

マイクロアーキテクチャ攻撃演習 2

九州大学 サイバーセキュリティセンター
谷本 輝夫

演習の概要

- ▶ Spectre を実際にプロセッサシミュレータで実行
- ▶ プロセッサ内で命令が実行される様子を実際に見て、攻撃の仕組みをより深く理解する

今回の内容

▶ 今回は実践編

- ▶ プロセッサシミュレータでspectreを動かし、その実行の様子をパイプラインビューアで確認する
- ▶ spectre の脆弱性の回避方法を実践する

▶ 主な内容

1. gem5でspectreを実行
2. 実行履歴をパイプラインビューアで表示
3. 攻撃箇所を特定し、プログラムと照らし合わせる
4. spectre のプログラムを一部変更して脆弱性を回避する

Spectre proof of concept

- ▶ <https://gist.github.com/ErikAugust/724d4a969fb2c6ae1bbd7b2a9e3d4bb6>
 - ▶ 一部修正が必要（Dockerでは適用済み）
 - ▶ /home/gem5user/gem5-spectre/spectre に配置済み
- ▶ そのまま実行してみましょう

```
$ cd /home/gem5user/gem5-spectre  
$ spectre/spectre | less
```

- ▶ 1 プロセス内で直接データにアクセスせずに値を推測できることを実証するプログラム

Spectre 実行結果

```
@de409c4f9b09:~/gem5-spectre
Reading 40 bytes:
Reading at malicious_x = 0xffffffffffffdd8430... Success: 0x54='T' score=2
Reading at malicious_x = 0xffffffffffffdd8431... Success: 0x68='h' score=2
Reading at malicious_x = 0xffffffffffffdd8432... Success: 0x65='e' score=2
Reading at malicious_x = 0xffffffffffffdd8433... Success: 0x20=' ' score=2
Reading at malicious_x = 0xffffffffffffdd8434... Success: 0x40='M' score=2
Reading at malicious_x = 0xffffffffffffdd8435... Success: 0x61='a' score=2
Reading at malicious_x = 0xffffffffffffdd8436... Success: 0x67='g' score=2
Reading at malicious_x = 0xffffffffffffdd8437... Success: 0x69='i' score=2
Reading at malicious_x = 0xffffffffffffdd8438... Success: 0x63='c' score=2
Reading at malicious_x = 0xffffffffffffdd8439... Success: 0x20=' ' score=2
Reading at malicious_x = 0xffffffffffffdd843a... Success: 0x57='W' score=2
Reading at malicious_x = 0xffffffffffffdd843b... Success: 0x6f='o' score=2
Reading at malicious_x = 0xffffffffffffdd843c... Success: 0x72='r' score=2
Reading at malicious_x = 0xffffffffffffdd843d... Success: 0x64='d' score=2
Reading at malicious_x = 0xffffffffffffdd843e... Success: 0x73='s' score=2
Reading at malicious_x = 0xffffffffffffdd843f... Success: 0x20=' ' score=2
Reading at malicious_x = 0xffffffffffffdd8440... Success: 0x61='a' score=2
Reading at malicious_x = 0xffffffffffffdd8441... Success: 0x72='r' score=2
Reading at malicious_x = 0xffffffffffffdd8442... Success: 0x65='e' score=2
Reading at malicious_x = 0xffffffffffffdd8443... Success: 0x20=' ' score=2
Reading at malicious_x = 0xffffffffffffdd8444... Success: 0x53='S' score=2
Reading at malicious_x = 0xffffffffffffdd8445... Success: 0x71='q' score=2
Reading at malicious_x = 0xffffffffffffdd8446... Success: 0x75='u' score=2
Reading at malicious_x = 0xffffffffffffdd8447... Success: 0x65='e' score=2
Reading at malicious_x = 0xffffffffffffdd8448... Success: 0x61='a' score=2
Reading at malicious_x = 0xffffffffffffdd8449... Success: 0x60='m' score=2
Reading at malicious_x = 0xffffffffffffdd844a... Success: 0x69='i' score=2
Reading at malicious_x = 0xffffffffffffdd844b... Success: 0x73='s' score=2
Reading at malicious_x = 0xffffffffffffdd844c... Success: 0x68='h' score=2
Reading at malicious_x = 0xffffffffffffdd844d... Success: 0x20=' ' score=2
Reading at malicious_x = 0xffffffffffffdd844e... Success: 0x4f='O' score=2
Reading at malicious_x = 0xffffffffffffdd844f... Success: 0x73='s' score=2
Reading at malicious_x = 0xffffffffffffdd8450... Success: 0x73='s' score=2
Reading at malicious_x = 0xffffffffffffdd8451... Success: 0x69='i' score=2
Reading at malicious_x = 0xffffffffffffdd8452... Success: 0x66='f' score=2
Reading at malicious_x = 0xffffffffffffdd8453... Success: 0x72='r' score=2
:
```

