国立大学法人 電気通信大学

Pham Laboratory
Integrated circuit design laboratory

Cache side-channel Attack on RISC-V processor

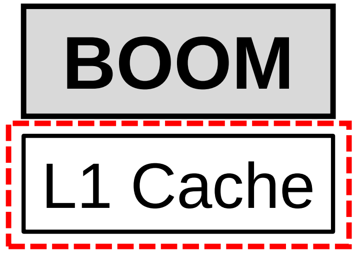
Tokyo, 15 Jan., 2022

国立大学法人
電気通信大学

Pham Laboratory
Integrated circuit design laboratory

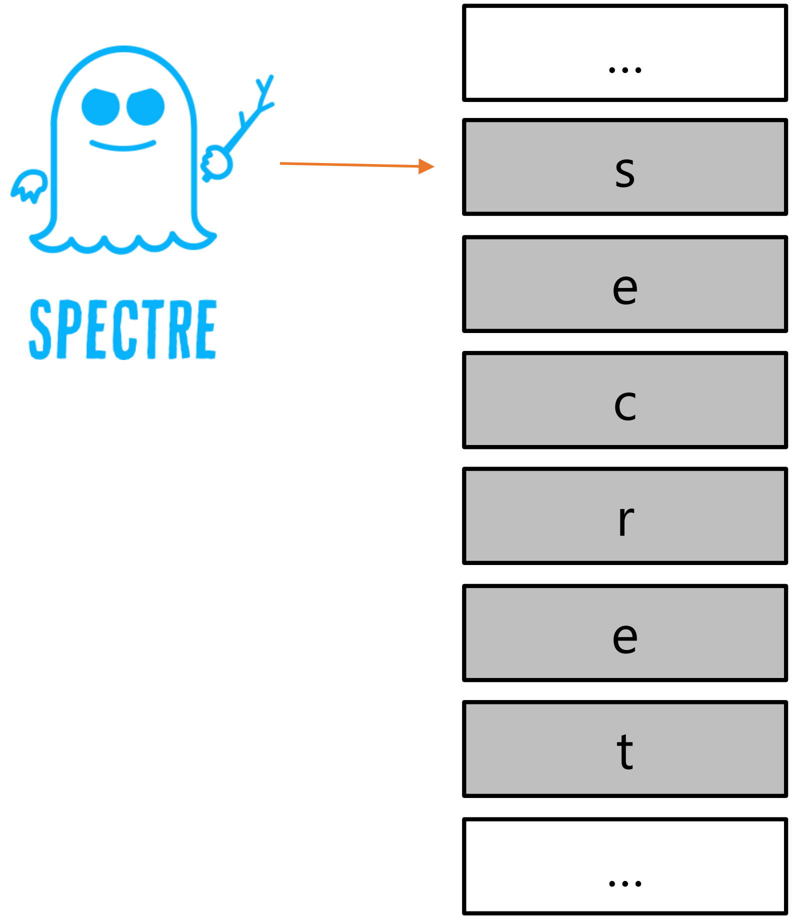
Introduction

- Spectre - Cache side-channel attack
- Target: RISC-V Out-of-order BOOM
- First variants:
  - Spectre v1: Bound Check Bypass
  - Spectre v2: Branch Target Injection



- Branch Predictor Unit
- Speculative Execution
- Caching
- …

BOOM suitable for Spectre

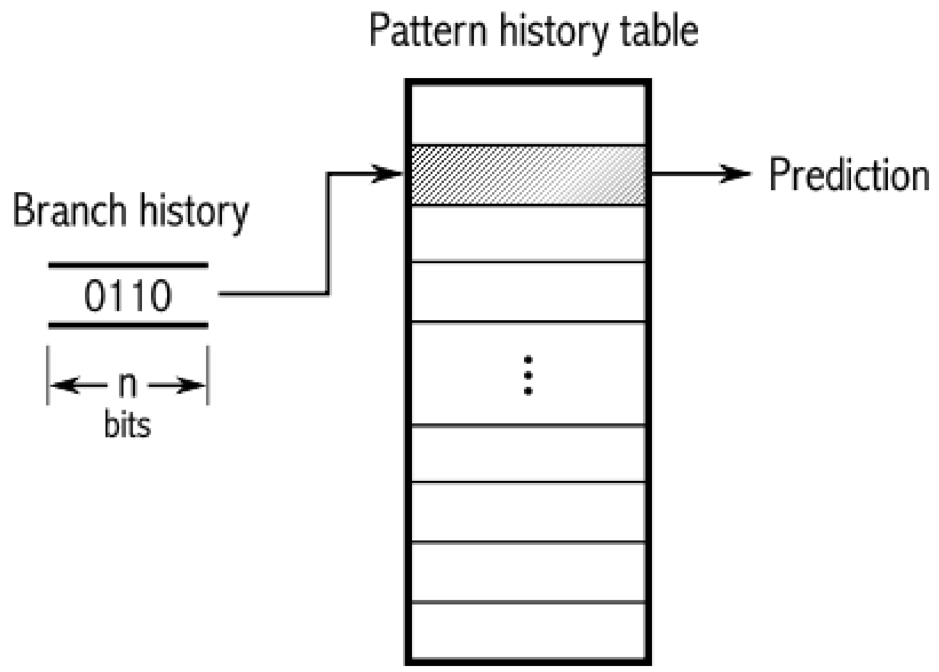


Cache memory

[1] BOOM (2015) [2] Spectre (2019)

