Current state of mitigations for Spectre within operating systems

Ben Stuart

August 9, 2018

Fachbereich DCSM
M.Sc. Informatik
Hochschule RheinMain

WAMOS 2018

Outline

1. Introduction
2. Spectre-based attacks
3. Mitigation strategies
4. Current state
5. Conclusion

Introduction

Introduction i

Key Questions:

- What is Spectre-based attack?
- How can we mitigate it?
- Who is effected?

In particular, how did the operating systems mitigate?

Introduction ii

This presentation will not cover:

- Processor vendor mitigations (microcode).
- Detailed internals of a branch predictor.
- Detailed Spectre attack setup.

Spectre-based attacks

Spectre-based attacks i

Goal
Exploit speculative execution to leak sensitive information.

Spectre has two variants with different chances and behavior:

Variant 1 Bound check bypass of a buffer to leak sensitive information of the system [4].

Variant 2 Mistrain the branch predictor to jump to arbitrary locations [2].

Spectre-based attacks ii

Variant 1

- The attacker can provide a malicous chosen offset [4].
- Extract data from the cache [4].

Variant 2

- The attacker can mistrain the branch predictor [4].
- Use gagdets to extract information [2].

Mitigation strategies

Mitigation strategies

Several strategies are viable:

- Utilize the compiler.
- Rely on microcode mitigations.
- Use external static analysis.
- OS apply mitigations.
- (Apply mitigation patches by hand.)

The chosen strategy should be easy to implement!

Mitigation options

The Mitigations options can be put into several categories [4]:

- Prevent speculative execution.
- Prevent access to secret data.
- Prevent branch poisoning.
- Prevent data from entering a covert channel.
- Limit data extraction from a covert channel.

− Preventing speculative execution overall is the least attractive choice.

+ Preventing branch poisoning and prevent the access to secret data is viable.

Mitigation Variant 1: Non-speculative-array-access

Goal: The ability to detect and limit the scope of harm of speculative execution.

1. Ensure no out-of-bounds data is accessed.
2. Detect a speculative execution.

```
unsigned long mask = ~(long)(offset | (size - 1 - offset))
                            >> (BITS_PER_LONG - 1);
// Additional mask checks
// ...
return array[offset & mask];
```

Mitigation variant 1: GCC Built-in functions

GCC provides the built-in function *__builtin_speculation_safe_value* [1].

Additional benefits
It uses previous mitigation to detect speculative execution paths on a greater scale. Use register to track, if a speculative execution occurred and provide a fallback [1].

Mitigation Variant 2: Retpoline i

Idea
Inspired by return-orientated-programming: setup an infinite loop to capture speculative execution [5].

The retpoline has two variants:

- Indirect branch.
- Indirect Call.

The retpoline can be shared e.g. functions.

Retpoline ii

```
jmp *%r11

call set_up_target;
capture_spec:
    pause;
    jmp capture_spec;
set_up_target:
    mov %r11, (%rsp);
    ret;
```

- Instead of jumping to the location of %r11.
- Call *set_up_target* and override the return address.
- Speculative Execution will be trapped within *capture_spec*.
- The *call* instruction manipulated the return-stack-buffer of the branch predictor.

Current state

Current State

Mitigation Overview:

OS	Variant 1	Variant 2
Microsoft	Microcode	Microcode
Linux	non-speculative-array	Retpoline + *lfence*-instruction

Performance

Experience Reports

Microsoft mostly uses microcode mitigations and claims variant 1 has no impact on performance. They provide an unrealiable mitigation for variant 1 via Visual C++ [3].

Red Hat Linux Retpoline in combination with microcode mitigation caused system instabilities [6]. Variant 1 mitigations 1 caused no performance penalty.

Google Uses the retpoline mitigation on their servers. User reports indicated no performance hit [7].

Conclusion

Conclusion

- Software based solutions seem to properly mitigate the first two Spectre attacks.
- Unclear, whether this mitigations open up other exploits.

Thank you for your attention!

References i

Corbet, J., July 2018.

Jann Horn, P. Z.
Reading privileged memory with a side-channel, January 2018.

Kocher, P.
Spectre mitigations in microsoft's c/c++ compiler, February 2018.

- Kocher, P., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., and Yarom, Y.
  Spectre attacks: Exploiting speculative execution.
  ArXiv e-prints (Jan. 2018).
- Paul Turner, Senior Staff Engineer, T. I.
  Retpoline: a software construct for preventing branch-target-injection, 2018.
- RedHat, April 2018.
- Sloss, B. T., Jan 2018.