Spectre の解析

▶ Objdumpしてみよう

```
$ cd /home/gem5user/gem5-spectre  
$ objdump -D spectre/spectre | less
```

```
void victim_function(size_t x) {  
    if (x < array1_size) {  
        temp &= array2[array1[x] * 512];  
    }  
}
```

投機実行させたい処理
spectre/spectre.c

Spectre の解析

▶ Objdumpしてみよう

```
$ cd /home/gem5user/gem5-spectre
$ objdump -D spectre/spectre | less
```

```
void victim_function(size_t x) {
    if (x < array1_size) {
        temp &= array2[array1[x] * 512];
    }
}
```

該当する命令列
投機実行させたい命令はどれ？

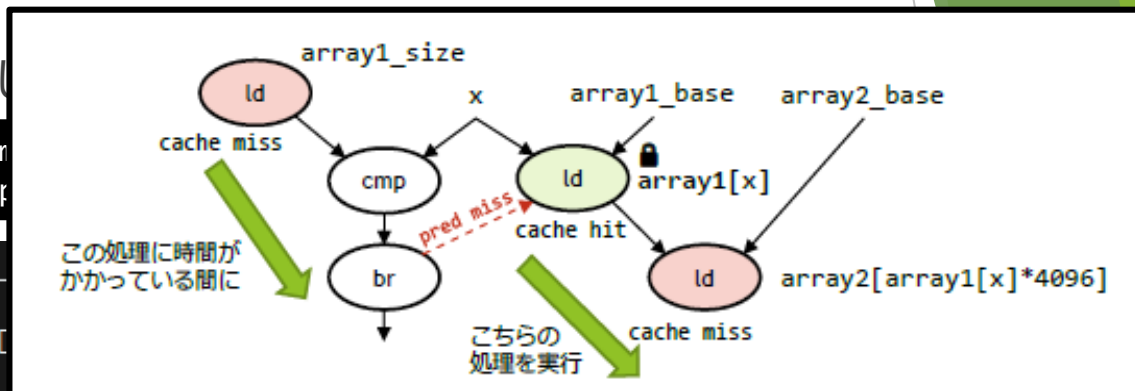
```
@de409c4f9b09:~/gem5-spectre
000000000400a7e <victim_function>:
400a7e: 55                push    %rbp
400a7f: 48 89 e5          mov     %rsp,%rbp
400a82: 48 89 7d f8       mov     %rdi,-0x8(%rbp)
400a86: 8b 05 74 36 2b 00 mov     0x2b3674(%rip),%eax    # 6b4100 <array1_size>
400a8c: 89 c0            mov     %eax,%eax
400a8e: 48 3b 45 f8       cmp     -0x8(%rbp),%rax
400a92: 76 2b            jbe     400abf <victim_function+0x41>
400a94: 48 8b 45 f8       mov     -0x8(%rbp),%rax
400a98: 48 05 20 41 6b 00 add     $0x6b4120,%rax
400a9e: 0f b6 00          movzbl (%rax),%eax
400aa1: 0f b6 c0          movzbl %al,%eax
400aa4: c1 e0 09          shl     $0x9,%eax
400aa7: 48 98            cltq
400aa9: 0f b6 90 20 72 6b 00 movzbl 0x6b7220(%rax),%edx
400ab0: 0f b6 05 69 52 2b 00 movzbl 0x2b5269(%rip),%eax    # 6b5d20 <temp>
400ab7: 21 d0            and     %edx,%eax
400ab9: 88 05 61 52 2b 00 mov     %al,0x2b5261(%rip)    # 6b5d20 <temp>
400abf: 5d              pop     %rbp
400ac0: c3              retq
```