Spectre attack (1)

User	Process
a = 1	TRUE => Run B
a = 2	TRUE => Run B
a = 3	TRUE => Run B
…	TRUE => Run B
a = X	*Maybe* TRUE => Run B

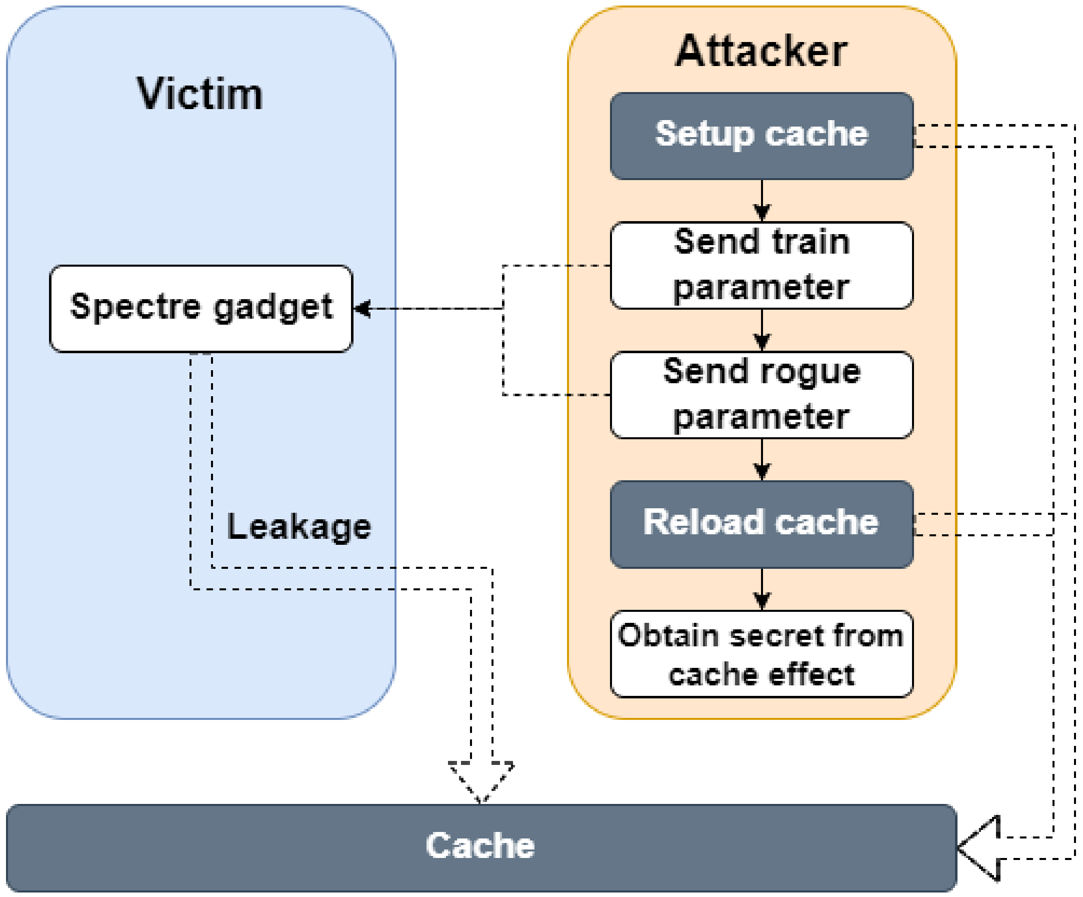


IF (a< 10)
Run B

Spectre attack (2)

