BranchSpec: Information Leakage Attacks Exploiting Speculative Branch Instruction Executions

Md Hafizul Islam Chowdhury
University of Central Florida
Orlando, FL

Hang Liu
Stevens Institute of Technology
Hoboken, NJ

Fan Yao
University of Central Florida
Orlando, FL

38th IEEE International Conference on Computer Design
October 18 - 21, 2020
Background

- Security issues of speculation are raising critical concerns.
- Microarchitectural state changes remain beyond speculation.
- Unintended data could be exfiltrated via side channels.
  - E.g., Spectre and Meltdown.
  - Demonstrated using Cache, TLB and function units.
Motivation

❖ Branch predictor unit (BPU) is one of the most critical components
❖ BPU is used to trigger mis-speculation in transient execution attacks
❖ BPU can transfer secret in non-speculative domain (e.g., BranchScope\(^1\))

❖ Can we use branch predictor as transmission medium in transient execution domain?

1. BranchScope: A New Side-Channel Attack on Directional Branch Predictor, ASPLOS’2018
Modern Branch Predictor Architecture

Branch Predictor Unit

- Branch address
- Selection logic
- Global history register
- Pattern history table

Strongly taken
Weakly taken
Weakly not taken
Strongly not taken

Not taken
Taken
1. *Initialize {not-taken} the PHT state*

2. *Trigger *speculation*

3. *Measure execution time {taken}*

*Figure 1:* Execution time of branch ② in step 3 for different outcome of the branch in step 2.
Key Observation:

Branches executed in the speculative path change PHT entry which are not restored in case of mis-speculation.
Step 1: Preset PHT entry ($\text{PHT}_v$) of victim branch ($b_v$)

- Attacker uses a congruent branch of $b_v$ (i.e., $b_a$)
- Executes $b_a$ twice with $\textit{taken}$ outcome

Initial state ($\text{PHT}_v$)

Attacker:
- Initialization

Victim

```c
// Parent branch
if (x < bound)
    ....
    ....

// Victim branch, $b_v$
if (array1[x])
    <some_operations>;
```
**Step 2: Victim executes $b_v$ speculatively**

- Attacker can trigger mis-speculation of parent branch using congruent branch
- PHT entry of victim branch ($\text{PHT}_v$) is updated based on $b_v$ outcome

**Victim**

```plaintext
// Parent branch
if (x < bound)
    ....
    ....
// Victim branch, $b_v$
if (array1[x])
    <some_operations>;
```

**Speculative execution**

- $b_v$ resolved as **Not taken**
  - **ST** $b_v : \text{not-taken} \rightarrow \text{WT}$

- $b_v$ resolved as **taken**
  - **ST** $b_v : \text{taken} \rightarrow \text{ST}$
BranchSpec: Side Channel Attack

Step 3: Attacker probes PHT \( v_0 \) to infer \( b_0 \) outcome
- Execute \( b_a \) twice with not taken outcome
- Measure execution time of second execution

WN \( b_a \): not-taken
SN \( b_a \): not-taken

Predict: taken
Predict: not-taken

Correct prediction of \( b_a \rightarrow \) Shorter execution time
Mis-prediction of \( b_a \rightarrow \) Longer execution time

\( b_v \) resolved as Not taken
\( b_v \) resolved as taken
Results and Characteristics of BranchSpec

- First work to show information leakage via branch predictor in transient execution attacks
  - Implemented on processors with and w/o SMT
  - Bit error rate is less than 4%
  - Potentially targeted applications: Crypto algorithms, image processing and ML programs

- Enables even stronger attack capabilities
  - Completely uses BPU for end-to-end attack
  - Utilizes more common code patterns than Spectre V1

---

Spectre V1 Gadgets

```c
if (x < array1_size)
y = array2(array1[x] * 4096);
```

BranchSpec Gadgets

```c
if (x < bound)
    if (array1[x]) // b
        <some_operations>;

if (x < bound)
    for (i = 0; i < bound; i++)
        if (array1[x + i]) // b
            <some_operations>;

for (i = x; i < bound; i++)
    if (array1[i]) // b
        <some_operations>;
```
BranchSpec: Covert Channel Attack

- Covert channel using BranchSpec
  - With optimizations, 131 Kbps transmission rate within 3.7% error rate

**Figure 2:** Illustration of BranchSpec covert channel protocol.

**Figure 3:** Latency traces for a 50-bit transmission by Spy corresponding to the covert channel in Figure 2.
Potential Mitigations

- Existing system level defenses are ineffective
  - E.g., Retpoline, IBRS and others

- Potential architecture level mitigations
  - Restoring states for transient branches
  - Delaying PHT update
  - Enabling invisible PHT entry update
Conclusion

- Branches executed in speculation change PHT states, which are not restored after transient execution finishes.
- The vulnerability allows BPU to be used as *transmitting medium* in transient execution attacks.
- We demonstrate new forms of side and covert channels exploiting the discovered threat.
- We discuss potential mitigations to secure branch executions in speculative domain.
Thanks! Questions?

Md Hafizul Islam Chowdhuryy
Email: reyad@knights.ucf.edu

Source code available: https://github.com/fanyao/branchspec