

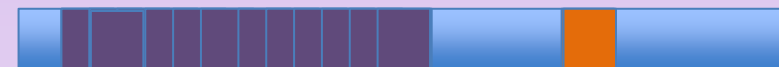
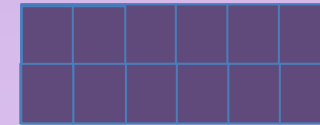
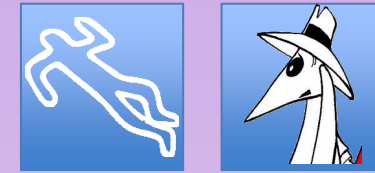
# LLC Attacks

# Topics

- Prime+Probe on the last level cache
  - Plus variants
- Cache Occupancy attacks

# The Prime+Probe Attack [OST06]

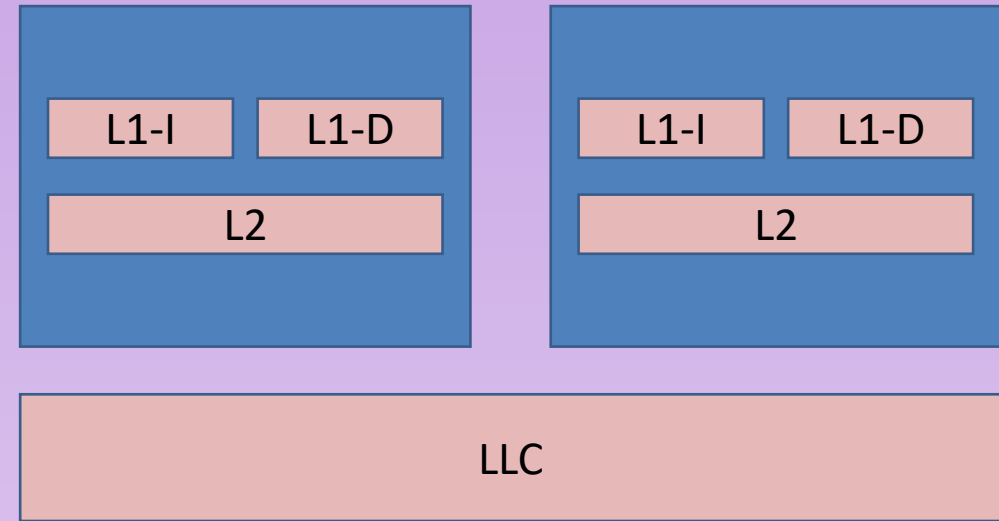
- Allocate a cache-sized memory buffer
- *Prime*: fills the cache with the contents of the buffer
- *Probe*: measure the time to access each cache set
  - Slow access indicates victim access to the set
- The probe phase primes the cache for the next round



Memory

# Cache Hierarchy

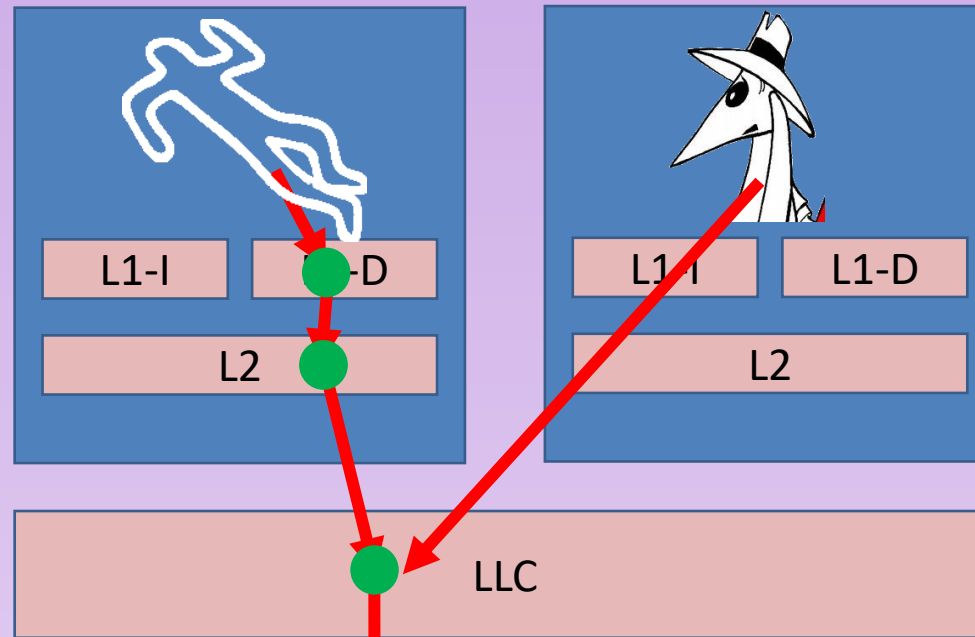
- Multiple caches serve different trade offs between speed and size



Level	Sets	Size	Latency
L1	64	32-48 KB	4
L2	256	256-2048 KB	7
LLC (L3)	4-128K	3-64 MB	~30

Why target other caches?


# Cross Core Attacks



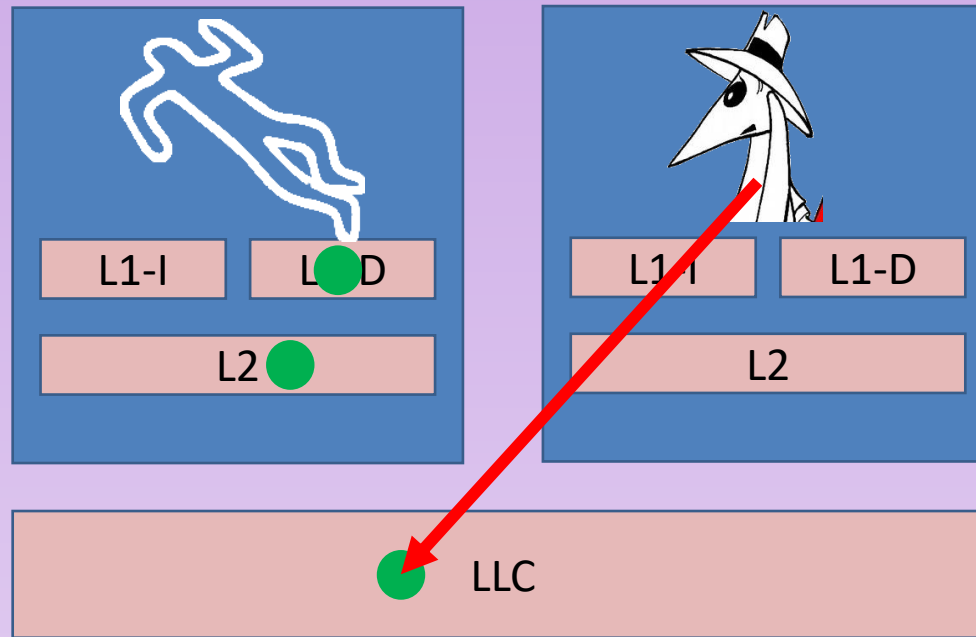
How can the spy evict victim data?

