Exploiting Exclusive System-Level Cache in Apple M-Series SoCs for Enhanced Cache Occupancy Attacks

Tianhong Xu Aidong Adam Ding and Yunsi Fei

Presenter: Tianhong Xu

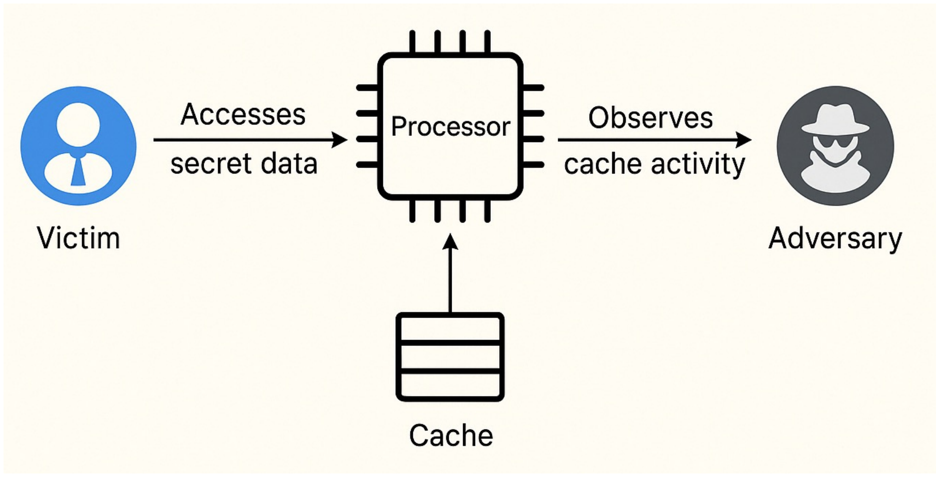
Electrical and Computer Engineering Department

Northeastern University, Boston, Massachusetts

This work was supported in part by National Science Foundation under grants SaTC-1929300 and CNS-1916762.

Cache Attacks: Leaking Secrets via Microarchitectural Side Channels

- Modern CPUs use caches to accelerate memory access
  - A shared resource - between processes, threads, and cores, and even cross CPU and iGPU
- Attackers can infer secret of the victim by observing cache access patterns
- Different types
  - Time-driven
  - Access-driven
  - Cache occupancy



Access-driven vs. Cache Occupancy

- Access-Driven Attacks

  Prime+Probe, Flush+Reload, Evict+Time…

  Leak victim’s address information

  Need high resolution timer

- Cache Occupancy Attacks

  Leak victim’s data amount information

  No need of high resolution timer

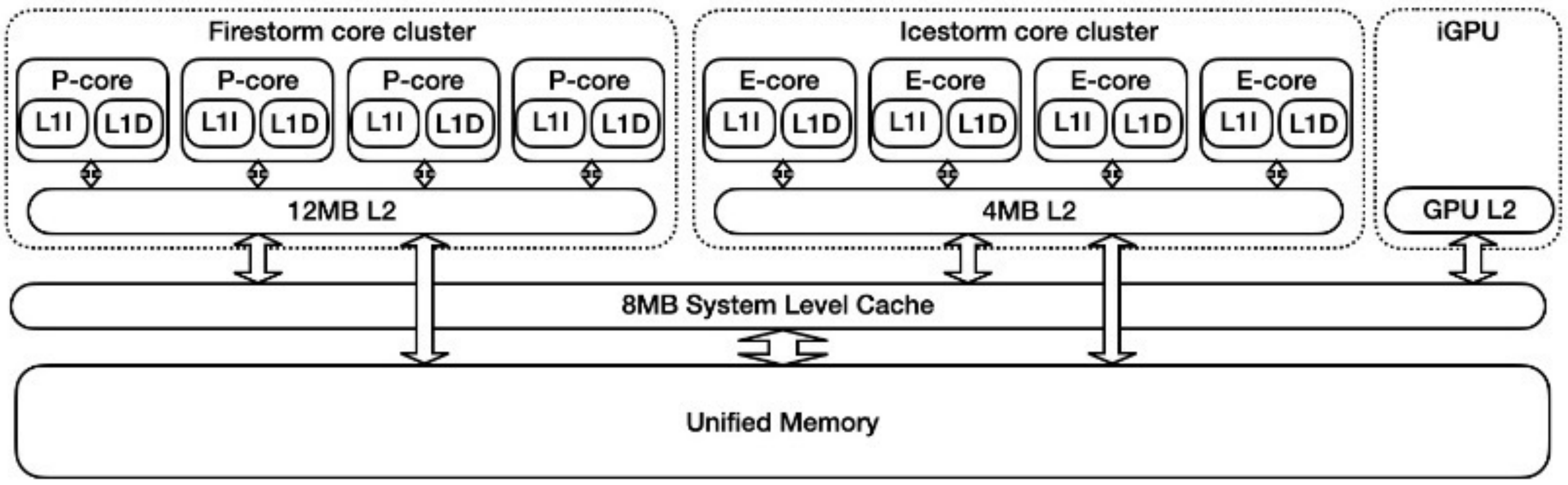
	Access-driven	Cache Occupancy
Granularity	Fine – cache line/set state	Coarse – entire cache state
Information leakage	Address related	Data volume
Measurement requirement	High-resolution timer	Low-resolution timer
Examples		