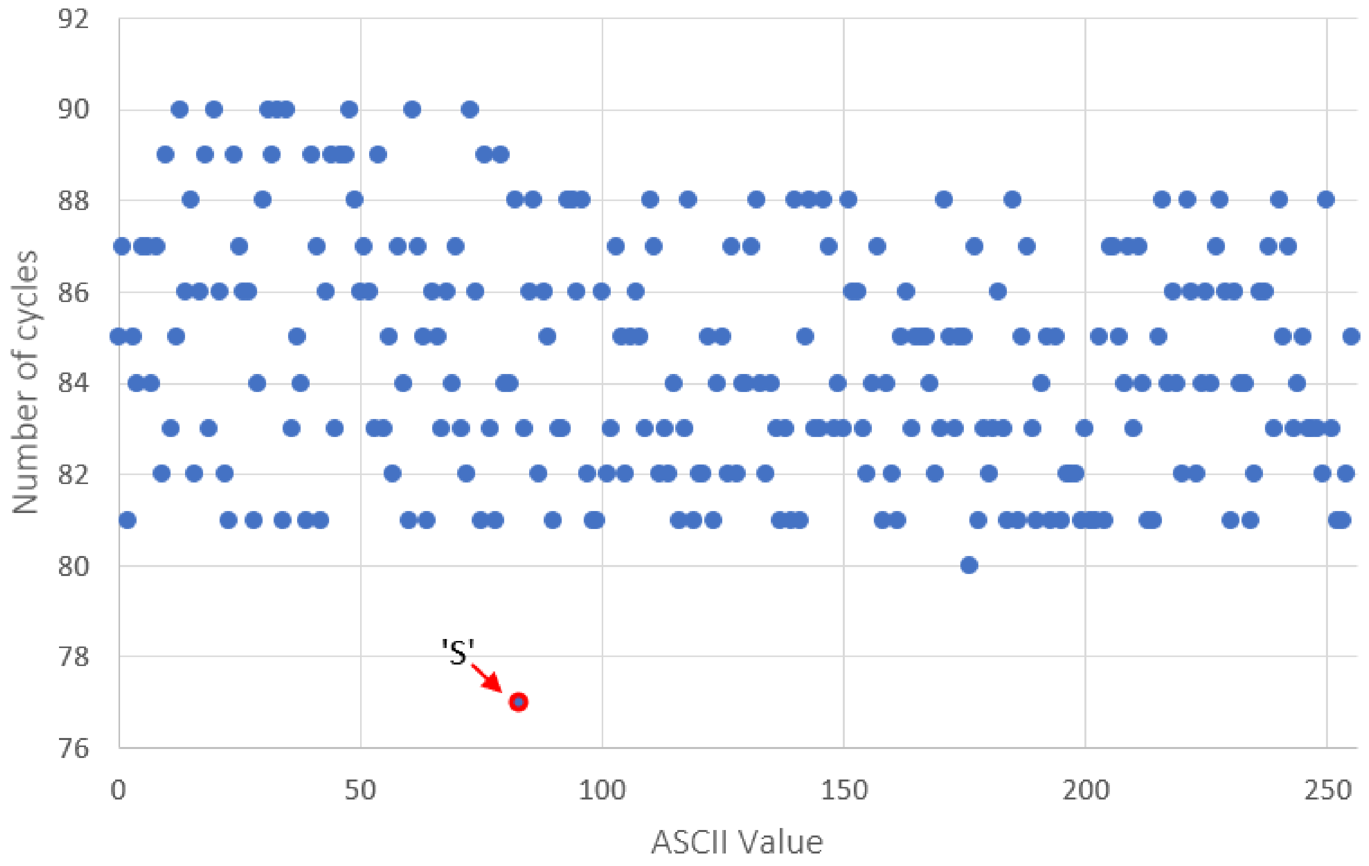
Typical attack strategy:

- Setup processor cache, for example, fill or flush all the cache lines, as in Flush+Reload, Prime+Probe timing attacks approaches.
- Force mis-speculation in victim code to leak secret into a side-channel
- Attacker recovers secret from side-channel effect in the cache (usually the access load time).



Spectre attack (3)

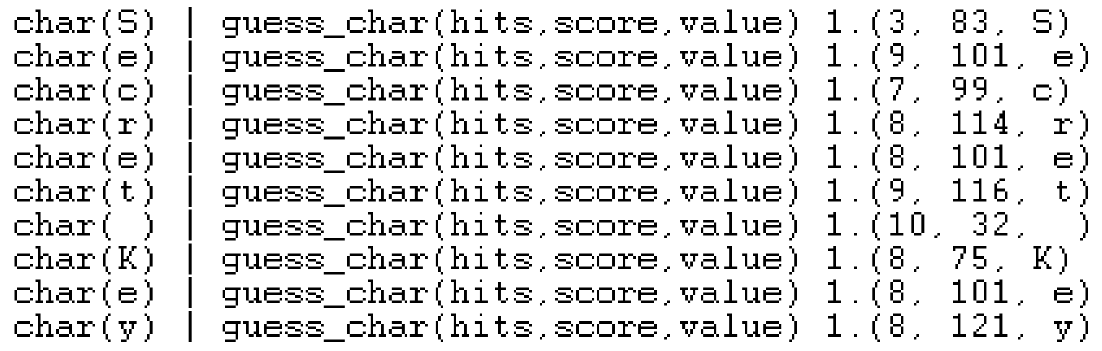
- Observe cache accessing time of a value.
- Each attack attempt is to collect 1 byte in secret string.