# Cache Inclusiveness



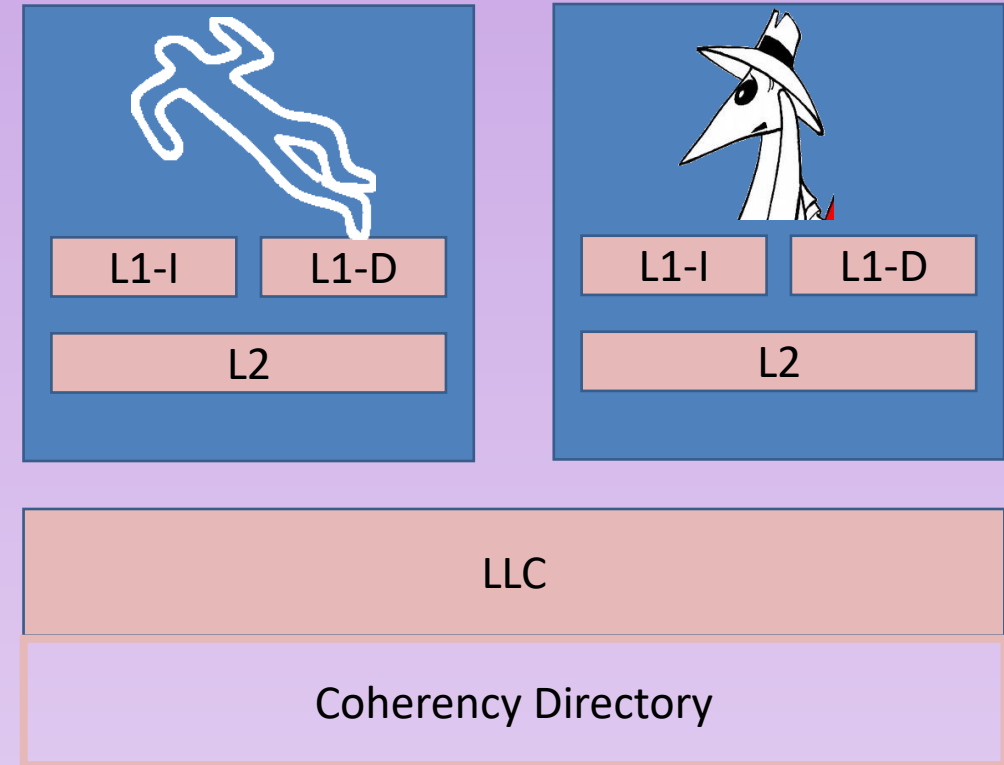
- Inclusive cache: stores copies of all data in higher level caches 
- Exclusive cache: does not store data cached in higher level caches
- Non-inclusive cache: a cache that is not strictly inclusive. (Can be exclusive.)
- With inclusive LLCs, evicting from the LLC also evicts from all core-private caches

# Attacking inclusive caches



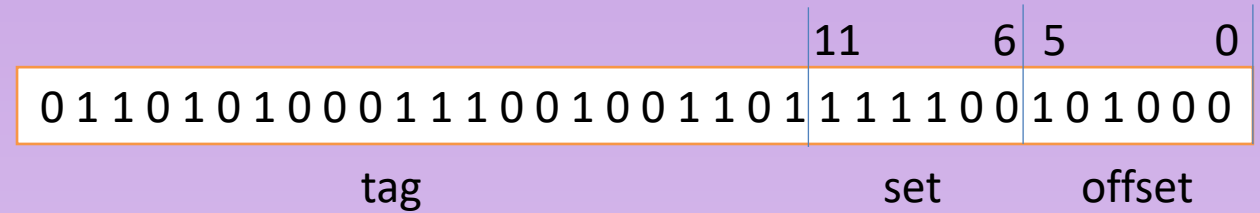
# Attacking non-inclusive caches

- Depends on coherency protocol
  - Coherency directory – can attack like a cache (Yan et al. IEEE SP 2019)
- Snoop filters: convince the victim...
- Autolock: prevents LLC eviction of lines stored in other core's L1/L2 (Green et al. USENIX Security 2017)



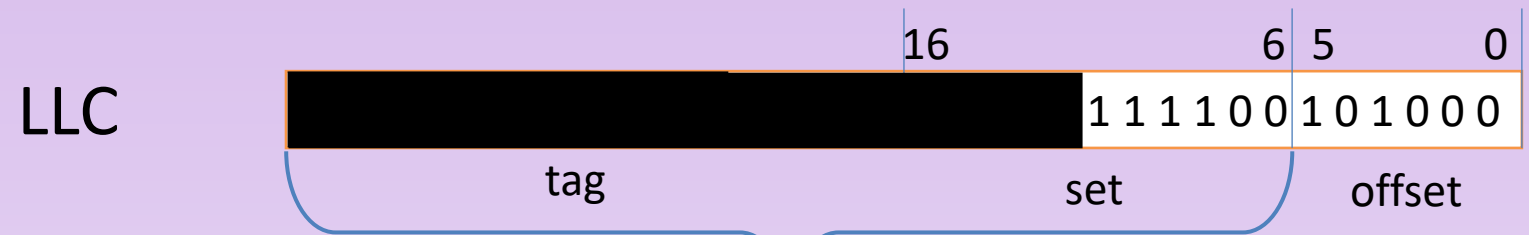
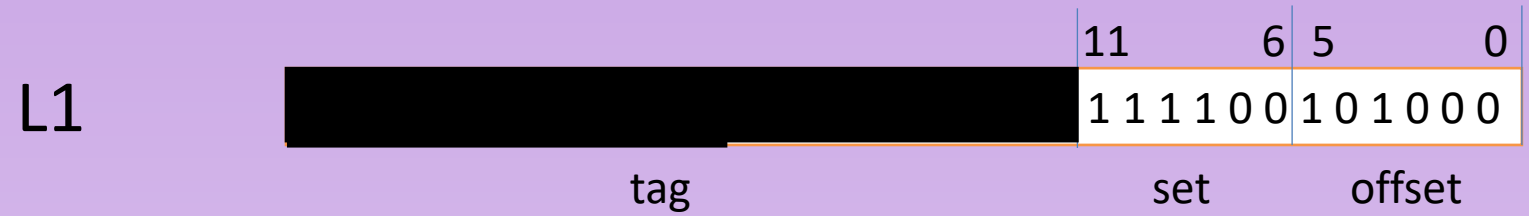


