IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-2021)

Detection of Cache Side Channel Attacks using Thread Level Monitoring of Hardware Performance Counters

Pavitra Prakash Bhade

pavitra19231101@iitgoa.ac.in

Sharad Sinha

sharad@iitgoa.ac.in

School of Mathematics and Computer Science

Indian Institute of Technology Goa (IIT Goa), India

Outline

- Problem Statement
- Cache Memory: Introduction and Architecture
- Cache Attack: Flush + Reload Attack
- Proposed Technique
- Experiments and Results
- Conclusion

Problem Statement

- Modern microprocessors have cache memory structure that is vulnerable to cache side channel attacks
- Directly moving to mitigation of such attacks, have the following issues :
  - Need hardware modification in the architecture
  - Software upgrades have certain performance impact
  - Type of mitigation may be dependent on the type of attack
- Hence, "Detection" ensures that mitigation is applied on need basis, to serve as first line of defence against such attacks

Cache Memory : Structure

- Fast and small memory for frequent accesses by the processor.
- Most multicore architectures (ex. Intel x86) have hierarchical cache levels.
- Level 1 and Level 2 are individual per core, whereas the last level cache (LLC) is common across all cores
- Last Level Cache is inclusive in Intel architecture:
  - Element evicted from LLC is evicted from other levels as well.

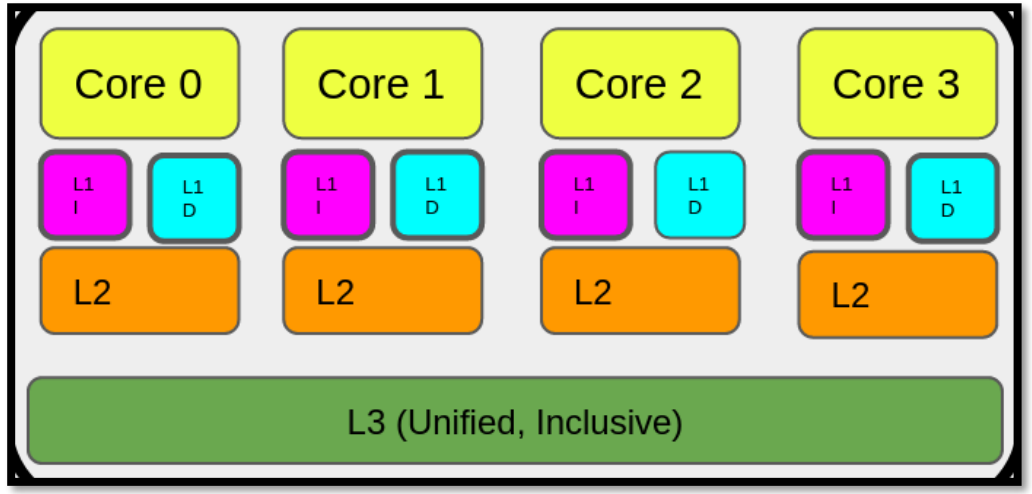


Fig: Intel Sandy Bridge Cache Architecture

Cache Memory : Timing analysis

- Presence and absence of data / instructions in the cache causes cache hit or cache miss respectively
- The time difference between cache hit and miss, can be analysed by an attacker to trace the victim activity and leak secrets from the shared cache.