Spectre attack on RISC-V Processor

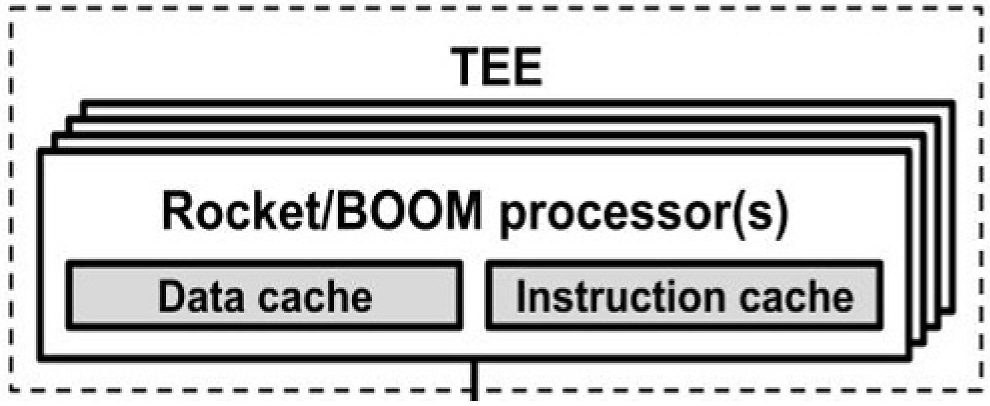
Implement RISC-V processor

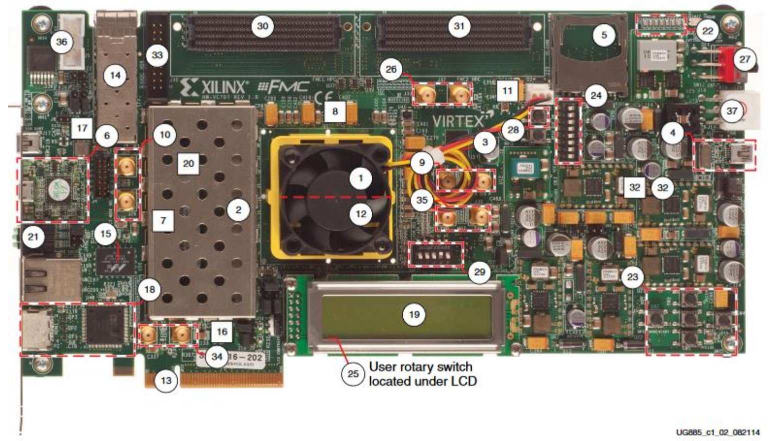
- BOOM core: exploited
- Rocket core: not exploited



```
char(S) | guess_char(hits,score,value) 1.(3, 83, S)
char(e) | guess_char(hits,score,value) 1.(9, 101, e)
char(c) | guess_char(hits,score,value) 1.(7, 99, c)
char(r) | guess_char(hits,score,value) 1.(8, 114, r)
char(e) | guess_char(hits,score,value) 1.(8, 101, e)
char(t) | guess_char(hits,score,value) 1.(9, 116, t)
char( ) | guess_char(hits,score,value) 1.(10, 32,  )
char(K) | guess_char(hits,score,value) 1.(8, 75, K)
char(e) | guess_char(hits,score,value) 1.(8, 101, e)
char(y) | guess_char(hits,score,value) 1.(8, 121, y)
```

Attack log (success case)





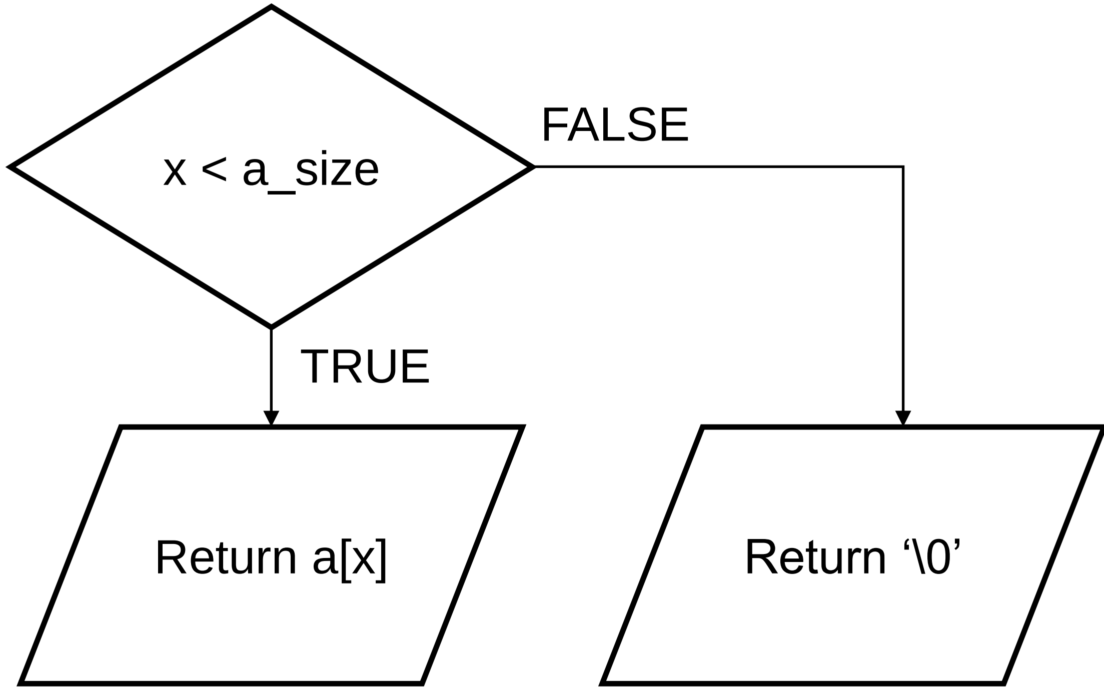
FPGA VC707

[3] TEE-Hardware (2020)   [4] Replicated Spectre (2020)