# Mapping Addresses to Cache Sets



We can use the 'set' bits in the address

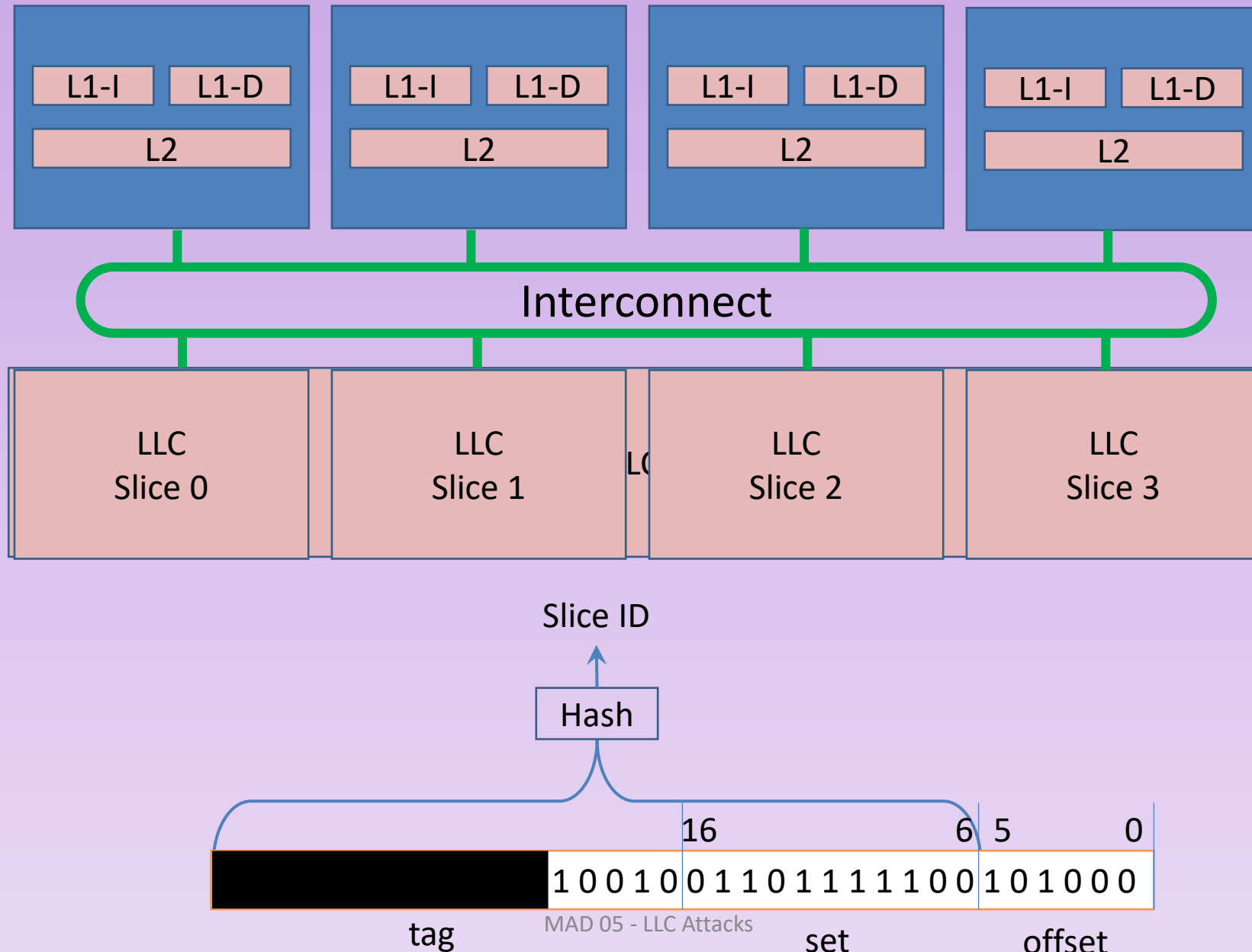
# Mapping Addresses to Cache Sets



Hash

Slice ID

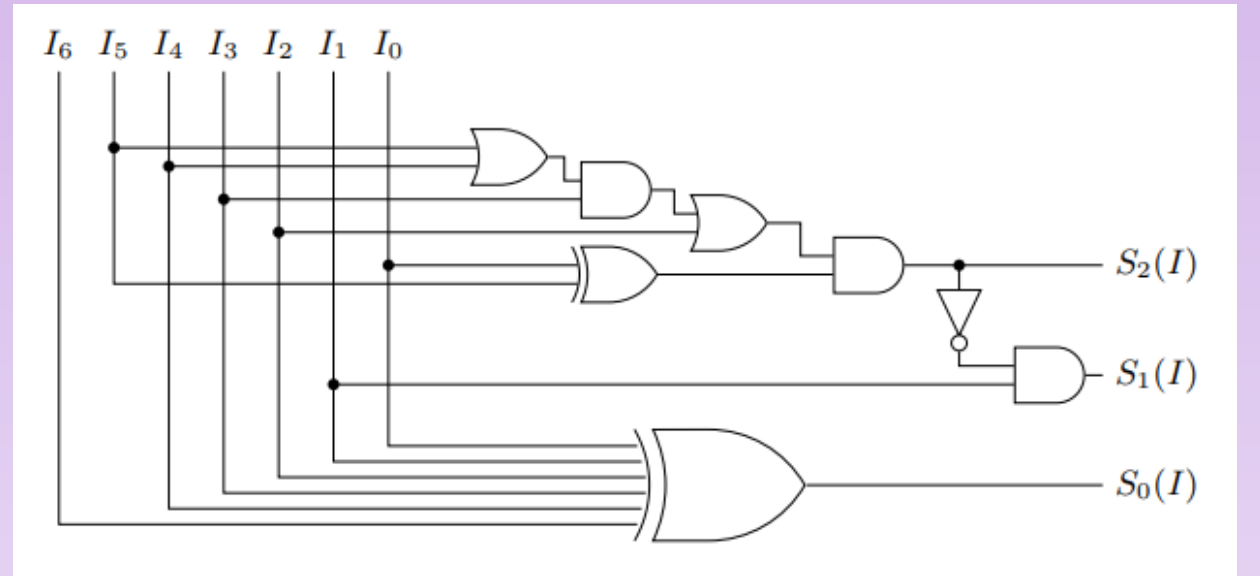
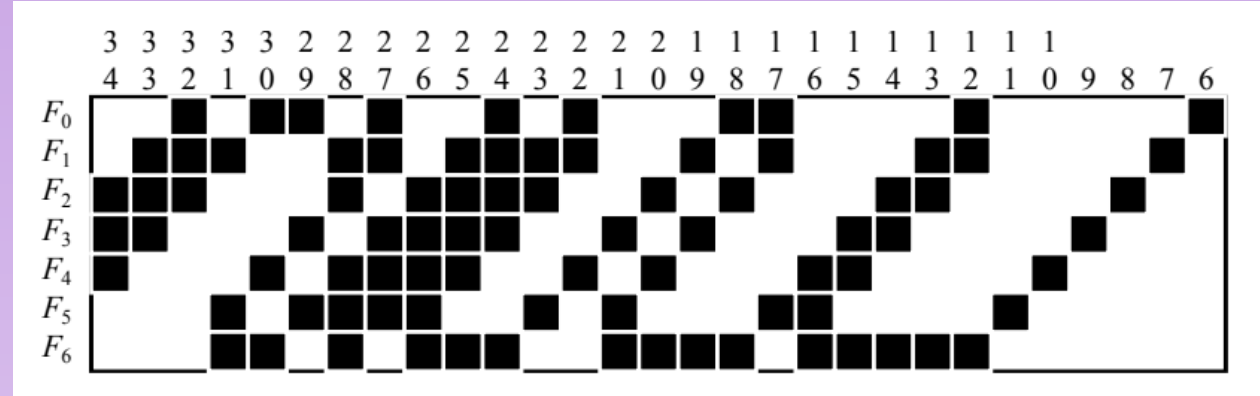
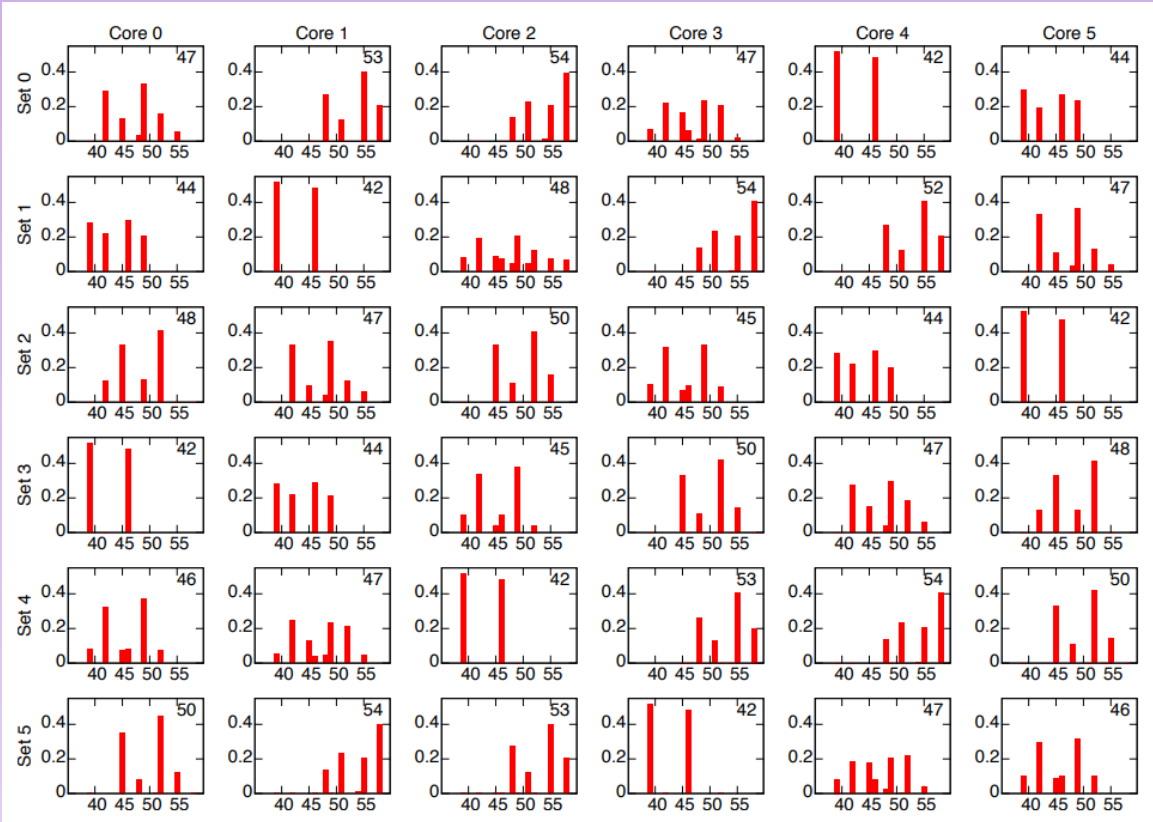
# Cache Slices