Overview of this work

- We present a novel suite of cache occupancy attacks
- Targeting the System Level Cache(SLC) of Apple M-series SoCs
  —an exclusive last-level cache GPU and CPU clusters
- Reverse-engineer the SLC’s sharing mechanism
- Propose cache occupancy attacks where adversary can monitor GPU and other CPU cluster activities from their own CPU cluster.

- Three attacks:
  - Website Fingerprinting Attack
  - Cross-Origin Pixel Stealing Attack
  - Screen Display Snooping Attack

Memory structure of APPLE M1



- SLC is shared between CPU and iGPU
- SLC is not inclusive to CPU’s L2 cache, makes it different from typical(intel’s) LLC.

Reverse engineer Apple M1’s SLC

- Inclusiveness Policy:

  A hybrid inclusiveness policy----inclusive with GPU cache but exclusive with the CPU cache.

- Set index mapping:

  A distinctive SLC set index mapping mechanism----Excludes the lowest 13 bits of the physical address for indexing and uses bits from the 14th position and above.

  (Typical cache configurations utilize lower bits of the memory address for cache indexing)

Reverse engineer Apple M1’s SLC

- Inclusiveness Policy:

  A hybrid inclusiveness policy----inclusive with GPU cache but exclusive with the CPU cache.

- Set index mapping:

  A distinctive SLC set index mapping mechanism----Excludes the lowest 13 bits of the physical address for indexing and uses bits from the 14th position and above.

  (Typical cache configurations utilize lower bits of the memory address for cache indexing)

SLC Occupancy Attack

- Cache occupancy attacks
  —attacker create a buffer to fill cache, keep probing it to monitor victim's cache usage
- New techniques for SLC occupancy attack-Filling SLC while bypassing L2 cache

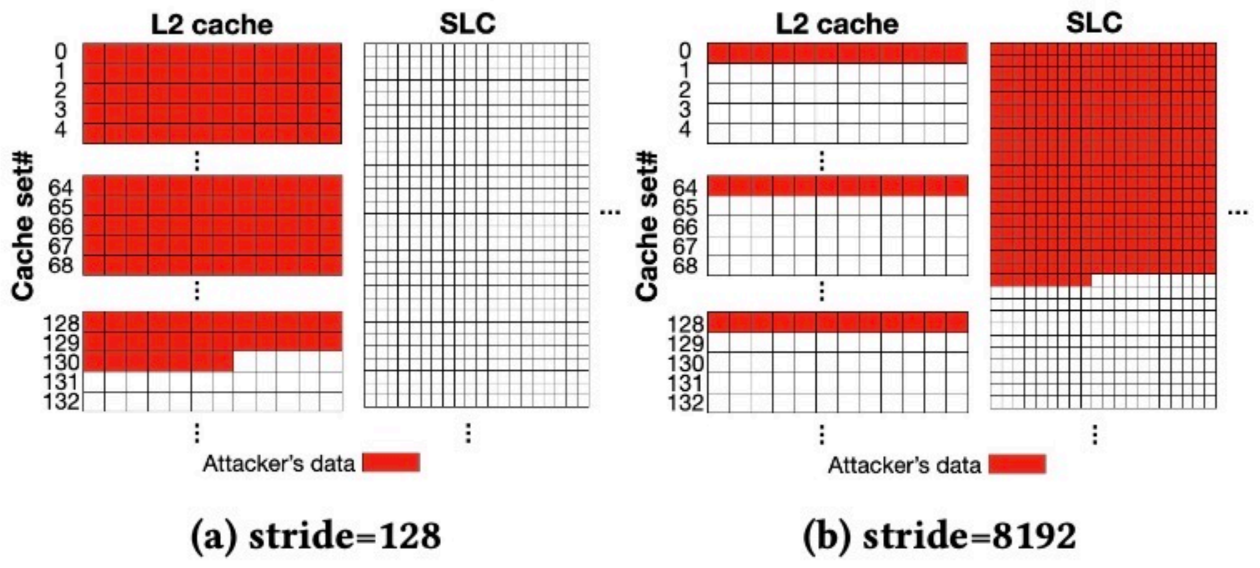


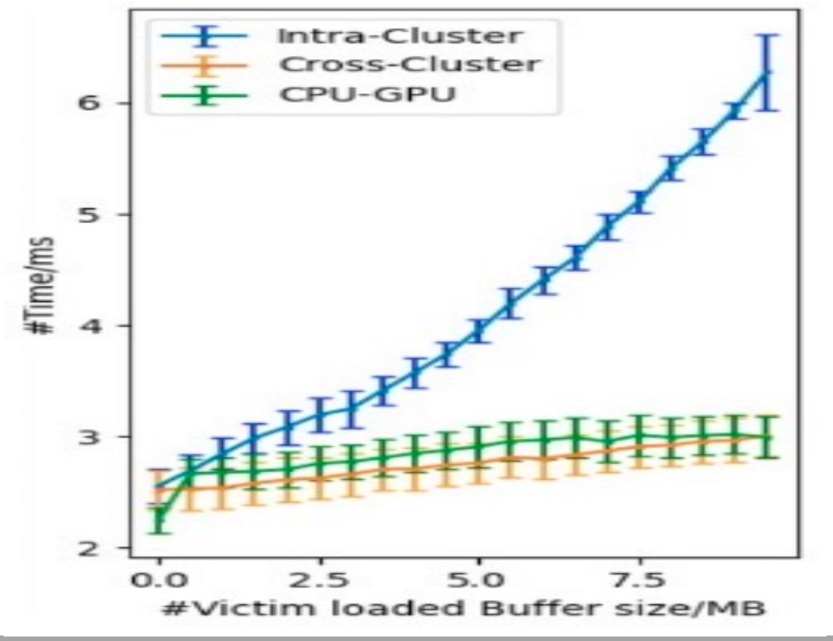
Figure 4: Cache filling with different data structure

SLC Occupancy Attack

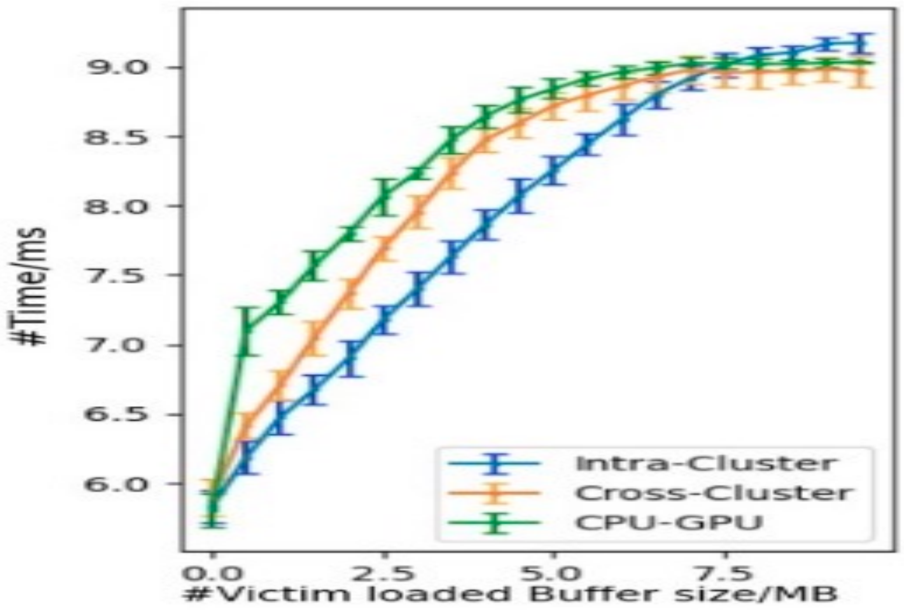
- Comparing with L2 cache occupancy attack

  3 set ups: Intra-cluster, Cross-Cluster and CPU-GPU
- Spy access time vs. victim activities:

L2 Occupancy



SLC Occupancy

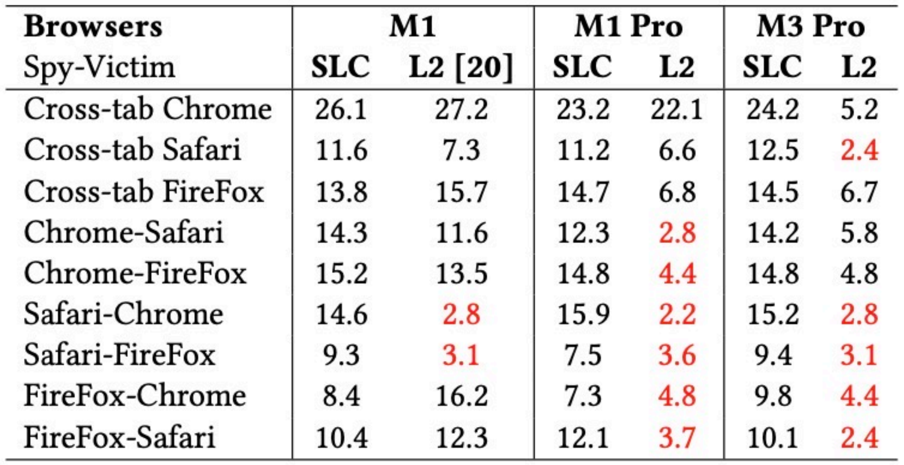


Case study on website environments

- Experimental environment:

  Devices: Apple M1, Apple M1 Pro, Apple M3 Pro
  Browsers: Chrome Firefox, Safari.

- Evaluating the performance of side channel on T-test
  SLC occupancy channel has better performance on Cross Browsers set



Browsers	M1		M1 Pro		M3 Pro	
Spy-Victim	**SLC**	**L2 [20]**	**SLC**	**L2**	**SLC**	**L2**
Cross-tab Chrome	26.1	27.2	23.2	22.1	24.2	5.2
Cross-tab Safari	11.6	7.3	11.2	6.6	12.5	2.4
Cross-tab FireFox	13.8	15.7	14.7	6.8	14.5	6.7
Chrome-Safari	14.3	11.6	12.3	2.8	14.2	5.8
Chrome-FireFox	15.2	13.5	14.8	4.4	14.8	4.8
Safari-Chrome	14.6	2.8	15.9	2.2	15.2	2.8
Safari-FireFox	9.3	3.1	7.5	3.6	9.4	3.1
FireFox-Chrome	8.4	16.2	7.3	4.8	9.8	4.4
FireFox-Safari	10.4	12.3	12.1	3.7	10.1	2.4

Table 1: Benchmark results/T-test score

Case study on website environments

- Website fingerprint attack:

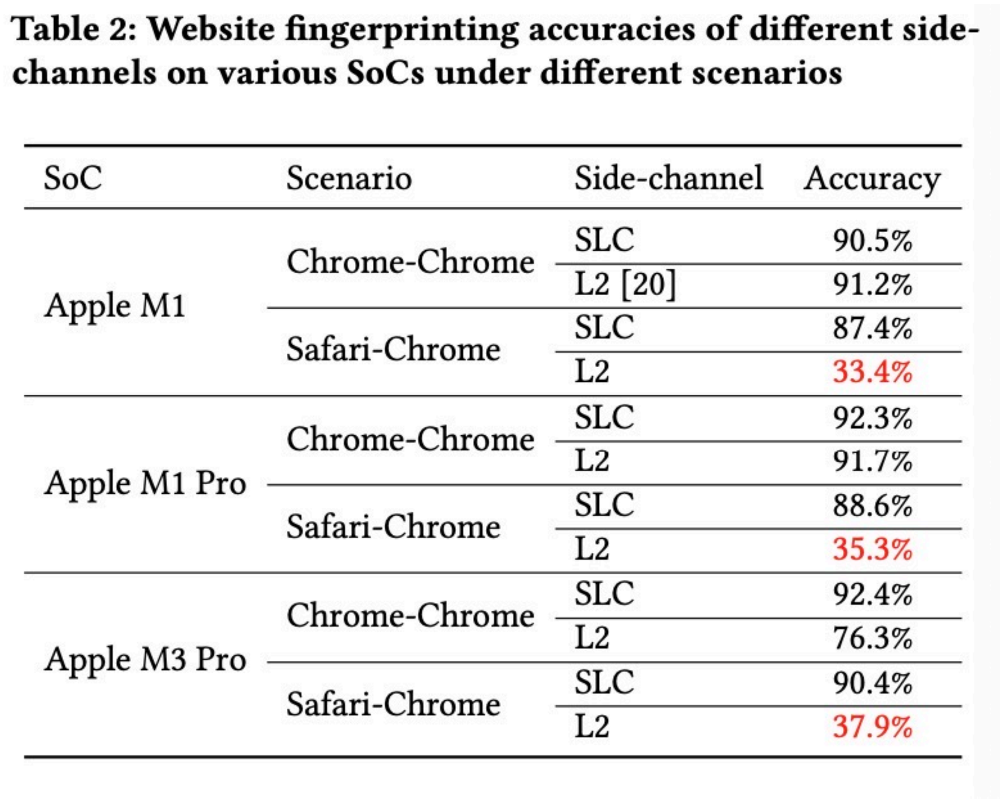
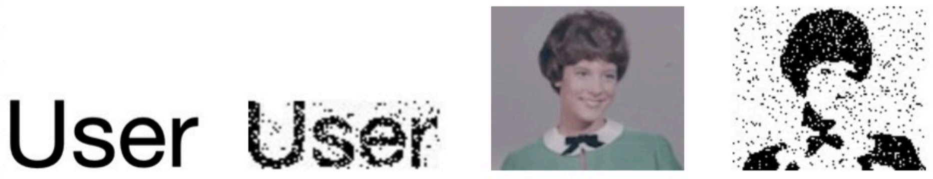


Table 2: Website fingerprinting accuracies of different side-channels on various SoCs under different scenarios

SoC	Scenario	Side-channel	Accuracy
Apple M1	Chrome-Chrome	SLC	90.5%
		L2 [20]	91.2%
	Safari-Chrome	SLC	87.4%
		L2	33.4%
Apple M1 Pro	Chrome-Chrome	SLC	92.3%
		L2	91.7%
	Safari-Chrome	SLC	88.6%
		L2	35.3%
Apple M3 Pro	Chrome-Chrome	SLC	92.4%
		L2	76.3%
	Safari-Chrome	SLC	90.4%
		L2	37.9%

- Pixel Stealing Attack:



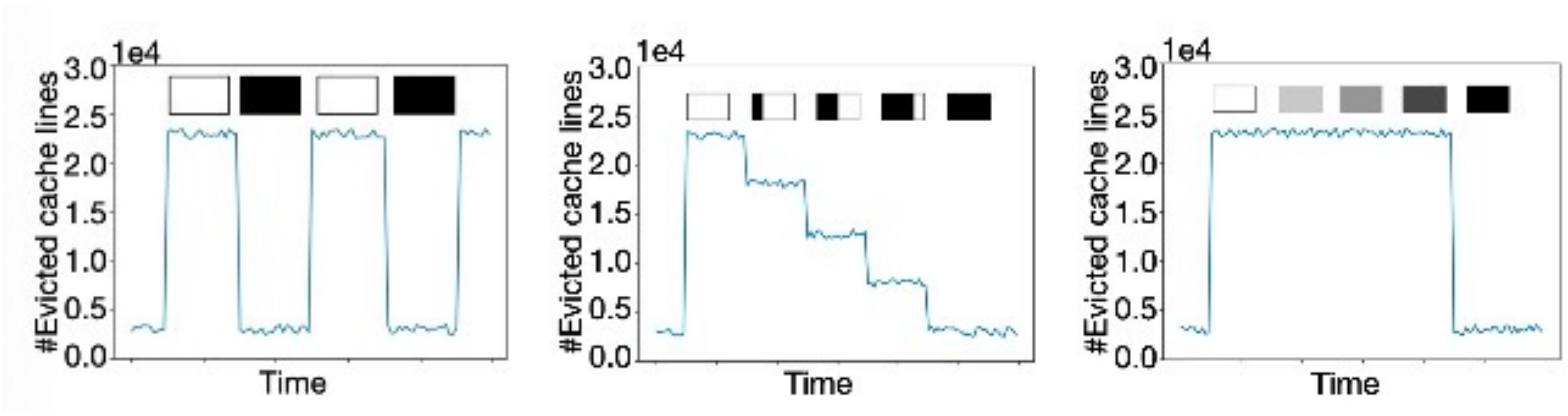
(a) Original text **(b) Retrieved text** **(c) Original image** **(d) Retrieved image**

Figure 8: Pixel stealing attack

Screen-capturing Attack

- Using SLC occupancy channel to monitor Screen showing process(Graphic rendering)

- New finding: Data usage of Screen showing is related to the screen’s contents while the screen if full of big blocks of Black/White areas.



Screen-capturing Attack

- New attack framework—Precisely monitoring frame activities

  SLC occupancy probing aligned with Vsync signal
  Calculate the difference between the two probing

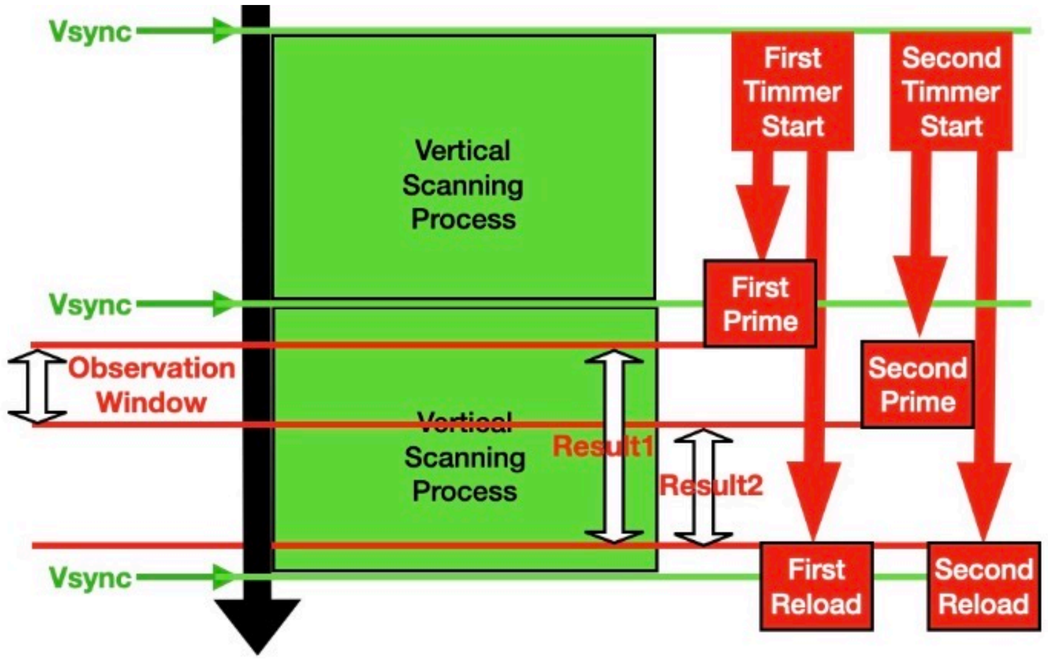


Figure 10: Precisely monitoring frame activities

Screen-capturing Attack

- Collecting a Trace for a screen frame (While the image on the screen doesn't change)
- Observe 28 short epochs during the frame rendering process, which are correspond to 28 evenly divided regions of the screen, vertically from top to bottom.

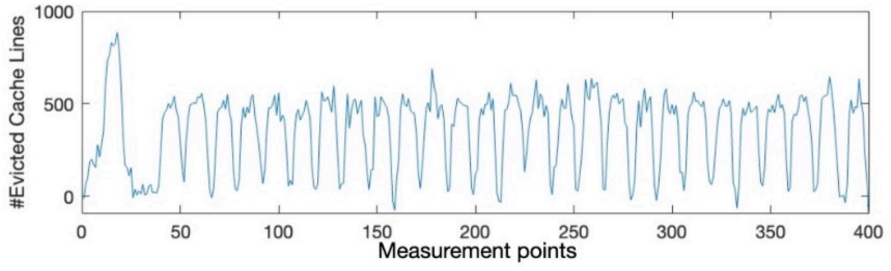


Figure 11: Trace of a full frame

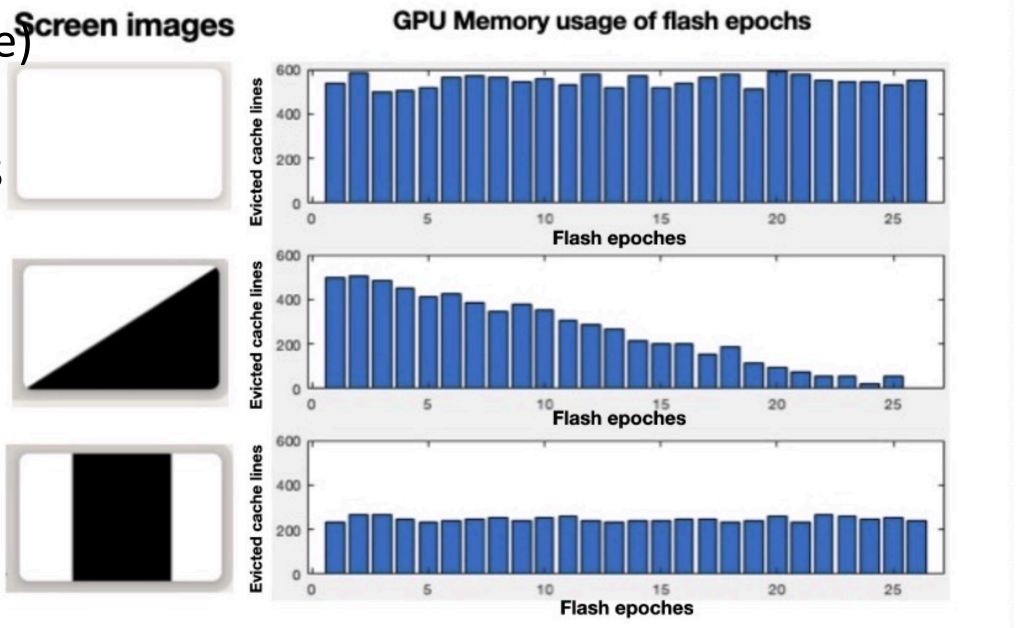


Figure 12: Correlation between a trace of flash points and the screen bars (with varying zero pixel values)

Screen-capturing Attack

- Attack example:

  Extracting print numbers

  Extracting Barcodes

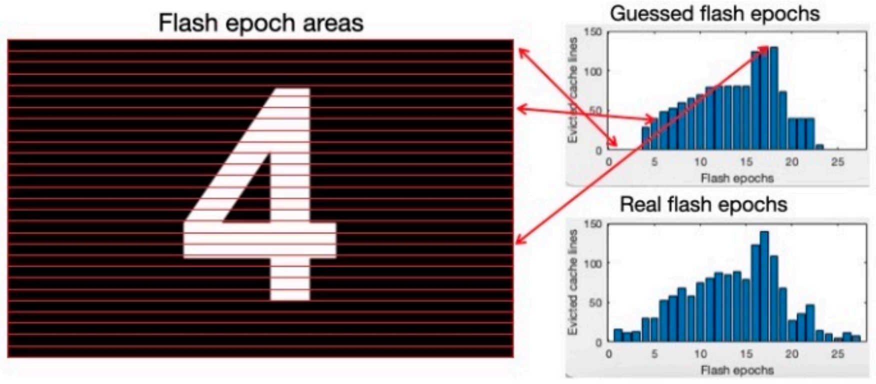
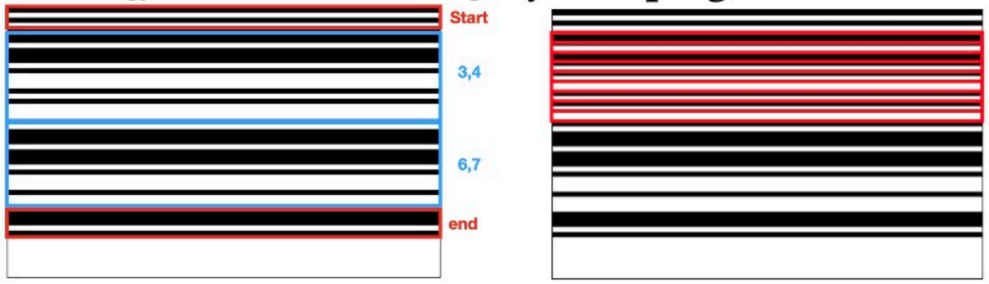


Figure 13: Screen display snooping attack method



(a) Characters **(b) Flash epoch segments**

Figure 14: ITF barcode

Q&A