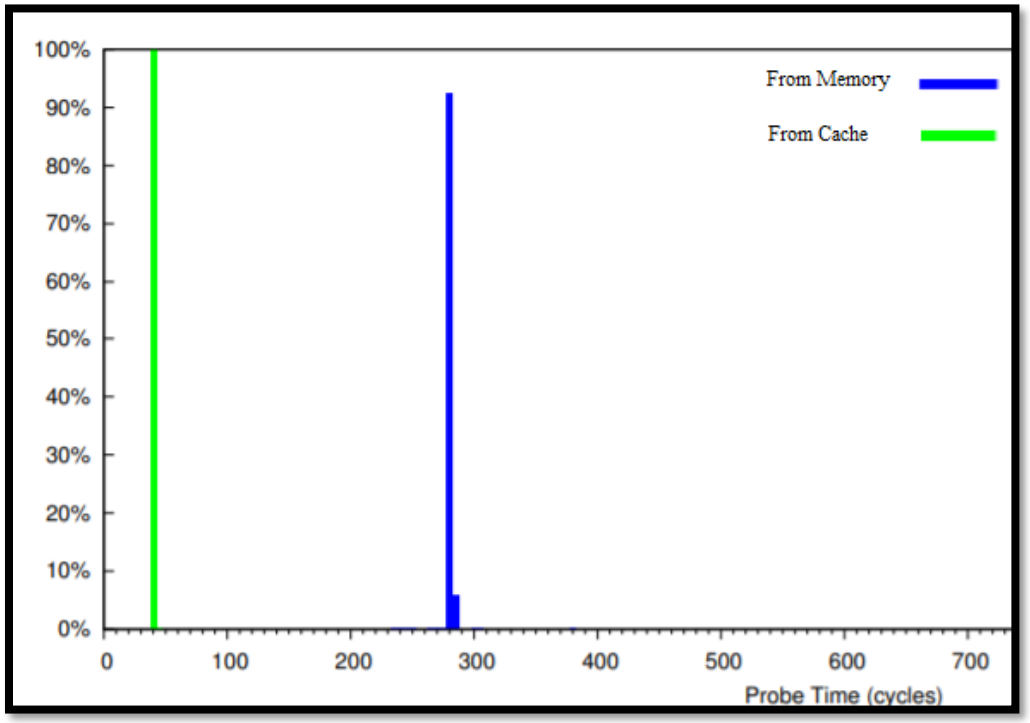


Fig: Difference in Cache Hit and Cache Miss Timings

Flush + Reload Attack

- Consider this square and multiply algorithm performing encryption/decryption function
- Steps 8 and 9 are dependent on the value of e
- If e is 1, 8 and 9 will be executed
- Else directly 10 will be executed
- Monitoring the cache hit/miss for instructions 8 and 9, reveals the value of e.
- Cache hit : e=1
- Cache miss : e=0

```
1  function exponent(b, e, m)
2  begin
3    x ← 1
4    for i ← |e| − 1 downto 0 do
5      x ← x^2
6      x ← x mod m
7      if (e_i = 1) then
8        x ← xb
9        x ← x mod m
10     endif
11   done
12   return x
13 end
```

Fig : Square and multiply algorithm for encryption/description

Detection of Cache Attacks

- Cache Side Channel attacks do not cause any architectural changes.
- But these attacks leave certain microarchitectural traces, which can be monitored for the purpose of detection
- Hardware performance counters can be monitored to detect suspicious activities if the counts go high.

Hardware Performance Counters (HPC)

- Eviction based cache attacks like Flush + Reload cause the data or instructions to forcefully be evicted from the cache
- This causes certain hardware performance counters like Cache hits, Cache misses, etc to show exorbitantly high counts
- Such counters can be monitored to detect the attack

Problem with system level HPCs

- The applications running on the system can sometimes mimic attack like scenario.
- Multiple genuine applications involving highly random memory accesses can cause high cache miss and other counts, which may give rise to false alarms in detection
- Such false alarms lead to unnecessary application of mitigation techniques, further impacting the performance