# Reverse Engineering – perf counters (Maurice et al. RAID 2015)

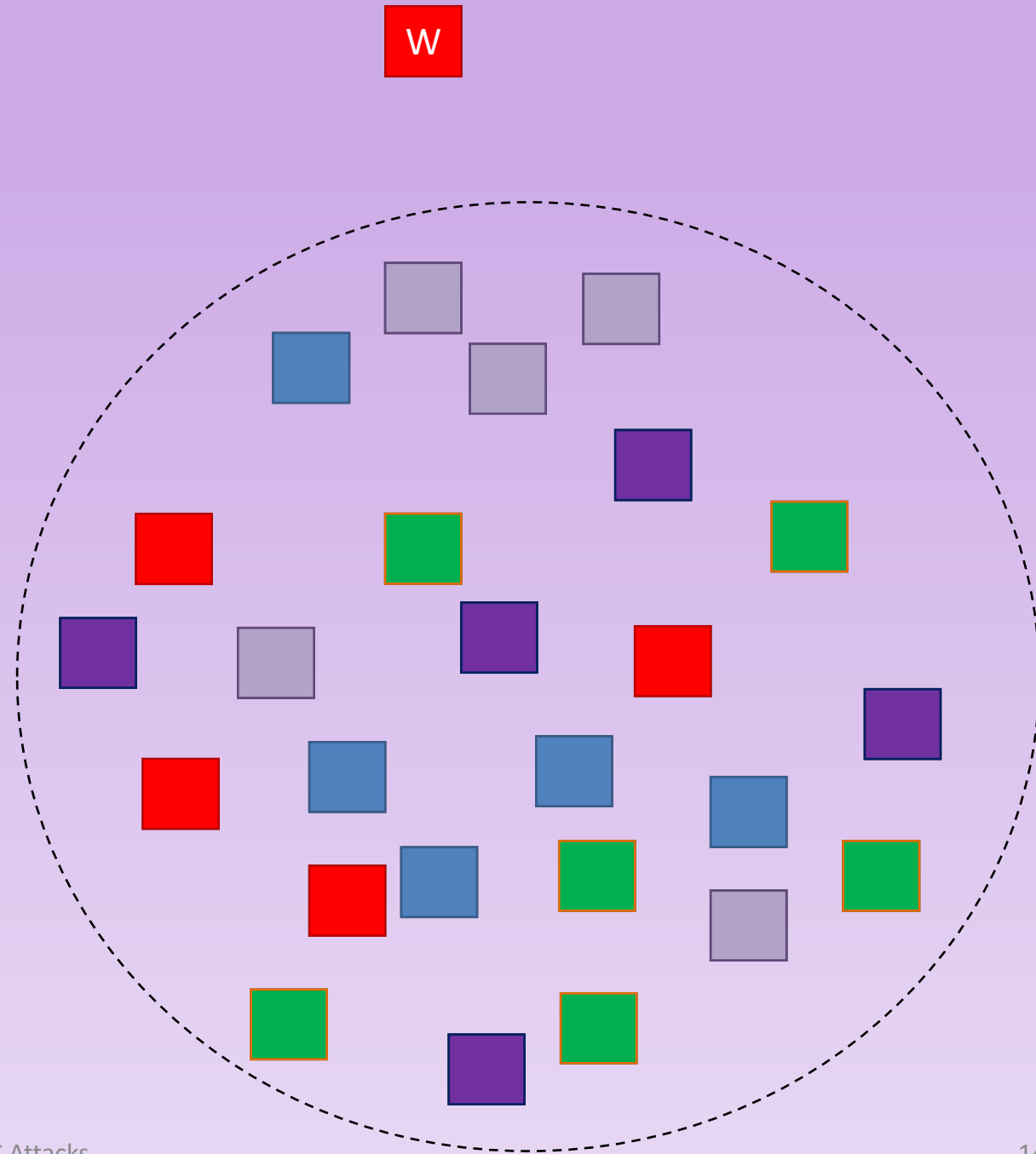
		Address Bit																															
		3 7	3 6	3 5	3 4	3 3	3 2	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6
2 cores	$o_0$						⊕		⊕		⊕	⊕	⊕	⊕	⊕		⊕		⊕		⊕	⊕	⊕		⊕		⊕		⊕				⊕
4 cores	$o_0$					⊕	⊕		⊕		⊕	⊕	⊕	⊕	⊕		⊕		⊕		⊕	⊕	⊕		⊕		⊕		⊕				⊕
	$o_1$				⊕	⊕		⊕		⊕	⊕		⊕		⊕	⊕	⊕	⊕	⊕		⊕		⊕		⊕		⊕		⊕			⊕	
8 cores	$o_0$		⊕	⊕		⊕	⊕		⊕		⊕	⊕	⊕	⊕	⊕		⊕		⊕		⊕	⊕	⊕		⊕		⊕		⊕				⊕
	$o_1$	⊕		⊕	⊕	⊕		⊕		⊕	⊕		⊕		⊕	⊕	⊕	⊕	⊕		⊕		⊕		⊕		⊕				⊕		
	$o_2$	⊕	⊕	⊕	⊕			⊕	⊕			⊕	⊕			⊕	⊕			⊕			⊕			⊕	⊕				⊕		