Software prevention (1)

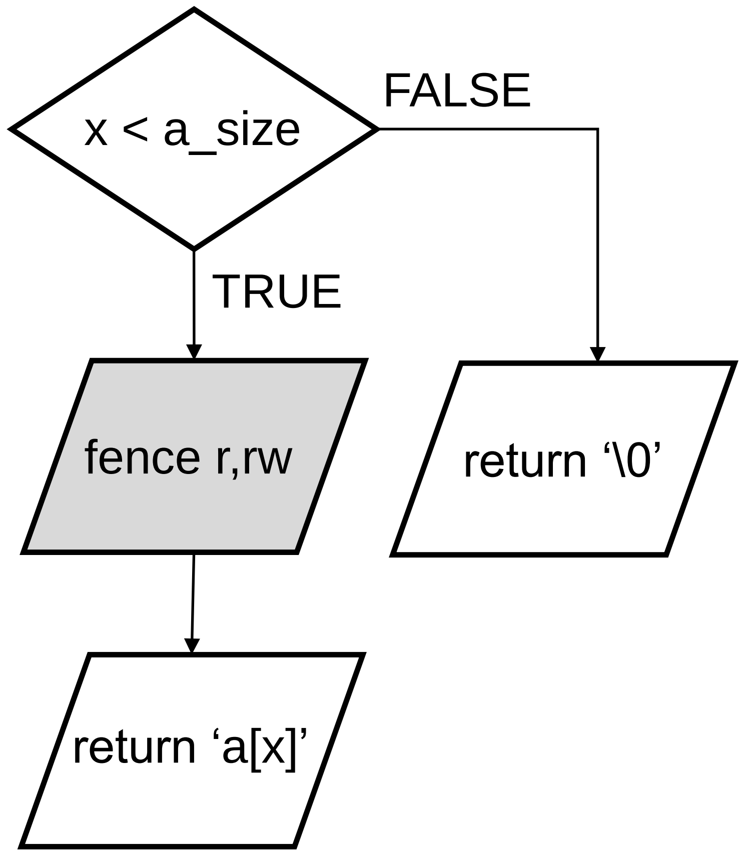
- Software method
  - Fence
  - Speculation Load Hardening
- Modify to strengthen victim program
- Require to re-compile source code
- Affect on performance

```
if (x < a_size)
   return a[x];
else
   return '0';
```