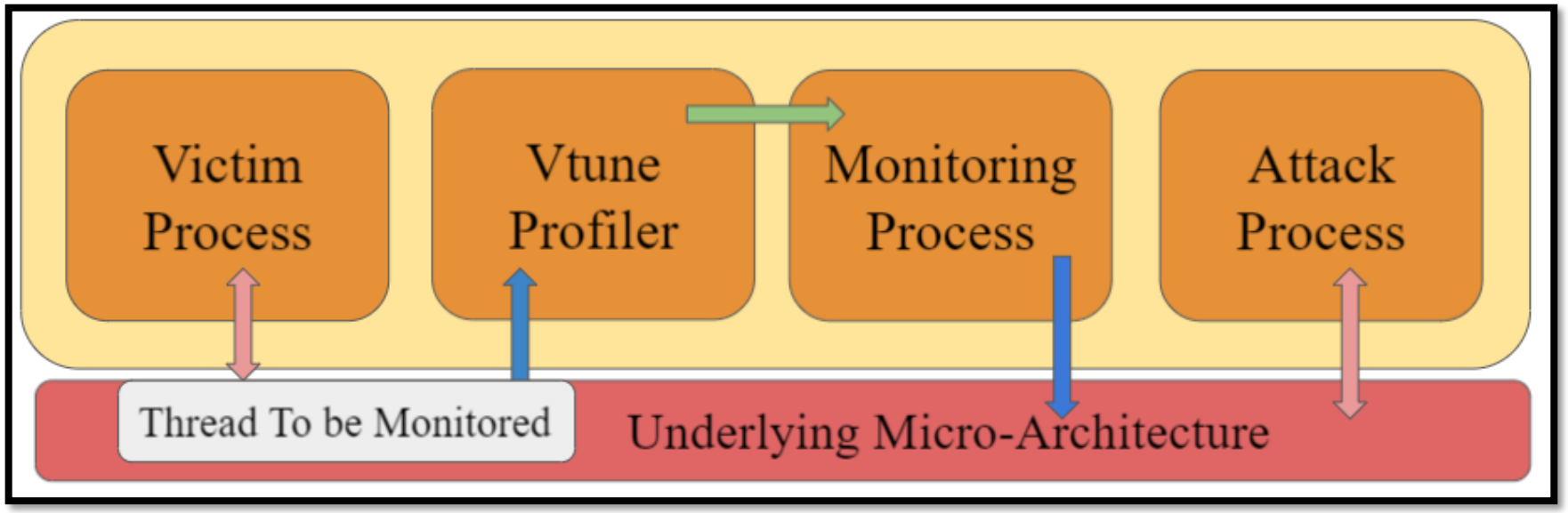
Proposed Technique : Thread Level HPC

- The Hardware performance counters can be fine grained to thread level, instead of the system level.
- Some section of the victim code is confidential and holds secrets.
- Protecting only this section is important, rather than the entire application or the system.
- This reduces monitoring overhead as well as chances of false alarms
- Hence only the thread performing the execution of this sensitive section should be monitored.

Implementation Details

- Operating System
  - Ubuntu 20.04 LTS
- Attack Performed
  - Cache Template Attack
    - Based on Flush + Reload method
    - Extracts keypresses of the victim
  - Three editors
    - Sublime Text
    - Gedit
    - Terminal
- Monitoring tool
  - Vtune Profiler
    - Filter using thread id