# Non-linear functions



# Finding Eviction Sets

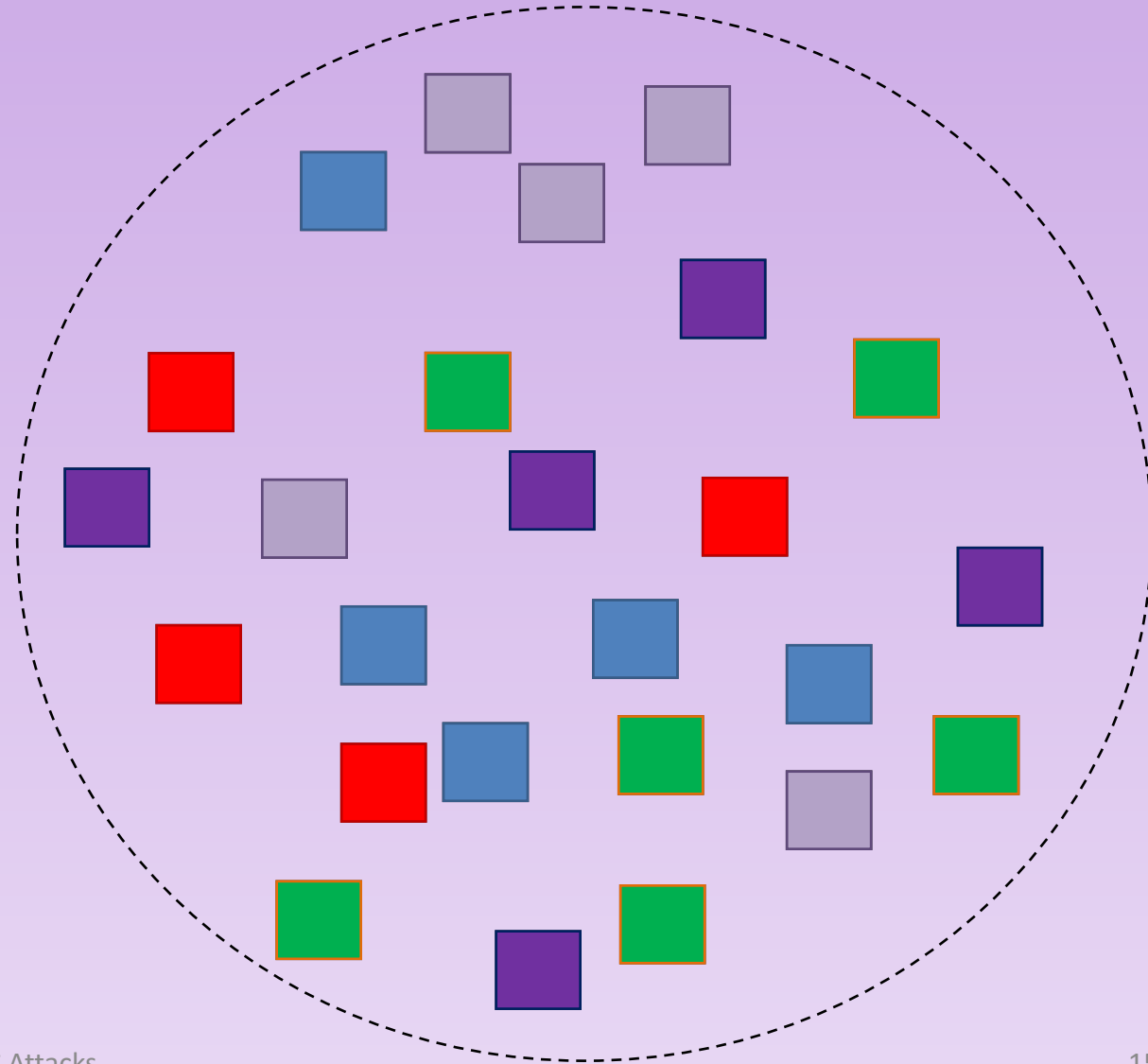
- Problem: Find a **minimal** set of addresses that evicts a given *witness*



# Finding Eviction Sets (Liu et al. IEEE SP 2015)



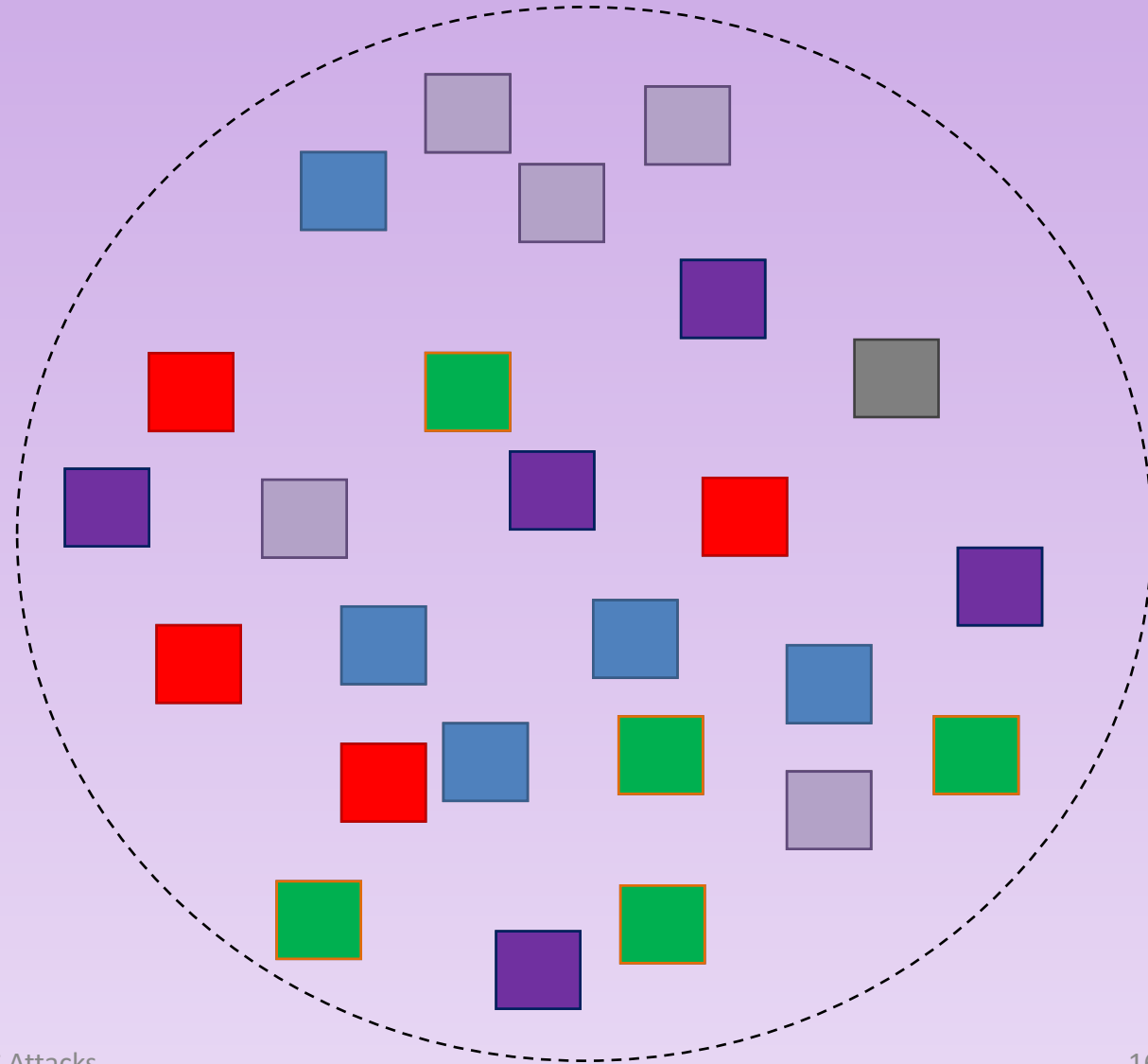
- Problem: Find a **minimal** set of addresses that evicts a given *witness*
- Testing eviction:
  - Access witness
  - Access candidate set
  - Check if witness is in cache



# Finding Eviction Sets (Liu et al. IEEE SP 2015)

W

- Problem: Find a **minimal** set of addresses that evicts a given *witness*
- Testing eviction:
  - Access witness
  - Access candidate set
  - Check if witness is in cache
- Reduce candidate set
  - Remove a candidate
  - Test eviction