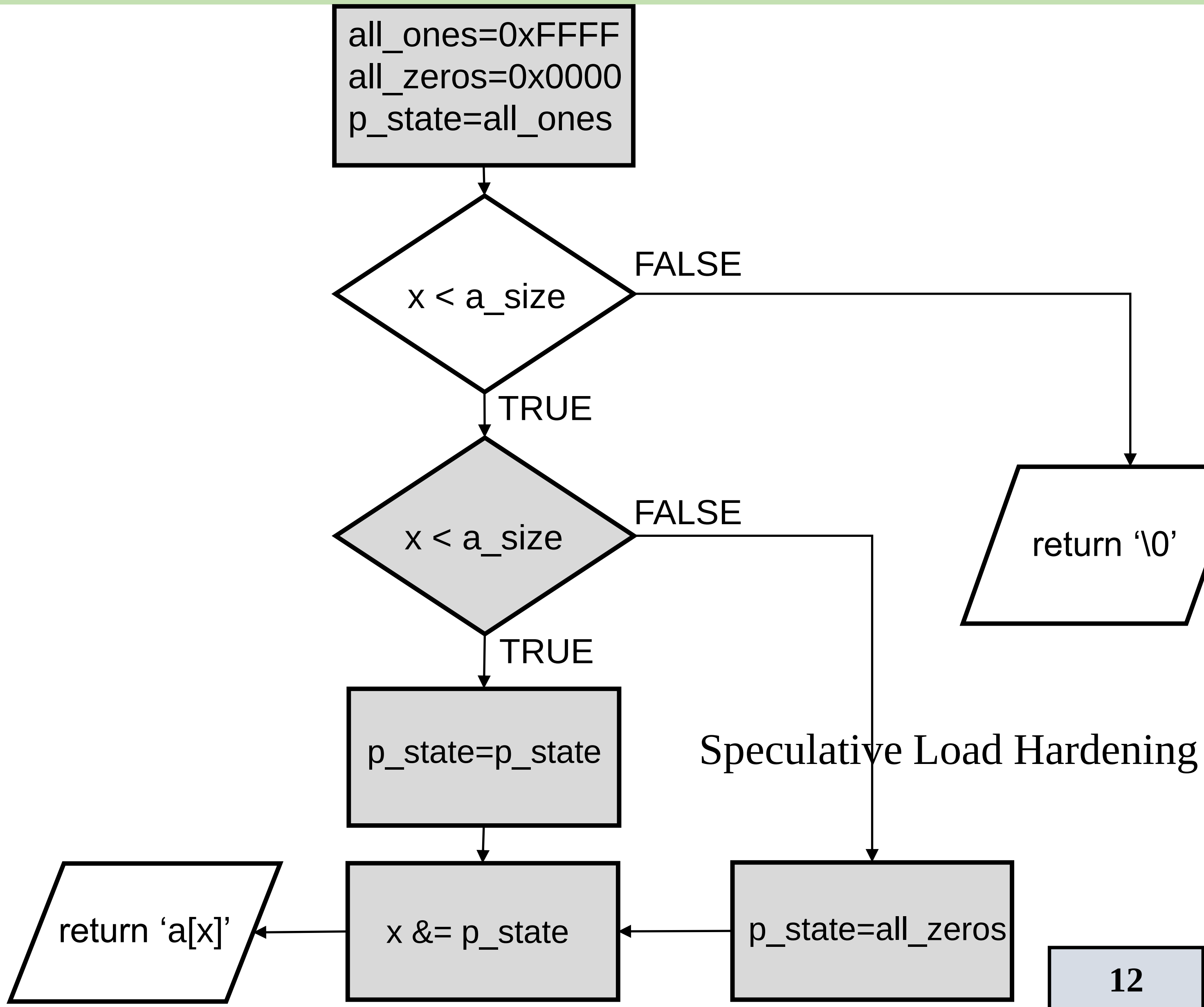


Original spectre v1 gadget

Software prevention (2)