Spectre の解析

► Objdumpしてみましょ

```
$ cd /home/gem5user/gen
$ objdump -D spectre/sp
```

```
void victim_function(size_
  if (x < array1_size) {
    temp &= array2[array1[
  }
}
```



```
@de409c4f9b09:~/gem5-spectre
000000000400a7e <victim_function>:
400a7e: 55                push    %rbp
400a7f: 48 89 e5          mov     %rsp,%rbp
400a82: 48 89 7d f8       mov     %rdi,-0x8(%rbp)
400a86: 8b 05 74 36 2b 00 mov     0x2b3674(%rip),%eax    # 6b4100 <array1_size>
400a8c: 89 c0            mov     %eax,%eax
400a8e: 48 3b 45 f8       cmp     -0x8(%rbp),%rax
400a92: 76 2b            jbe     400abf <victim_function+0x41> ← if (x < array1_size)
400a94: 48 8b 45 f8       mov     -0x8(%rbp),%rax
400a98: 48 05 20 41 6b 00 add     $0x6b4120,%rax
400a9e: 0f b6 00         movzbl (%rax),%eax ← array1[x]
400aa1: 0f b6 c0         movzbl %al,%eax
400aa4: c1 e0 09         shl     $0x9,%eax
400aa7: 48 98            cltq
400aa9: 0f b6 90 20 72 6b 00 movzbl 0x6b7220(%rax),%edx ← load array2[array1[x] * 512]
400ab0: 0f b6 05 69 52 2b 00 movzbl 0x2b5269(%rip),%eax    # 6b5d20 <temp> ← load temp
400ab7: 21 d0            and     %edx,%eax
400ab9: 88 05 61 52 2b 00 mov     %al,0x2b5261(%rip)    # 6b5d20 <temp> ← store temp
400abf: 5d              pop     %rbp
400ac0: c3              retq
```


gem5でspectreを実行する

- ▶ 以下のようにして実行してみましょう

```
$ cd /home/gem5user/gem5-spectre
$ gem5/build/X86/gem5.opt ¥
  --debug-flags=O3PipeView ¥
  --debug-file=pipeview.txt ¥
  --debug-start=1306234700 ¥
  -d gem5out/spectre ¥
gem5/configs/learning_gem5/part1/two_level_o3ltage.py ¥
spectre/spectre
```

- ▶ ¥ はバックスラッシュに読み替えてください
- ▶ --debug-flags : シミュレータのデバッグフラグを有効に
- ▶ --debug-file : デバッグ情報の出力先指定
- ▶ --debug-start : 指定した時刻からデバッグ出力を開始
- ▶ Success: を含む行が 2 ~ 3 行出力されたらControl-Cでシミュレータを停止

出力ファイルの変換

- ▶ 以下のコマンドで出力ファイルを変換

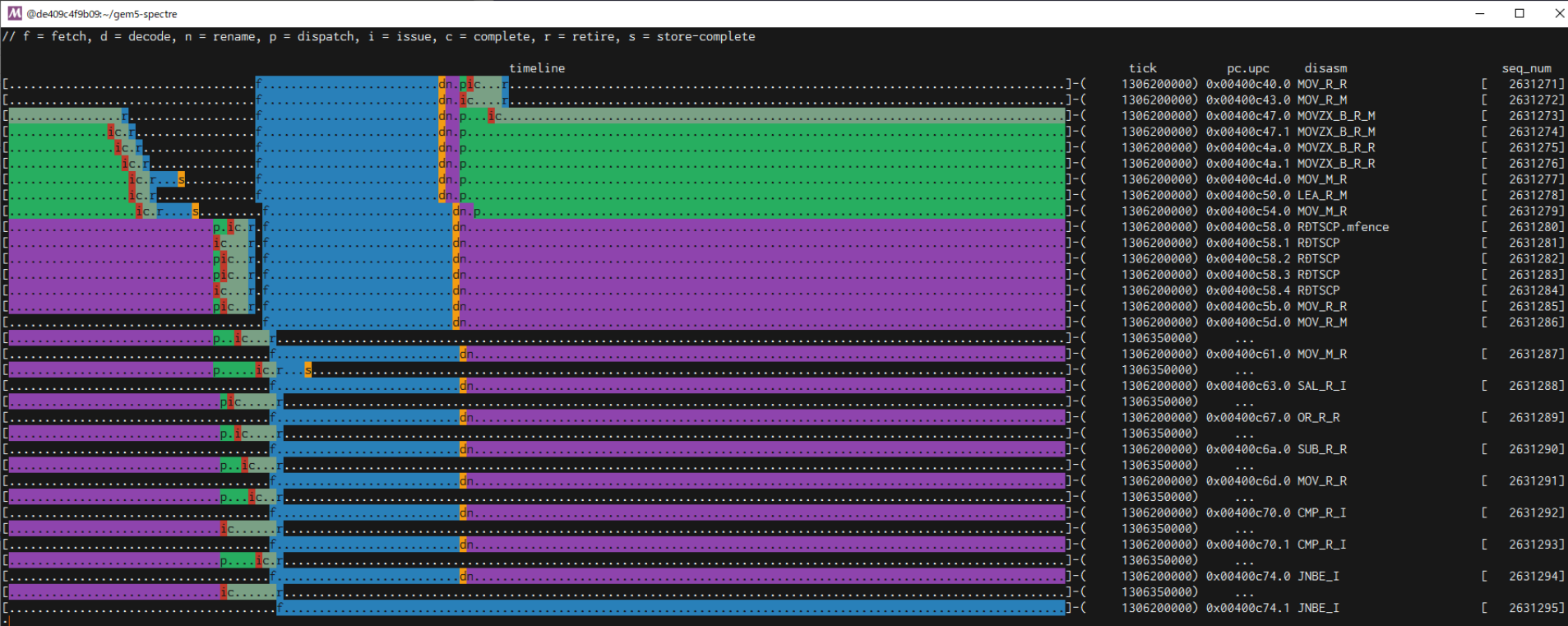
```
$ gem5/util/o3-pipeview.py --store_completions ¥  
gem5out/spectre/pipeview.txt --color -w 150
```