Proposed Framework

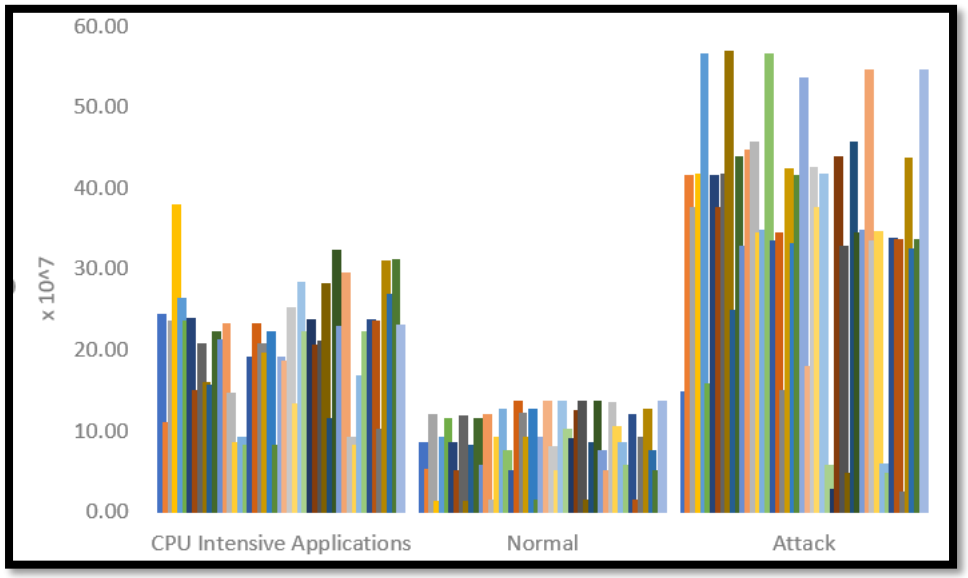


Experiments and Results

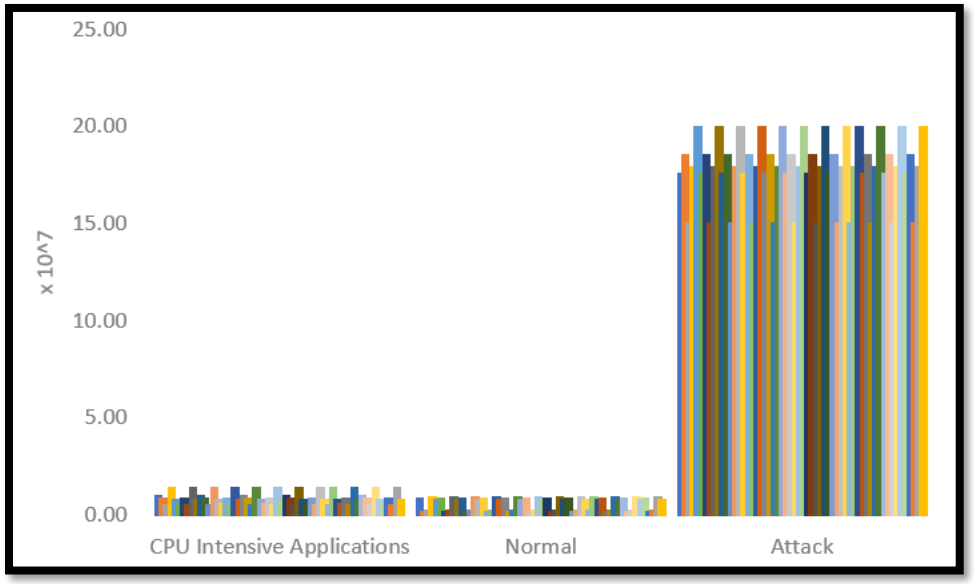
- Four Event Counters have shown exorbitantly high counts on an event of attack
  - DTLB LOAD MISS
  - MEM LOAD RETIRED L1 MISS
  - MEM LOAD RETIRED L3 HIT
  - DTLB STORE MISSES
- The experiments were conducted in three scenarios:
  - Attack (When there is a victim and attacker code running)
  - No Attack (When there is only victim code running)
  - No Attack system overloaded (When the victim code is running with other heavy applications running in the background, no attacker code)