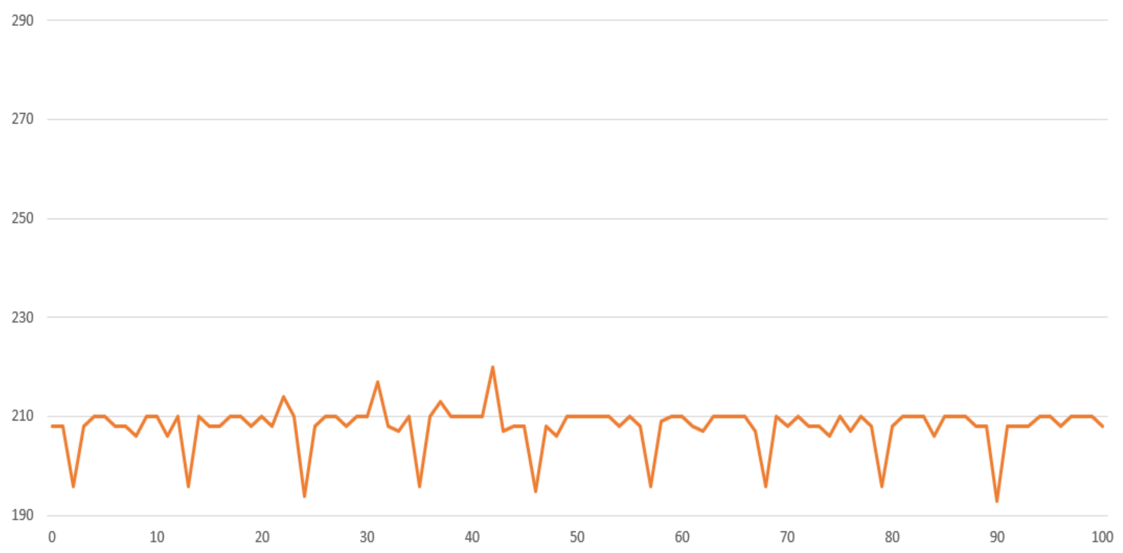


fence



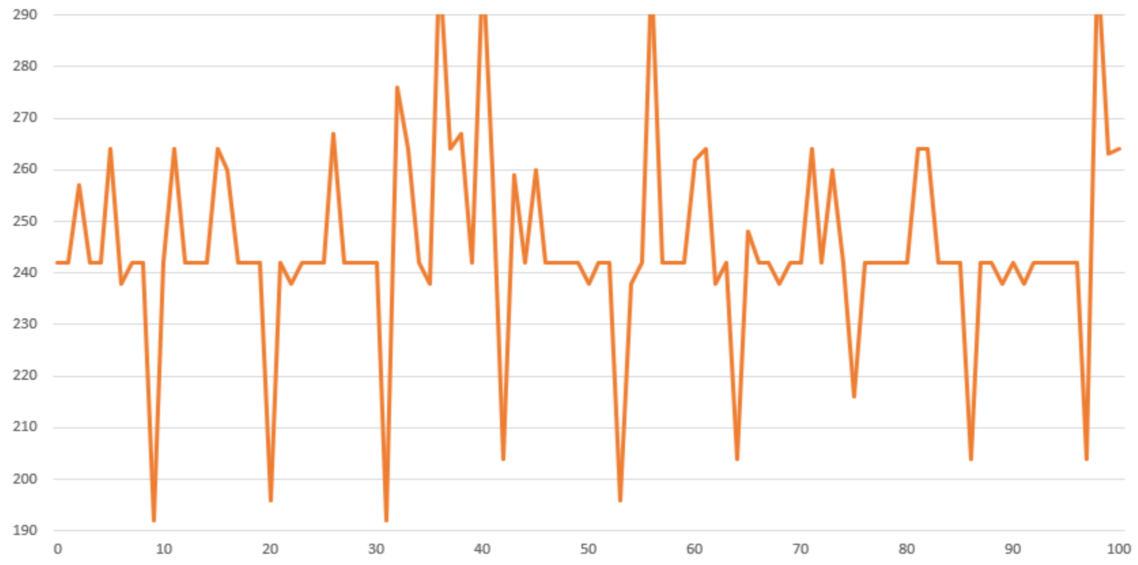
Speculative Load Hardening

Software prevention (2)



No mitigation

- Normal execution cycle: 210

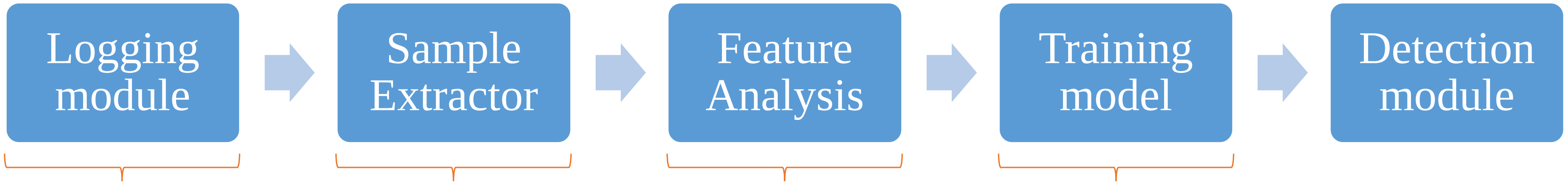


Mitigation using fence

- Normal execution cycle: 242 - 290
- Performance loss: 15 – 43%

Real-Time Spectre Attack Detection System (1)

- Target: RISC-V BOOM processor
- Using hardware performance counter (HPM) and machine learning
- Create a real-time background services
  - Gather hardware performance counter
  - Analyse to detect cache side-channel attack events
- System procedure:



Background monitoring program

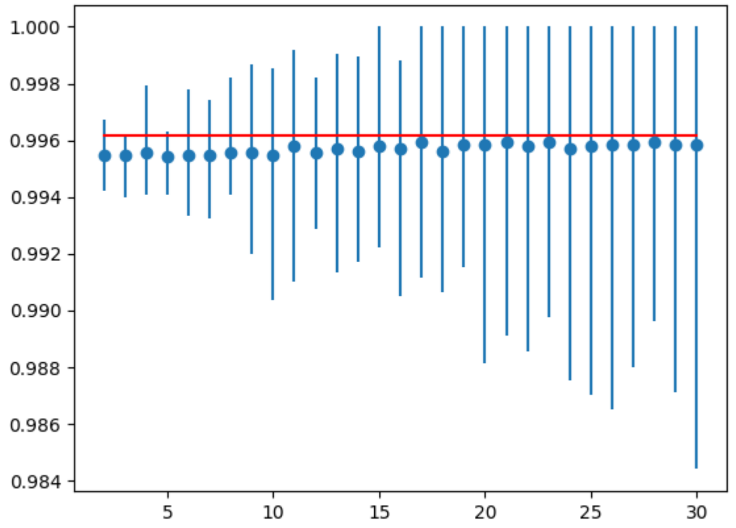
100ms/sample

Counters:
- Total of cycle
- Instruction retired
- Branch misprediction
- TLB miss

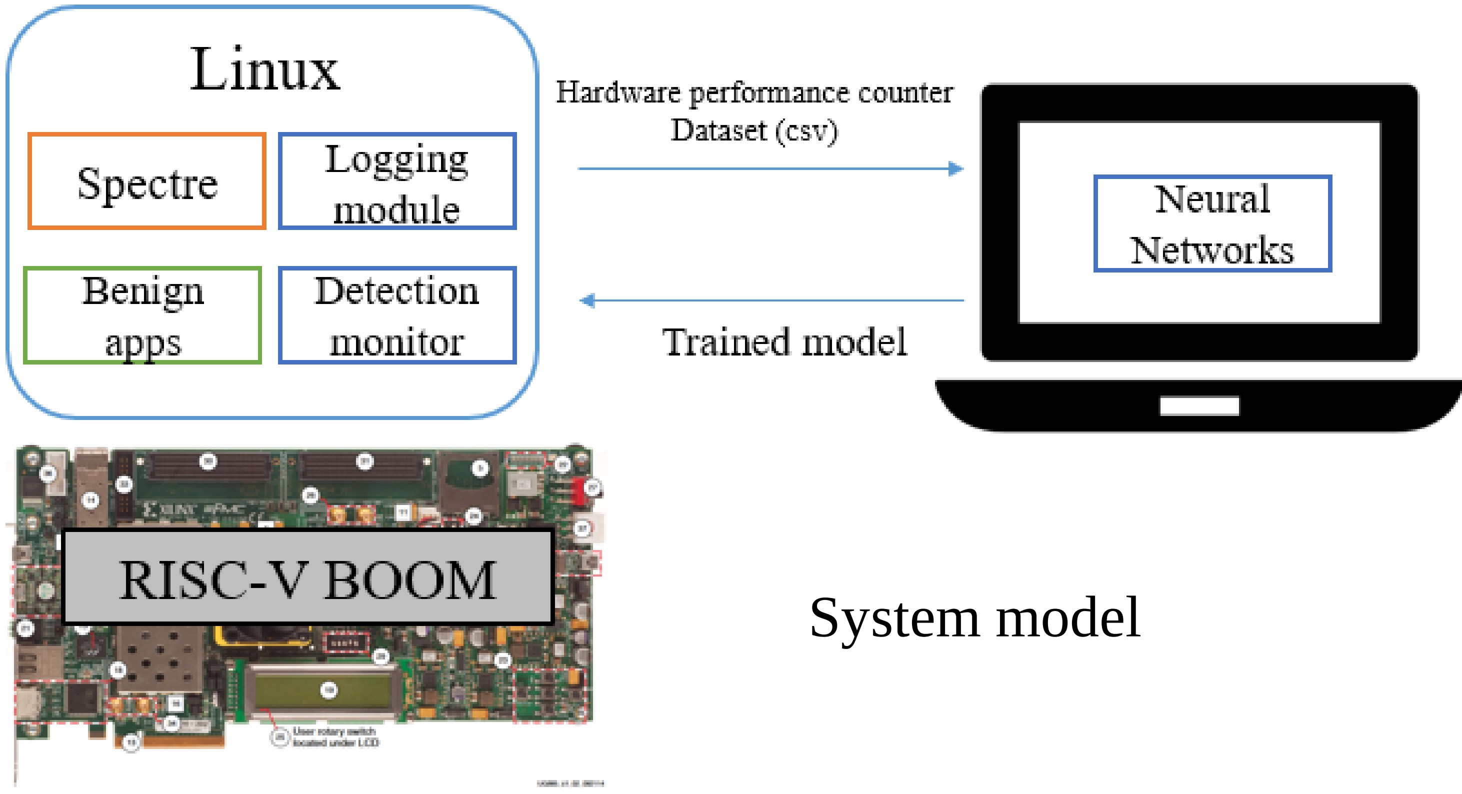
Model:
- Logistic Regression
- Support Vector Machine
- MLP network

Real-Time Spectre Attack Detection System (2)

- Dataset: 20.000 sample of HPMs
- Detection accuracy: 99.6 % (cross-validation)
- Performance overhead: 2.25%



Mean accuracy for cross-validation k-values.



System model

[4] Spectre Detection (2021)

Spectre mitigation researches on RISC-V Processor

Mitigation strategy	Detection	Prevention	
		Software approach	Hardware approach(*)
Idea or Research	• Analyse hardware performance counter • Use machine learning	• Fence instruction • Speculation Load Hardening (Index masking)	• Customized cache/processor
Benefits	• High accuracy and simple • Low performance overhead	• Strengthen victim program • Simple to implement	• Low performance overhead • Apply for wide range of threats
Drawbacks	• Need to find action after detection • Need to re-create model for new threat	• Require to re-compile victim code • High performance overhead: 15-43%	• Complicated. • Time consuming to develop

(*) Currently on research stage

Conclusion

1. **Replicated Spectre attack on FPGA:** Spectre (v1-v2), BOOM Core.
2. **Software mitigation methods:** Fence & Speculative Load Hardening, high performance overhead
3. **Real-Time Spectre Attack Detection System:** ~99.6% accuracy

References

1. C. Celio *et al.*, "*The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor*," *2015*.
2. P. Kocher *et al.*, "*Spectre attacks: Exploiting speculative execution*," in 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 1–19.
3. T. Hoang *et al.*, "*Quick Boot of Trusted Execution Environment With Hardware Accelerators*," in IEEE Access, vol. 8, pp. 74015-74023, 2020.
4. A. Le *et al*, "*Experiment on replication of side channel attack via cache of RISC-V Berkeley out-of-order machine (BOOM) implemented on FPGA*", CARRV 2020.
5. A. Le *et al*, "*A Real-Time Cache Side-Channel Attack Detection System on RISC-V Out-of-Order Processor*" in IEEE Access, vol. 9, pp. 164597-164612, 2021.

Thank You

Tokyo, 15 Jan., 2022