- ▶ -w 150 はターミナルの幅なので、お使いのターミナルの幅に合わせて変更してください
- ▶ o3-pipeview.out が出力されます

パイプラインの確認

- ▶ 以下のコマンドでパイプラインを表示

```
$ less -r o3-pipeline.out
```



投機ミスした処理

@de409c4f9b09:~/gem5-spectre

[.....i.c.....f.....dn.....]	1306350000	...	
[.....p.....i.c.....f.....dn.....]	1306200000	0x00400c70.1 CMP_R_I	[2631293]
[.....i.c.....f.....dn.....]	1306350000	...	
[.....i.c.....f.....dn.....]	1306200000	0x00400c74.0 JNBE_I	[2631294]
[.....i.c.....f.....dn.....]	1306350000	...	
[.....dn.i.c.....f.....dn.....]	1306200000	0x00400c74.1 JNBE_I	[2631295]
[.....dn.i.c.....f.....dn.....]	1306350000	...	
[.....dn.p.....i.c.....f.....dn.....]	1306200000	0x00400c74.2 JNBE_I	[2631296]
[.....dn.p.....i.c.....f.....dn.....]	1306350000	...	
[.....dn.i.c.....f.....dn.....]	1306200000	0x00400cb4.0 ADD_M_I	[2631297]
[.....dn.i.c.....f.....dn.....]	1306350000	...	
[.....dn.i.c.....f.....dn.....]	1306200000	0x00400cb4.1 ADD_M_I	[2631298]
[.....dn.i.c.....f.....dn.....]	1306350000	...	
[.....dn.p.....i.c.....f.....dn.....]	1306200000	0x00400cb4.2 ADD_M_I	[2631299]
[.....dn.p.....i.c.....f.....dn.....]	1306350000	...	
[.....dn.p.....i.c.....f.....dn.....]	1306200000	0x00400cb4.3 ADD_M_I	[2631300]
[.....dn.p.....i.c.....f.....dn.....]	1306350000	...	
[.....dn.i.c.....f.....dn.....]	1306200000	0x00400cb8.0 CMP_M_I	[2631301]
[.....dn.i.c.....f.....dn.....]	1306350000	...	
[.....dn=i.c.....f.....dn.....]	1306200000	0x00400cb8.1 ----CMP_M_I	[2631302]
[.....dn=p=i.c.....f.....dn.....]	1306200000	0x00400cb8.2 ----CMP_M_I	[2631303]
[.....dn=i.c.....f.....dn.....]	1306200000	0x00400cbf.0 ----JLE_I	[2631304]
[.....dn=i.c.....f.....dn.....]	1306200000	0x00400cbf.1 ----JLE_I	[2631305]
[.....dn=p=i.c.....f.....dn.....]	1306200000	0x00400cbf.2 ----JLE_I	[2631306]
[.....dn=p=i.c.....f.....dn.....]	1306200000	0x00400c00.0 ----MOV_R_M	[2631307]
[.....dn=i.c.....f.....dn.....]	1306200000	0x00400c03.0 ----IMUL_R_R_I	[2631308]
[.....dn=p.....f.....dn.....]	1306200000	0x00400c03.1 ----IMUL_R_R_I	[2631309]
[.....dn=p.....f.....dn.....]	1306200000	0x00400c03.2 ----IMUL_R_R_I	[2631310]
[.....dn=i.c.....f.....dn.....]	1306200000	0x00400c03.3 ----IMUL_R_R_I	[2631311]
[.....dn=p.....f.....dn.....]	1306200000	0x00400c09.0 ----ADD_R_I	[2631312]
[.....dn=p.....f.....dn.....]	1306200000	0x00400c09.1 ----ADD_R_I	[2631313]
[.....dn=i.c.....f.....dn.....]	1306200000	0x00400c0c.0 ----AND_R_I	[2631314]
[.....dn=p.....f.....dn.....]	1306200000	0x00400c0c.1 ----AND_R_I	[2631315]
[.....dn=p.....f.....dn.....]	1306200000	0x00400c11.0 ----MOV_M_R	[2631316]
[.....dn=p.....f.....dn.....]	1306200000	0x00400c14.0 ----MOV_M_R	[2631317]
[.....dn=p.....f.....dn.....]	1306200000	0x00400c17.0 ----SAL_R_I	[2631318]
[.....dn=p.....f.....dn.....]	1306200000	0x00400c1a.0 ----CDQE_R	[2631319]