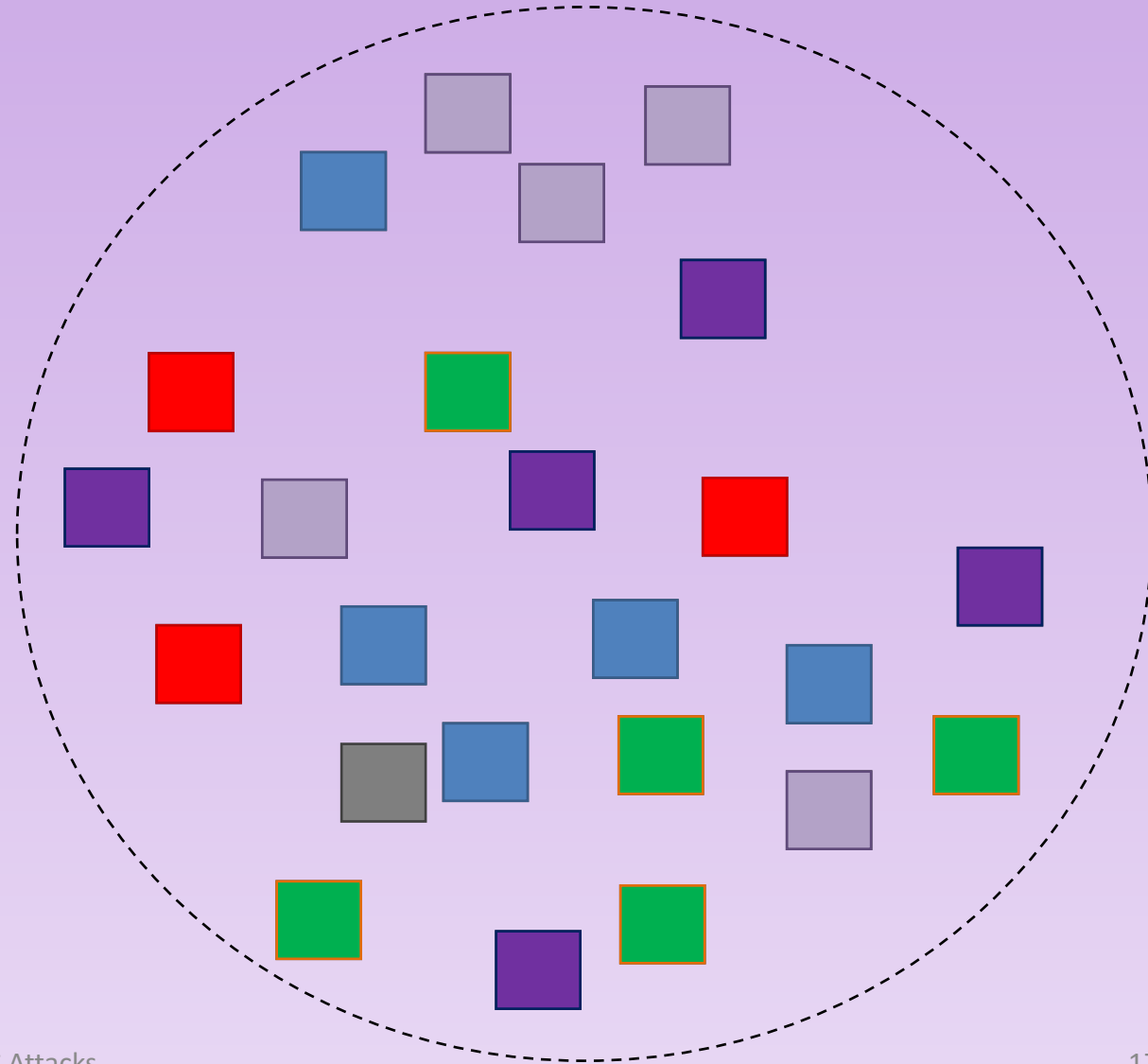




# Finding Eviction Sets (Liu et al. IEEE SP 2015)

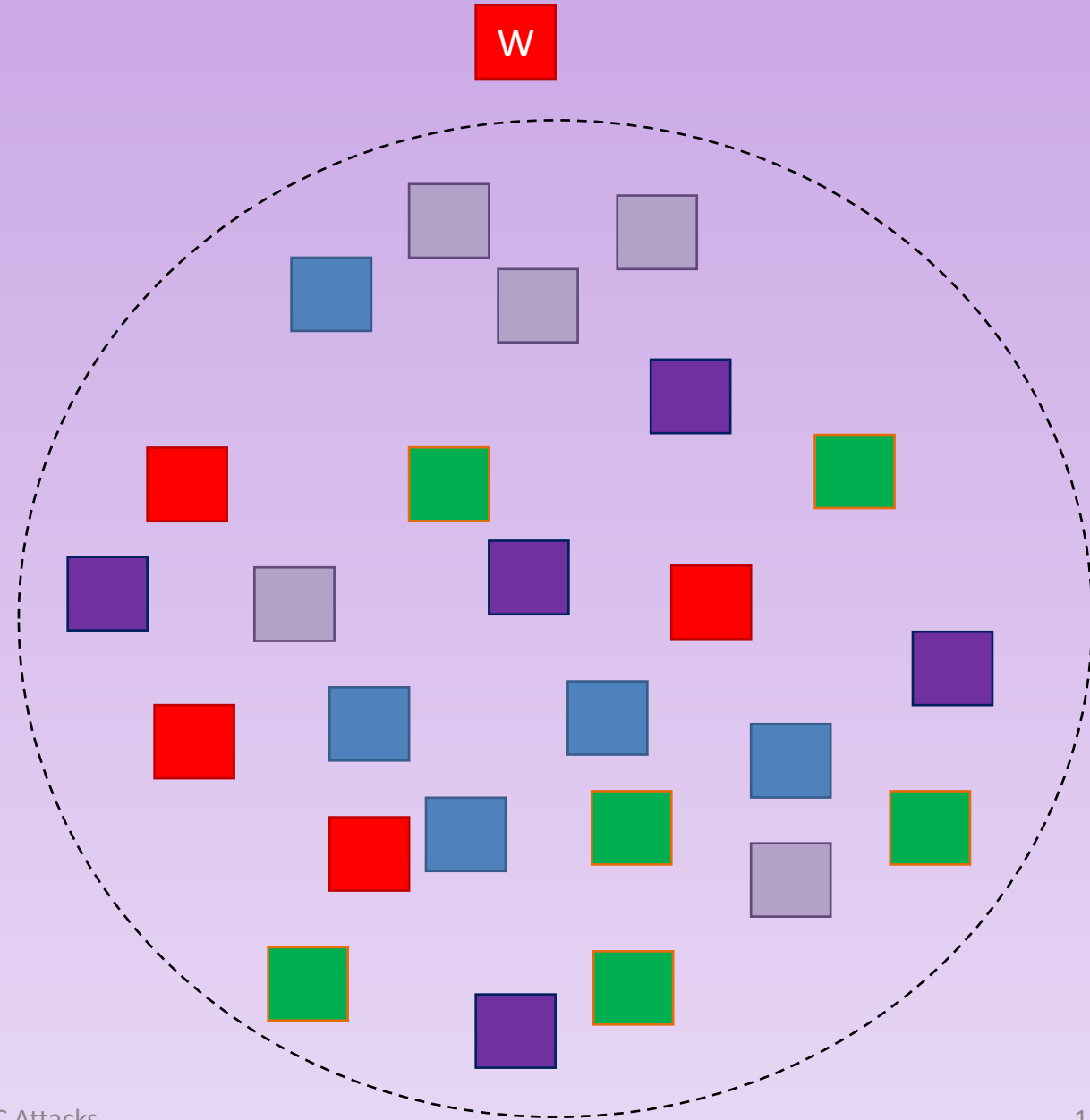
W

- Problem: Find a **minimal** set of addresses that evicts a given *witness*
- Testing eviction:
  - Access witness
  - Access candidate set
  - Check if witness is in cache
- Reduce candidate set
  - Remove a candidate
  - Test eviction
  - If not evicted, return candidate
  - Repeat until no candidates can be removed



