Page Cache Side Channel Attacks

Pankaj Gupta
Software Developer(Kernel & Virtualization), Red Hat
Kernel meetup, 28th Apr, 2019

Agenda

What is Cache?

Introduction to memory mapping.

Introduction to Side Channel attacks.

Page Cache Side Channel attacks.

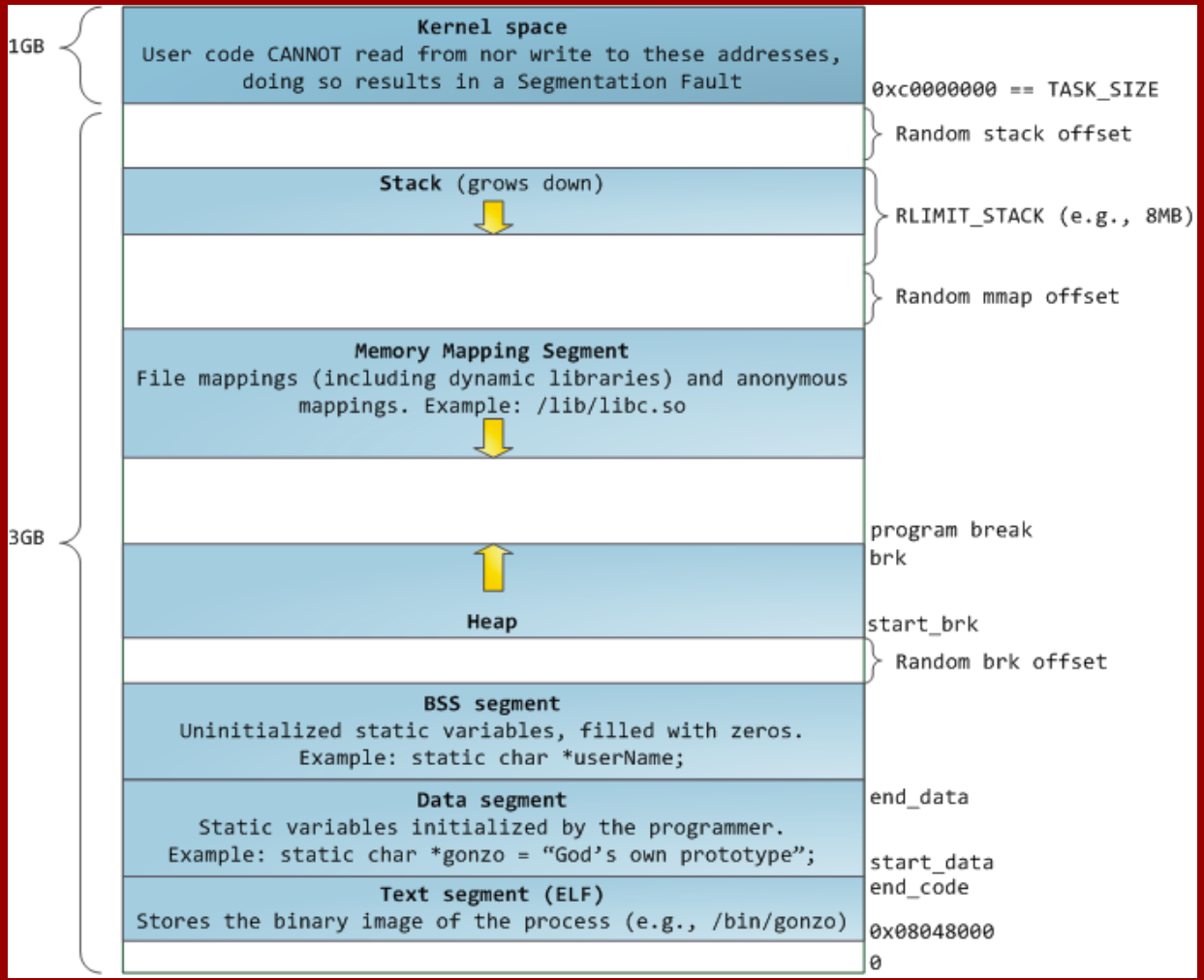
Possible Mitigations.

Cache?

- Stores data for serving future requests faster
- CPU Caches
- TLB (Translation lookaside buffer)
- Page Cache

Memory region

- Reserve linear(virtual) address range.
- vm_area_struct
- /proc/{pid}/maps


1GB
3GB
Kernel space
User code CANNOT read from nor write to these addresses, doing so results in a Segmentation Fault
0xc0000000 == TASK_SIZE
Random stack offset
Stack (grows down)
RLIMIT_STACK (e.g., 8MB)
Random mmap offset
Memory Mapping Segment
File mappings (including dynamic libraries) and anonymous mappings. Example: /lib/libc.so
program break
brk
Heap
start_brk
Random brk offset
BSS segment
Uninitialized static variables, filled with zeros.
Example: static char *userName;
Data segment
Static variables initialized by the programmer.
Example: static char *gonzo = "God's own prototype";
end_data
start_data
Text segment (ELF)
Stores the binary image of the process (e.g., /bin/gonzo)
end_code
0x08048000
0

Memory Mapping

- Creates a new linear address interval.
- PRIVATE vs SHARED.
- MAP_ANON vs MAP_FILE.

Side Channel Attacks

- A way to extract sensitive information from a system by some means other than intended input and output channels.
- Timing information, power consumption, sound or electromagnetic leaks.
- Spectre & Meltdown.

Cache side channel techniques

- Flush-reload
- Prime-probe

Flush-reload

- Step 1. Flush: Flush shared address from cache
- Step 2. Wait for victim
- Step 3. Reload: Time access for accessing the shared address:

If fast timing in 3 was fast it was placed in cache by victim. If slow victim did not use the address

Page Cache Side Channel

Dave Chinner (LKML) on : kvm "virtio pmem" device

“Hmmmm. Sharing the host page cache direct into the guest VM. Sounds like a good idea, but.....

This means the guest VM can now run timing attacks to observe host side page cache residency, and depending on the implementation I'm guessing that the guest will be able to control host side page cache eviction, too (e.g. via discard or hole punch operations).

Which means this functionality looks to me like a new vector for information leakage into and out of the guest VM via guest controlled host page cache manipulation.

Mitigations

- mm/mincore: allow for making sys_mincore() privileged
- For virtualization:

  Guest should not evict R/O copy of shared page in host.

Linus said:

“And no, we're *never* going to stop all side channel leaks. Some parts of caching (notably the timing effects of it) are pretty fundamental.

So at no point is this going to be some kind of absolute line in the sand _anyway_. There is no black-and-white "you're protected", there's only levels of convenience.”

References

- Spectre and Meltdown: Powerful Reminders of Side Channel Attacks https://bit.ly/2ZIqx8W
- Page Cache Attacks: https://arxiv.org/pdf/1901.01161.pdf
- https://lwn.net/Articles/776801/
- https://lwn.net/Articles/778437/
- Cache side channel attacks: CPU Design as a security problem : https://bit.ly/2RZNR14

Questions?

pagupta@redhat.com