Spectre Attacks: Exploiting Speculative Execution

P. Kocher et al., "Spectre Attacks: Exploiting Speculative Execution," 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2019, pp. 1-19, doi: 10.1109/SP.2019.00002.

Citations:
731 in paper
7 in patents
23805 full-text views

Presented by:

Prashant Mishra (2019CS50506)

Dinesh Joshi (2023ANZ8471)

Video Presentation Recording Link: https://drive.google.com/file/d/1wn2AWk6DLi30vegKp8HKvIigVBOOn0aq/view?usp=drive_link

Improving the Processor Performance

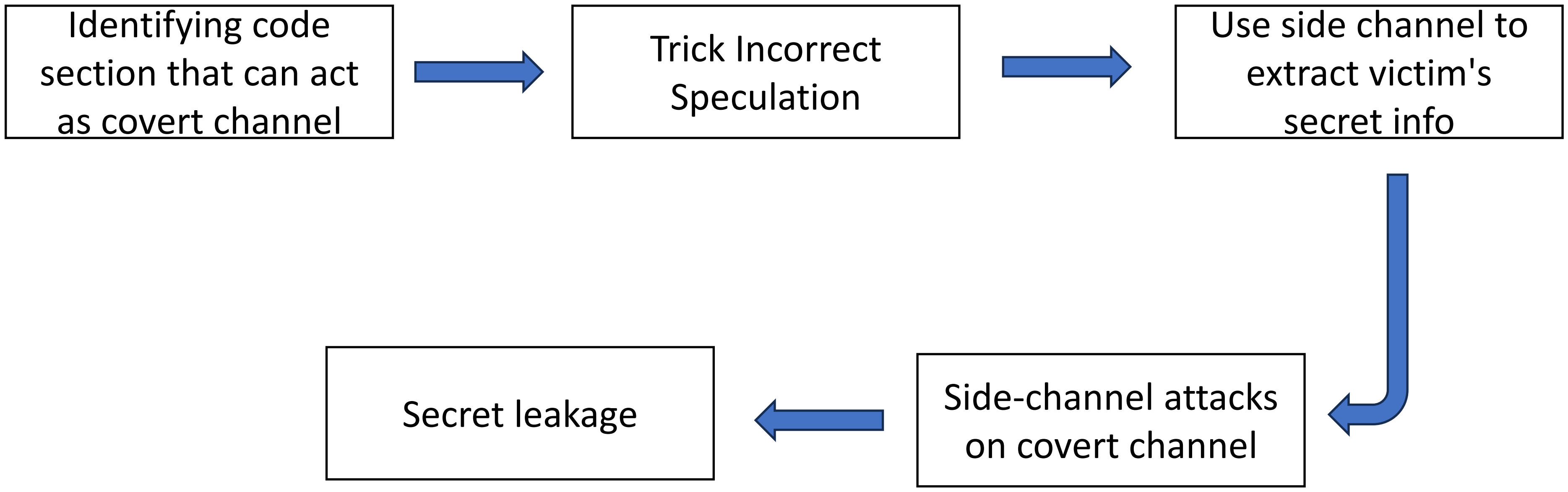
- Speculative Execution
  - Branch Prediction
  - Execution
  - Recovery

- Underlying Concept:
  - Predict branches based on previous history, and based on prediction, speculatively perform tasks early.
  - Later check if prediction correct or not. If yes, continue, else, do recovery.

Spectre

- Tricks the processor into speculatively executing instruction sequences.
- These instructions should not have been executed under correct program execution.
- Such instructions are called transient instructions.
- By influencing which transient instructions are speculatively executed, the victim's memory address information can be leaked.
- Spectre attacks violate memory isolation boundaries by combining speculative execution with data exfiltration via microarchitectural covert channels.

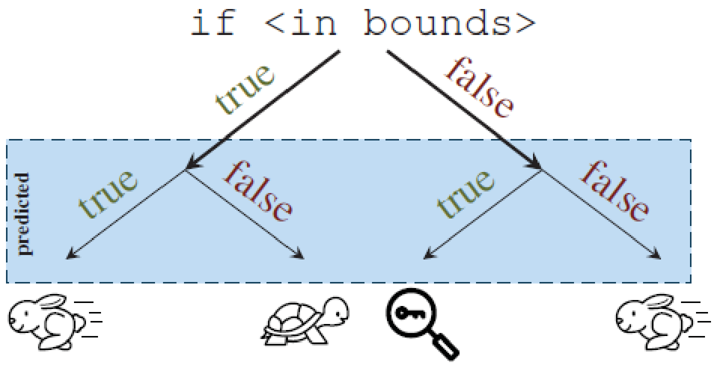
Mounting an Attack



Variant 1: Exploiting Conditional Branches

- Attacker mistrains the CPU’s branch predictor into mispredicting the direction of a branch
- CPU violates program semantics by executing code that would not have been executed otherwise.
- This incorrect speculative execution allows an attacker to read secret information stored in the program’s address space.