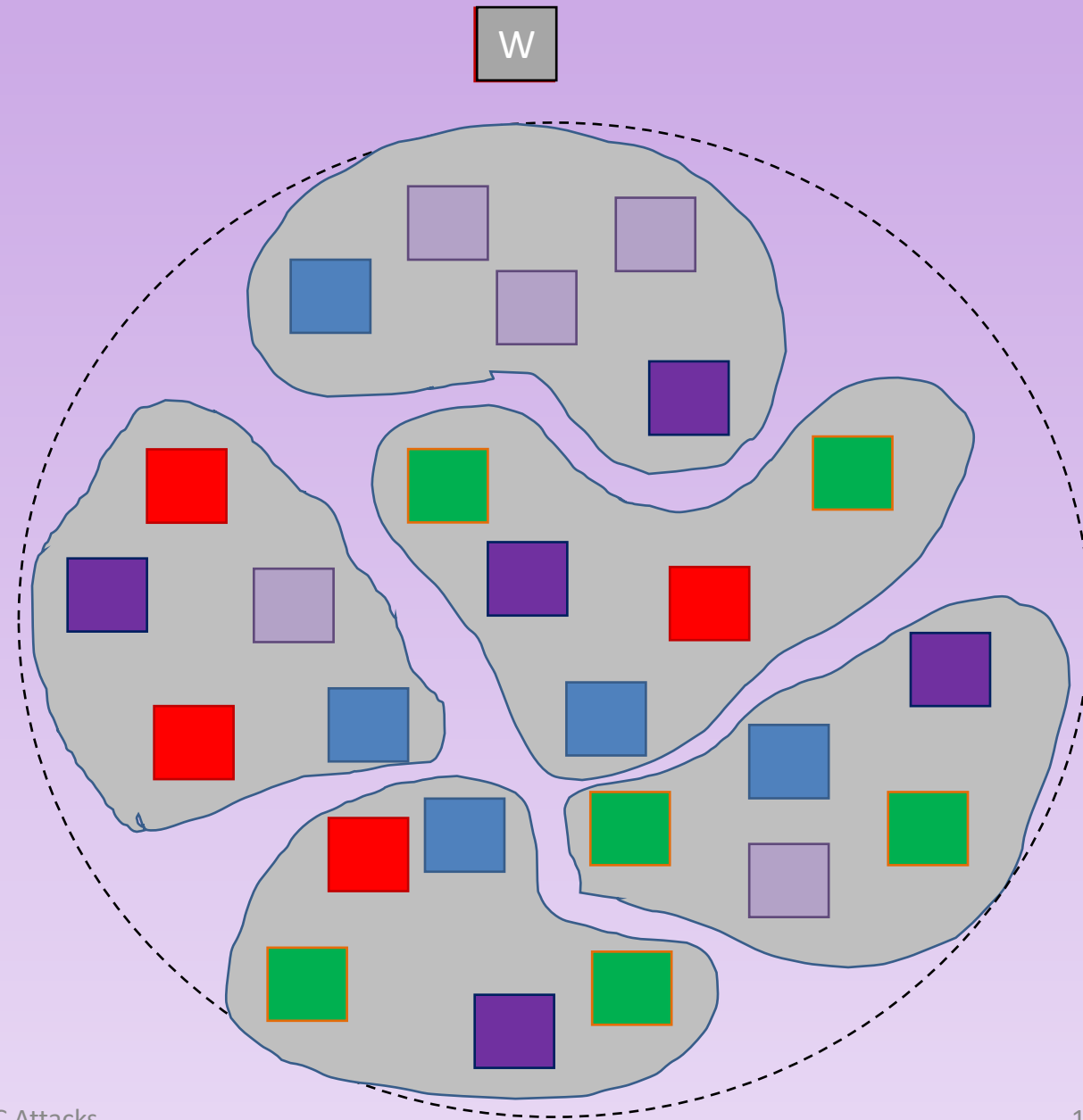
# Improving search time (Vila et al. IEEE SP 2019)

- Quadratic search takes a long time



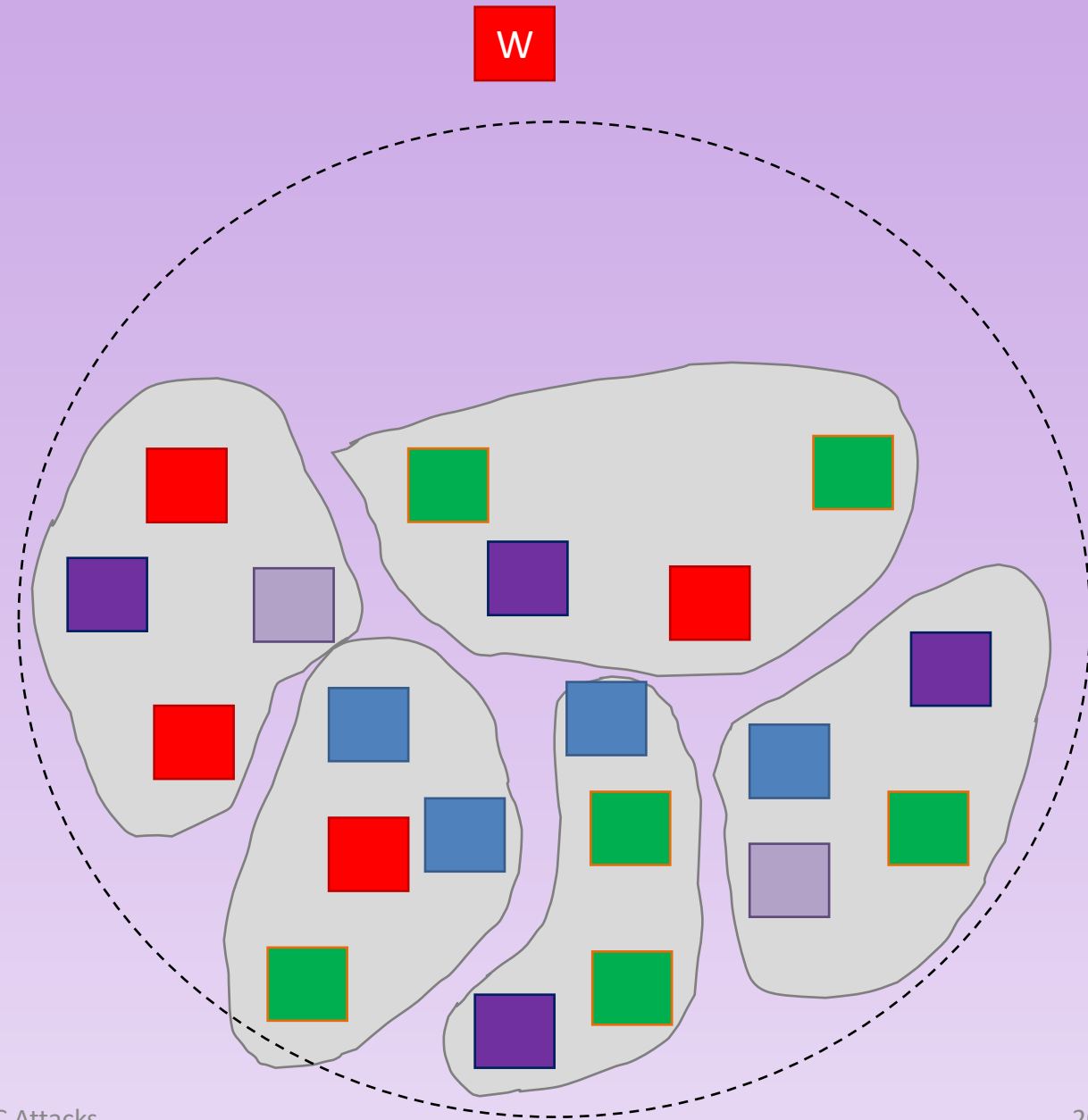
# Improving search time (Vila et al. IEEE SP 2019)