Experiments and Results : DTLB LOAD MISS

System level

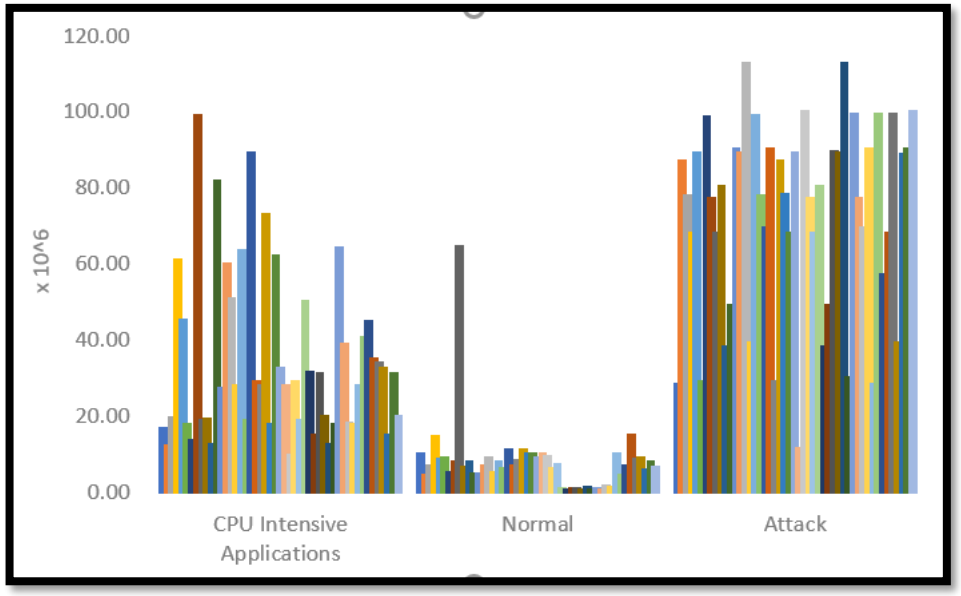


Thread Level

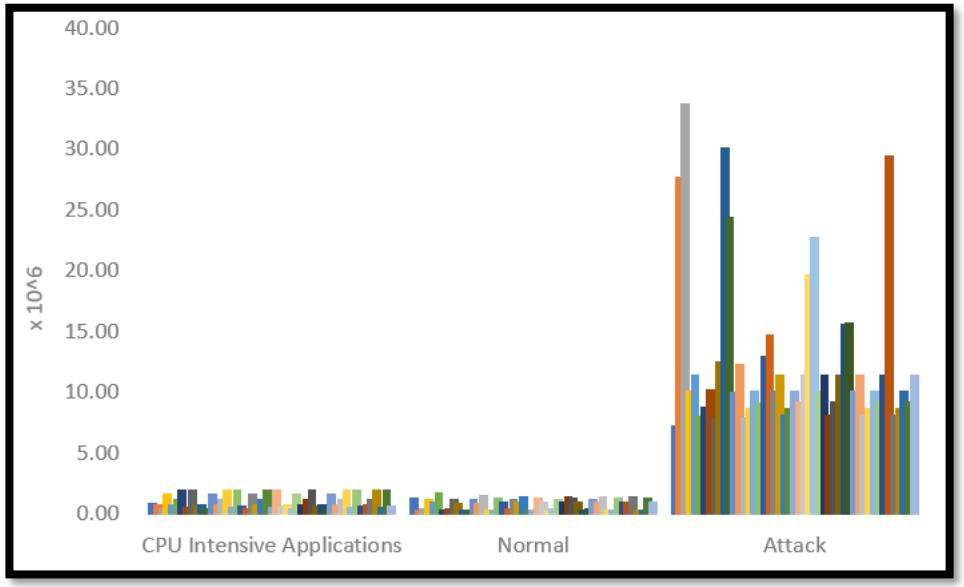


Experiments and Results : MEM LOAD RETIRED L1 MISS

System level

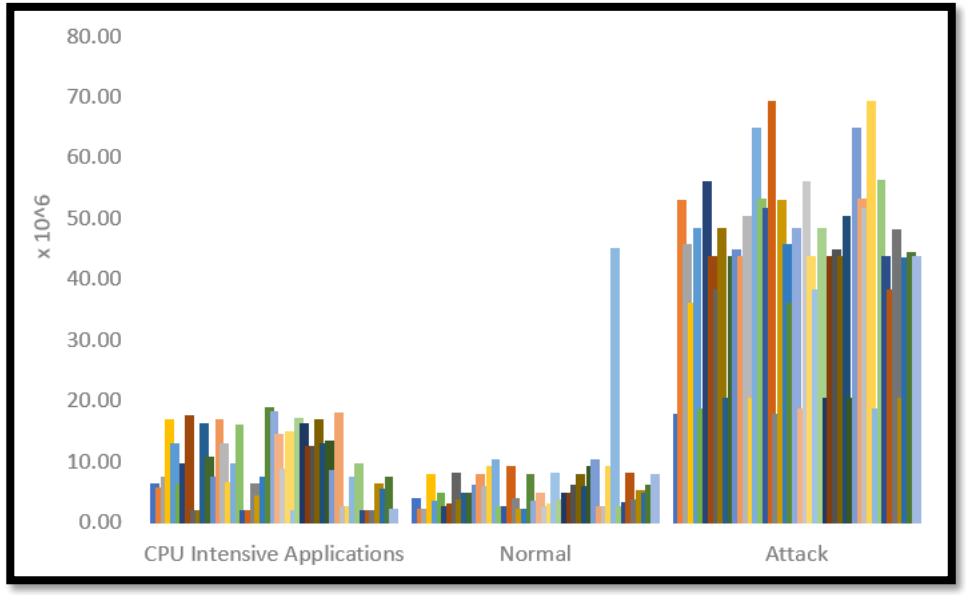


Thread Level

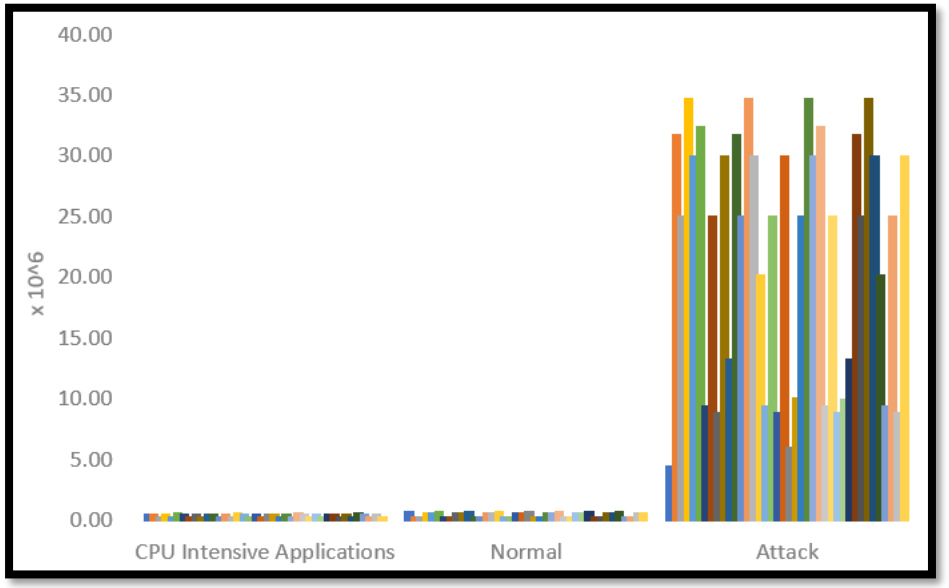


Experiments and Results : MEM LOAD RETIRED L3 HIT

System level

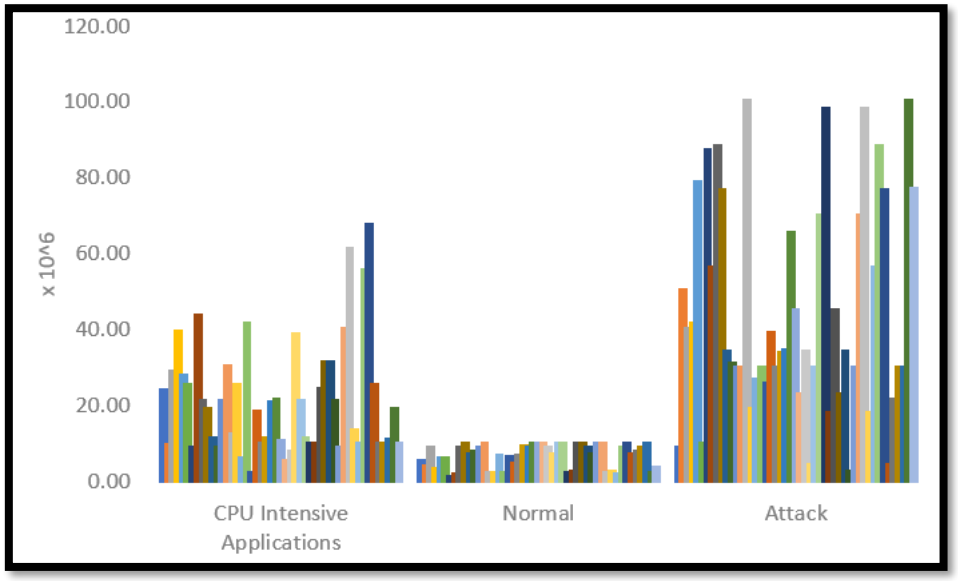


Thread Level

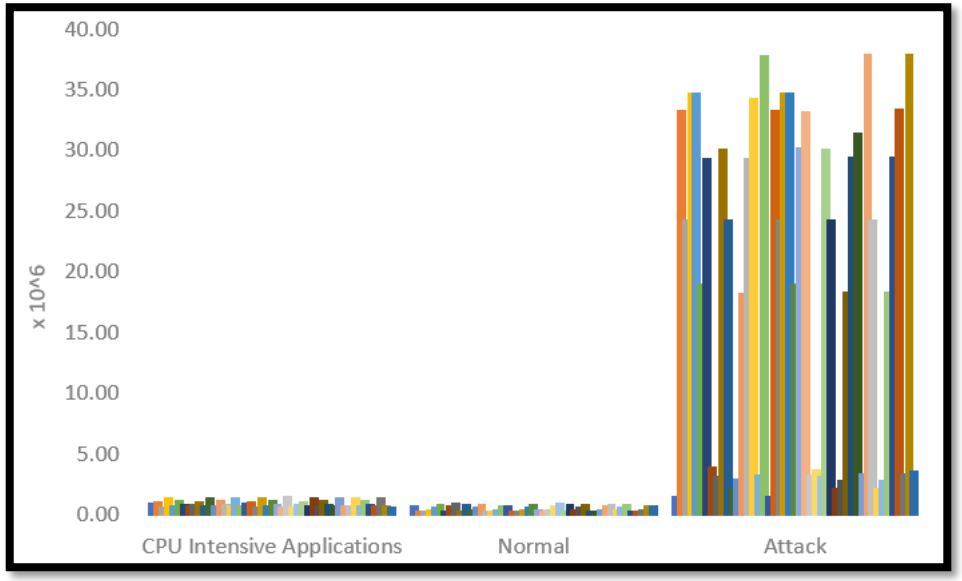


Experiments and Results : DTLB STORE MISS

System level



Thread Level



False positives at System v/s Thread level

EVENT COUNT	GEDIT (SYSTEM LEVEL)	TERMINAL (SYSTEM LEVEL)	SUBLIME TEXT (SYSTEM LEVEL)	ALL THREE EDITORS (THREAD LEVEL)
DTLB LOAD MISS	60	52	64	0
MEMORY LOAD RETIRED L1 MISS	44	37	46	0
MEMORY LOAD RETIRED L_3 HIT	40	38	44	0
DTLB STORE MISS	48	43	48	0

Thread level monitoring shows nil false positives, whereas system level shows 48% false positives

False negatives at System v/s Thread level

- The highest count value in case of High CPU Intensive application scenario, acts as the lower limit for the threshold value
- Anything above that indicates attack
- This derivation becomes very evident in case of thread level monitoring, giving rise to zero false negatives, as against non zero false negatives in system level monitoring.

Conclusion

- Thread level monitoring:
  - Reduces false alarms
  - Reduces performance impact due to unnecessary application of mitigation techniques
  - Reduces monitoring overhead, as only sensitive thread is being monitored and not the entire system.
  - Results of thread level monitoring are more clear to derive threshold, further reducing false negatives

References

1. M. Mushtaq, J. Bricq, M. K. Bhatti, A. Akram, V. Lapotre, G. Gogniat, and P. Benoit, "WHISPER: A tool for run-time detection of side-channel attacks," IEEE Access, vol. 8, pp. 83 871–83 900, 2020.
2. C.-A. A. Pellegrini and S. Carn`a, "Detecting Cache-based Side Channel Attacks using Hardware Performance Counters," no. 1649134, 2018.
3. M. Mushtaq, A. Akram, M. K. Bhatti, M. Chaudhry, V. Lapotre, and G. Gogniat, "Nights-watch: a cache-based side-channel intrusion detector using hardware performance counters," in Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy, HASP@ISCA 2018, Los Angeles, CA, USA, June 02-02, 2018, J. Szefer, W. Shi, and R. B. Lee, Eds. ACM, 2018, pp. 1:1–1:8. [Online]. Available: http://dl.acm.org/citation.cfm?id=3214292
4. J. Cho, T. Kim, T. Kim, and Y. Shin, "Real-time detection on cache side channel attacks using performance counter monitor," in ICTC. IEEE, 2019, pp. 175–177.

THANK YOU

pavitra19231101@iitgoa.ac.in

sharad@iitgoa.ac.in