hardware for guest OSes. `libvmi` is a cross-platform VMI library which allows low level introspection into the behavior of a guest OS. The system architecture is shown in Figure 1.

4 Evaluation

Unfortunately, since our proposal is somewhat exploratory, the only metric we can be absolutely certain to be able to use is whether we are able to generate deterministic output from the Linux Random Number Generator, or not. The rest of our approach involves investigating the side-effects required of any implementation that does so, and to what extent these can be detected. One ultimate validation of this approach would be if we could detect such an attack in the wild, but since this attack is currently hypothetical, the absence of detection in the real world should not be construed as failure.

5 Scope

In the pursuit of building a working prototype, we have several options we can pursue. We plan to investigate both approaches (intercepting RNG accesses and control of sources of entropy), and implement at least one working attack by the checkpoint date. We then plan to use the remaining time to explore the creation of a kernel module that can reliably detect our attack.

Outside the scope of this course project, we plan to continue this work, explore both attack vectors and appropriate detection methods for each, and deploy our detection mechanisms across prominent cloud service providers to attempt to detect such an attack in the wild.

We also plan to open-source our work so that the rest of the community can benefit and build on top of these techniques without needing to reimplement them.

References

[1] Matthew Alt et al. *Entropy Poisoning from the Hypervisor*. Unpublished class project. 2015. URL: `https://courses.csail.mit.edu/6.857/2016/files/alt-barto-fasano-king.pdf`.

[2] Adam Everspaugh et al. “Not-So-Random Numbers in Virtualized Linux and the Whirlwind RNG”. In: *Proceedings of the 35th IEEE Symposium on Security and Privacy*. 2014. URL: `http://pages.cs.wisc.edu/~ace/papers/not-so-random.pdf`.

[3] 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`.

[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`.