Spectreが起きている箇所

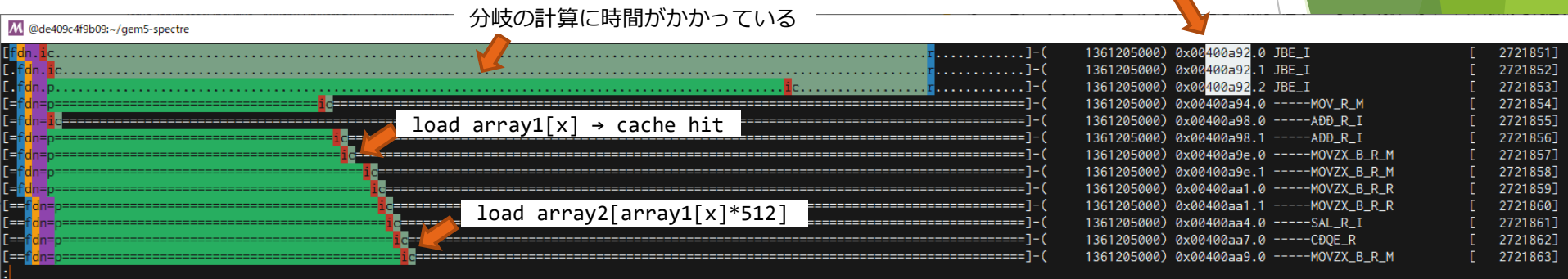
- ▶ パイプラインを調べてSpectreの攻撃が起きている箇所を特定しましょう
- ▶ ヒント

```
@de409c4f9b09:~/gem5-spectre
000000000400a7e <victim_function>:
400a7e: 55          push    %rbp
400a7f: 48 89 e5    mov     %rsp,%rbp
400a82: 48 89 7d f8  mov     %rdi,-0x8(%rbp)
400a86: 8b 05 74 36 2b 00  mov     0x2b3674(%rip),%eax    # 6b4100 <array1_size>
400a8c: 89 c0       mov     %eax,%eax
400a8e: 48 3b 45 f8  cmp     -0x8(%rbp),%rax
400a92: 76 2b       jbe     400abf <victim_function+0x41> ← リタイアしていること
400a94: 48 8b 45 f8  mov     -0x8(%rbp),%rax
400a98: 48 05 20 41 6b 00  add     $0x6b4120,%rax
400a9e: 0f b6 00    movzbl (%rax),%eax
400aa1: 0f b6 c0    movzbl %al,%eax ← 投機実行されていること
400aa4: c1 e0 09    shl     $0x9,%eax
400aa7: 48 98       cltq
400aa9: 0f b6 90 20 72 6b 00  movzbl 0x6b7220(%rax),%edx
400ab0: 0f b6 05 69 52 2b 00  movzbl 0x2b5269(%rip),%eax    # 6b5d20 <temp>
400ab7: 21 d0       and     %edx,%eax
400ab9: 88 05 61 52 2b 00  mov     %al,0x2b5261(%rip)    # 6b5d20 <temp>
400abf: 5d         pop     %rbp
400ac0: c3         retq
```

Spectreが起きている箇所

- ▶ 見つかりましたか？

予測ミスを起こす分岐命令



- ▶ 分岐結果が決まるまでの間にload命令が投機実行されてデータがキャッシュに乗る

Spectre 脆弱性の回避

- ▶ 資料 1 p.59を参考に spectre を回避しましょう
- ▶ 回避できたら、その実行の様子をパイプラインビューアで確認してください
- ▶ 対策前と比較して、性能にどのような影響があるか考察してください