- Quadratic search takes a long time
- Linear-time eviction set creation
  - Divide candidates into  $n+1$  partition
  - Remove one of the partitions, do an eviction test with the remaining  $n$
  - Repeat for other partitions
  - At least one of the choices of  $n$  partitions must succeed in evicting the witness



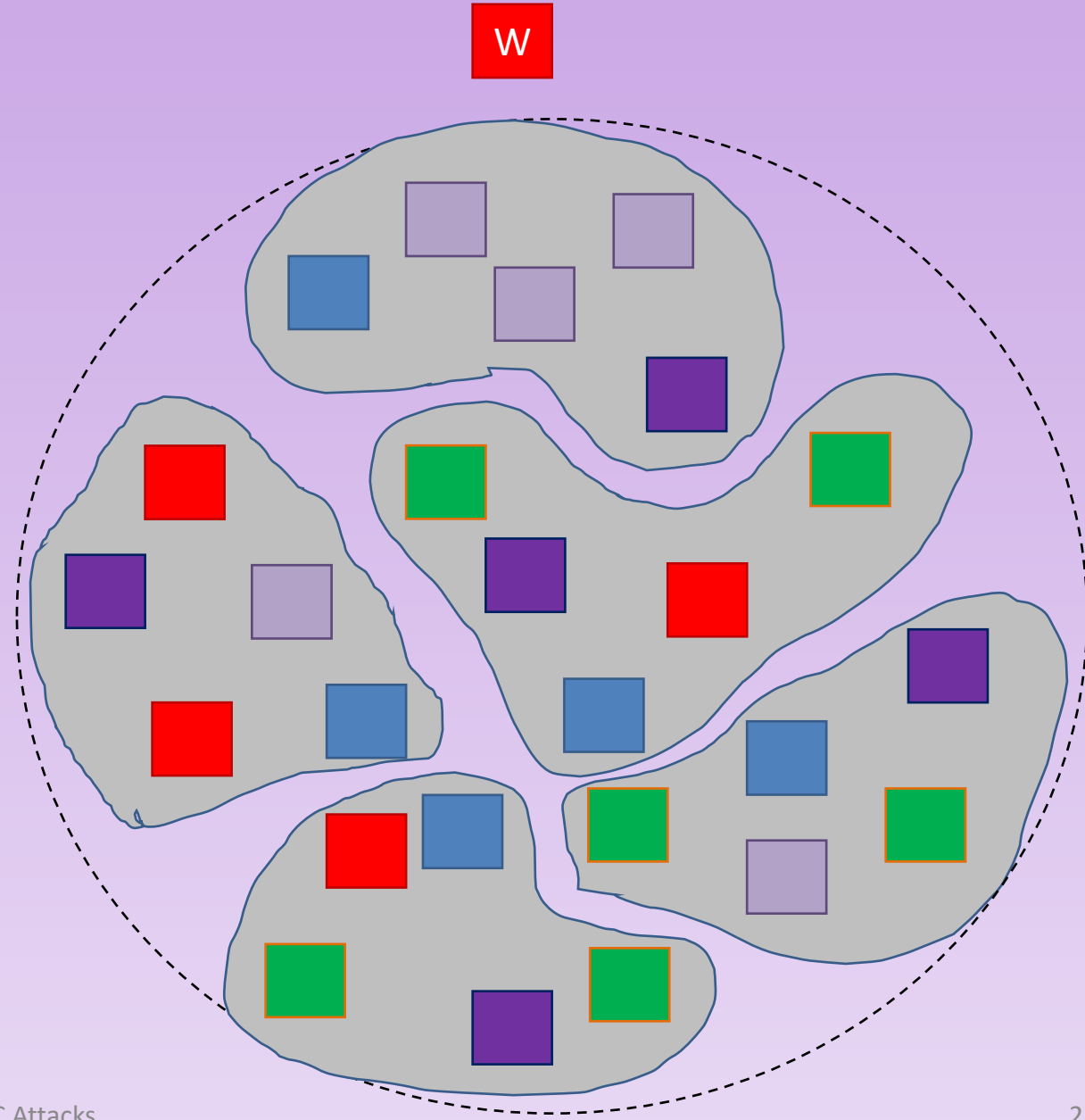
# Improving search time (Vila et al. IEEE SP 2019)

- Quadratic search takes a long time
- Linear-time eviction set creation
  - Divide candidates into  $n+1$  partition
  - Remove one of the partitions, do an eviction test with the remaining  $n$
  - Repeat for other partitions
  - At least one of the choices of  $n$  partitions must succeed in evicting the witness
- Repeat until only  $n$  candidates remain



# Complexity

- Starting with  $X$  candidates
- Do about  $(n + 1) \frac{nX}{n+1} = nX$  memory accesses to eliminate  $\frac{X}{n+1}$  candidates
- $$f(X) = nX + f\left(\frac{nX}{n+1}\right)$$
$$= nX + \frac{n^2X}{n+1} + f\left(\frac{n^2X}{(n+1)^2}\right)$$
$$\approx n(n+1)X = O(X)$$



# Prime+Abort (Disselkoen et al. USENIX Security 2017)

- LLC Prime+Probe has a low temporal resolution ~2000 cycles
- Exploit TSX transactions.
  - Transaction code executes atomically or not at all
  - Aborts when detecting a conflict
  - Uses the caches to track modified/read memory
- Prime+Abort:
  - Begin a transaction
  - Access an eviction set
  - On abort, check reason



```
xbegin <fallback>
code
code
...
code
xend
```

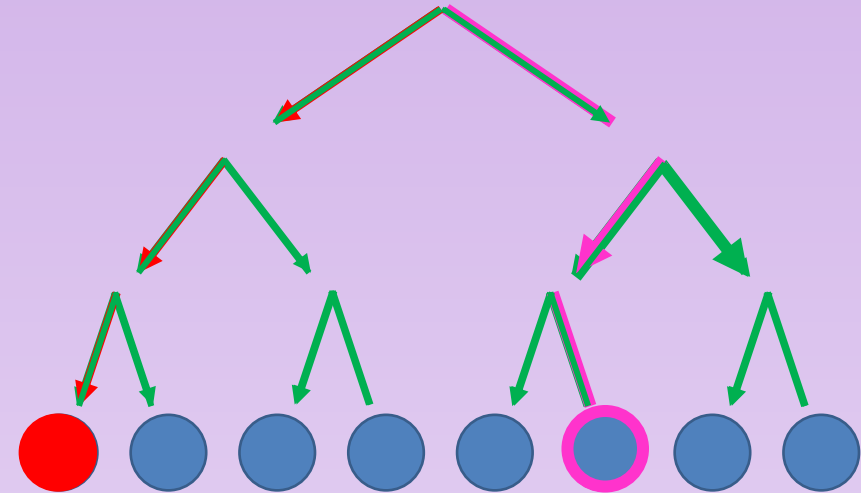
```
<fallback>:
```

# