



Greg Kroah-Hartman



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 – Rouge Data Cash Load (RDCL)
- 3a – Rouge System Register Read (RSRE)
- 4 – Speculative Store Bypass (SSB)
- 5 – Lazy FP State Restore (LazyFP)



variant 1 - Bounds check bypass

- Use 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 - 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

```
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!



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 both kernel and 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)
- 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 17
- Linux kernel fixed in 2016



variant 5 – Fix dates

- x86
 - 4.6 15 May 2016
 - 4.4.138 16 June 2018

Why this is “unique”

- CPU bugs that require software & microcode fixes
- 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 directly with 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!



OPEN SOURCE SUMMIT

JAPAN

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT



AUTOMOTIVE LINUX SUMMIT





OPEN SOURCE SUMMIT
JAPAN

THE LINUX FOUNDATION



Greg Kroah-Hartman



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 – Rouge Data Cache Load (RDCL)
- 3a – Rouge System Register Read (RSRE)
- 4 – Speculative Store Bypass (SSB)
- 5 – Lazy FP State Restore (LazyFP)

1 2 and 3 were in the initial announcement,
January 4

3a and 4 were made public May 21, 2018

5 “leaked” June 13, full details to be published
June 27



variant 1 – Bounds check bypass

- Use 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 – 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

```
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!



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 both kernel and 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)
- Slows down enter/exit of the kernel
- Implemented differently for different kernel releases and distros

This is variant “3” of spectre

It allows a userspace program to be able to read data from within the kernel

It is fixed with a large number of kernel patches that implement “page table isolation”. This moves all kernel memory outside of the system when entering/exiting userspace, preventing userspace from being able to see kernel memory entirely.

The “Kaiser” paper first suggested this solution for a different type of kernel vulnerability, and this is what it is sometimes called

It slows down every time you enter or exit the kernel from userspace, which means that processes that do a lot of I/O accesses are hit hard.



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

This is variant “3” of spectre

It allows a userspace program to be able to read data from within the kernel

It is fixed with a large number of kernel patches that implement “page table isolation”. This moves all kernel memory outside of the system when entering/exiting userspace, preventing userspace from being able to see kernel memory entirely.

The “Kaiser” paper first suggested this solution for a different type of kernel vulnerability, and this is what it is sometimes called

It slows down every time you enter or exit the kernel from userspace, which means that processes that do a lot of I/O accesses are hit hard.



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 17
- Linux kernel fixed in 2016



variant 5 – Fix dates

- **x86**
 - 4.6 15 May 2016
 - 4.4.138 16 June 2018

Why this is “unique”

- CPU bugs that require software & microcode fixes
- 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 directly with 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!



THE
LINUX
FOUNDATION