Greg Kroah-Hartman

THE LINUX FOUNDATION

Disclaimer

- This talk vastly over-simplifies things.
- See notes for full details and resources.

https://github.com/gregkh/presentation-spectre

Spectre

- Hardware bugs
- Valid code can be “tricked” into exposing sensitive data to attacking programs.
- Exploits the speculative execution model of modern CPUs.
- Many different variants.
- Is going to be with us for a very long time!

Different Variants

- 1 – Bounds Check Bypass (BCB)
- 2 – Branch Target Isolation (BTI)
- 3 – Rogue Data Cash Load (RDCL)
- 3a – Rogue System Register Read (RSRE)
- 4 – Speculative Store Bypass (SSB)
- 5 – Lazy Floating Point State Restore (LazyFP)
- L1TF – L1 Terminal Fault “Foreshadow”

variant 1 – Bounds check bypass

- Uses the kernel to read memory of another process or virtual machine.
- Fixed by core kernel changes.
- Lots of individual drivers still need to be fixed.

variant 1 – vulnerable code

```
int load_array(int *array, unsigned int user_value)
{
        if (user_value >= MAX_SIZE)
                return 0;

        return array[user_value];
}
```

variant 1 – vulnerable code

```
int load_dependent_array(int *array1, int *array2,
                         int index)
{
        int value1, value2;

        value1 = load_array(array1, index);
        value2 = load_array(array2, value1);

        return value2;
}
```

variant 1 – fixed code

```
int load_array(int *array, unsigned int user_value)
{
        if (user_value >= MAX_SIZE)
                return 0;

        user_value = array_index_nospec(user_value, MAX_SIZE);

        return array[user_value];
}
```

variant 1 – fixed code

```
#define array_index_nospec(index, size)                                 \
({                                                                      \
        typeof(index) _i = (index);                                     \
        typeof(size) _s = (size);                                       \
        unsigned long _mask = array_index_mask_nospec(_i, _s);\
                                                                        \
        BUILD_BUG_ON(sizeof(_i) > sizeof(long));                        \
        BUILD_BUG_ON(sizeof(_s) > sizeof(long));                        \
                                                                        \
        (typeof(_i)) (_i & _mask);                                      \
})
```

variant 1 – fixed code - x86

```
static inline unsigned long
        array_index_mask_nospec(unsigned long index,
                                unsigned long size)
{
        unsigned long mask;

        asm ("cmp %1,%2; sbb %0,%0;"
                        :"=r" (mask)
                        :"g"(size),"r" (index)
                        :"cc");
        return mask;
}
```

variant 1 – fixed code

```
int load_array(int *array, unsigned int user_value)
{
        if (user_value >= MAX_SIZE)
                return 0;

        user_value = array_index_nospec(user_value, MAX_SIZE);

        return array[user_value];
}
```

variant 1 – Fix dates*

x86

4.14.14	17 January 2018
4.9.77	17 January 2018
4.4.113	23 January 2018

ARM

4.15.4	17 February 2018
4.14.21	22 February 2018

*Fixes keep coming

- These are the “first fixed” dates.
- Later kernels get more fixes and improvements.
- Keep updating your kernel & microcode!

variant 1 – Fix dates again

x86

4.16.11	22 May 2018
4.14.43	22 May 2018
4.9.102	22 May 2018

ARM

4.16	1 April 2018
4.9.95	20 April 2018

variant 2 – Branch target injection

- Abuses the CPU branch predictor.
- Read data from kernel or other virtual machine.
- Fixed by compiler, kernel, & microcode updates.
- “retpoline”

variant 2 – Fix dates*

x86

4.15.9	11 March 2018
4.14.26	11 March 2018
4.9.87	11 March 2018
4.4.121	11 March 2018

*More fixes and optimizations happened in later kernels

Meltdown

- Spectre variant “3”
- Read kernel data from userspace.
- Fixed with page table isolation kernel changes (Kaiser for older kernels).
- Fix slows down enter/exit of the kernel.
- Implemented differently for different kernel releases and distros.

Meltdown – fix dates

x86

4.14.11	02 January 2018
4.9.75	05 January 2018
4.4.110	05 January 2018

ARM

4.15.4	17 February 2018
4.14.20	17 February 2018
4.9.93	08 April 2018

variant 3a – Rouge system register read

- Abuses the reading of system registers.
- Read data from kernel or other virtual machine.
- Kernel fix for Meltdown solves this problem.

variant 4 – Speculative Store Bypass

- Can execute and read beyond what is expected.
- Read data from kernel or other virtual machine.
- Minor kernel changes.
- Microcode update required for full protection.

variant 4 – Speculative Store Bypass

x86

4.16.11	22 May 2018
4.14.43	22 May 2018
4.9.102	22 May 2018

variant 5 – Lazy FP state restore

- Uses the old “lazy floating point restore” method to read memory of another process or virtual machine.
- Details to be published June 27.
- Linux kernel fixed in 2016.

variant 5 – Fix dates

x86

4.6	15 May 2016
4.4.138	16 June 2018

”Foreshadow” - L1 Terminal Fault

- Read any data in the L1 cache
- Breaks SGX “secure” enclaves
- “simple” fix for local attacks
- “complex” fix for virtual machines
- Intel-only
- Microcode update to help resolve the issue.

”Foreshadow” - Fix dates

x86

4.18.1	15 April 2018
4.14.63	15 April 2018
4.9.120	15 April 2018
4.4.148	15 April 2018

”Foreshadow” - Fix dates again

x86

4.18.2	17 April 2018
4.18.3	18 April 2018
4.18.4	22 April 2018
4.18.5	24 April 2018

Why this is such a big deal

- CPU bugs require software & microcode fixes.
- All operating systems are affected.
- Performance will decrease.
- Totally new class of vulnerabilities.
- We will be finding, and fixing, these for a very long time.

Linux's response

- Companies were notified, but not developers.
- Developers notified very late, resulting in delay of fixes.
- Majority of the world runs non-corporate kernels.
- Intel is now working with some developers.

Keeping a secure system

- Take ALL stable kernel updates.
  - Do NOT cherry-pick patches.
- Enable hardening features.
- Update to a newer major kernel version where ever possible.
- Update your microcode/BIOS!