Linux Plumbers Conference 2019

Contribution ID: **58** Type: **not specified**

Kernel Address Space Isolation

Tuesday, 10 September 2019 12:45 (45 minutes)

Recent vulnerabilities like L1 Terminal Fault (L1TF) and Microarchitectural Data Sampling (MDS) have shown that the cpu hyper-threading architecture is very prone to leaking data with speculative execution attacks.

Address space separation is a proven technology to prevent side channel vulnerabilities when speculative execution attacks are used. It has, in particular, been successfully used to fix the Meltdown vulnerability with the implementation of Kernel Page Table Isolation (KPTI).

Kernel Address Space Isolation aims to use address spaces to isolate some parts of the kernel to prevent leaking sensitive data under speculative execution attacks.

A particularly good example is KVM. When running KVM, a guest VM can use speculative execution attacks to leak data from the sibling hyper-thread, thus potentially accessing data from the host kernel, from the hypervisor or from another VM, as soon as they run on the same hyper-thread.

If KVM can be run in an address space containing no sensitive data, and separated from the full kernel address space, then KVM would be immune from leaking secrets no matter on which cpu it is running, and no matter what is running on the sibling hyper-threads.

A first proposal to implement KVM Address Space Isolation has recently been submitted and got some good feedback and discussions:

https://lkml.org/lkml/2019/5/13/515

This presentation would show progress and challenges faced while implementing KVM Address Space Isolation. It also looks forward to discuss the possibility to have a more generic kernel address space isolation framework (not limited to KVM), and how it can be interfaced with the current memory management subsystem in particular.

MERGED with:

Address space isolation has been used to protect the kernel from the
userspace and userspace programs from each other since the invention of
the virtual memory.

Assuming that kernel bugs and therefore vulnerabilities are inevitable
it might be worth isolating parts of the kernel to minimize damage
that these vulnerabilities can cause.

Recently we've implemented a proof-of-concept for "system call
isolation (SCI)" mechanism that allows running a system call with
significantly reduced page tables. In our model, the accesses to a
significant part of the kernel memory generate page faults, thus
giving the "core kernel" an opportunity to inspect the access and
refuse it on a pre-defined policy.

Our first target for the system call isolation was an attempt to
prevent ROP gadget execution [1], and despite its weakness it makes a
ROP attack harder to execute and as a nice side effect SCI can be used
as Spectre mitigation.

Another topic of interest is a marriage between namespaces and address spaces. For instance, the kernel objects that belong to a particular network namespace can be considered as private data and they should not be mapped in other network namespaces.

This data separation greatly reduces the ability of a tenant in one namespace to exfiltrate data from a tenant in a different namespace via a kernel exploit because the data is no longer mapped in the global shared kernel address space.

We believe it would be helpful to discuss the general idea of address space isolation inside the kernel, both from the technical aspect of how it can be achieved simply and efficiently and from the isolation aspect of what actual security guarantees it usefully provides.

[1] https://lore.kernel.org/lkml/1556228754-12996-1-git-send-email-rppt@linux.ibm.com/

I agree to abide by the anti-harassment policy

Yes

Primary author: CHARTRE, Alexandre (Oracle)

Presenters: CHARTRE, Alexandre (Oracle); BOTTOMLEY, James (IBM); RAPOPORT, Mike (IBM); NIDER, Joel (IBM Research)

Session Classification: LPC Refereed Track