Branch predictor continues with the most likely branch target, leading to an overall execution speed-up if the outcome was correctly predicted. However, if the bounds check is incorrectly predicted as true, an attacker can leak secret information in certain scenarios.



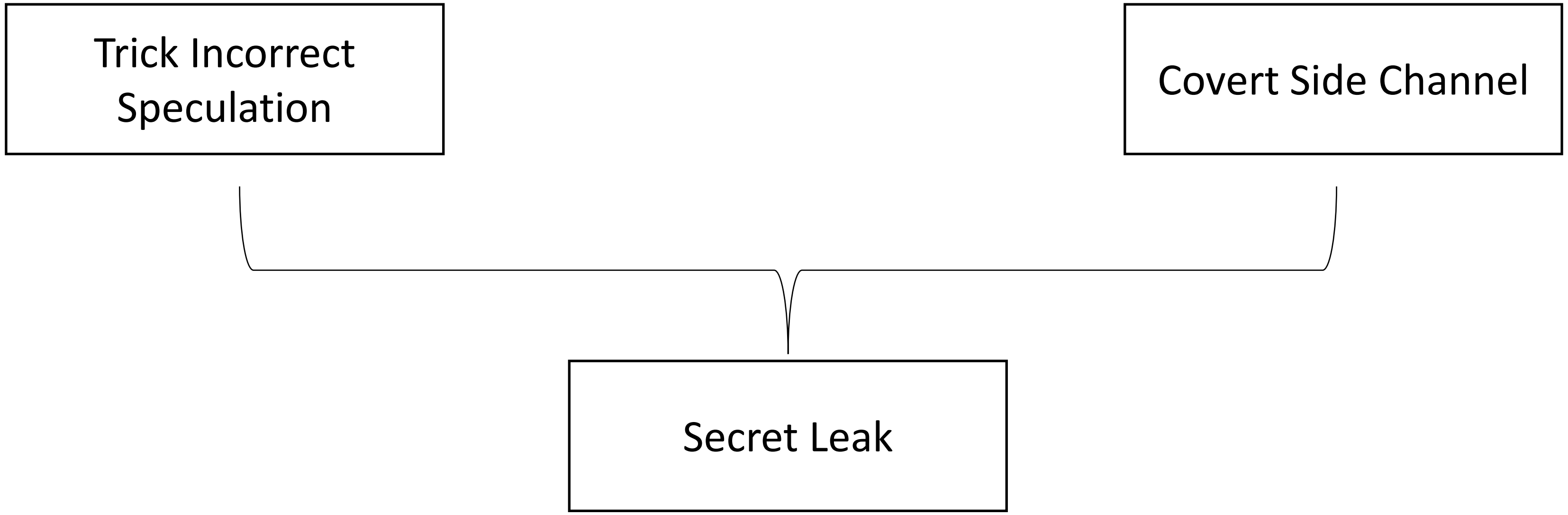

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

- Attacker controls x and mistrains to cause if condition to match.
- As a result, the array2 content is read and cached.
- Eventually the program correctness will be ensured, but array2 cached content is still in cache, which can be exploited.

Variant 2: Exploiting Indirect Branches

- The attacker chooses a gadget from the victim's address space and influences the victim to speculatively execute the gadget.
- Unlike Return Oriented Programming (ROP), the attacker does not rely on a vulnerability in the victim code.
- Instead, the attacker trains the Branch Target Buffer (BTB) to mispredict a branch from an indirect branch instruction to the address of the gadget, resulting in speculative execution of the gadget.
- While the effects of incorrect speculative execution on the CPU's nominal state are eventually reverted, their effects on the cache are not.
- Hence allowing the gadget to leak sensitive information via a cache side channel.

Many Other Variants



Spectre Mitigation Options

- Disable speculative execution
    - Add speculation disablement in software.
    - Add serialization modes to processor design.
    - Use lfence instructions.
- Preventing access to secret information
    - Apply bitmask to index instead of array bound checks, WebKit.
    - XORing pointers with pseudo random poison values.
- Preventing data from entering covert channels.
    - Detect if execution is in speculative mode and prevent entering covert channels in that case.

Conclusion

- Nearly all modern systems are impacted by Spectre.
- Spectre exploits architectural features.
- Fix requires software as well as hardware updates.
- Till now performance > security.
- Need to revisit architectural choices in view of increasing hardware and system security aspects.
- Performance vs Security trade-off.

Thank You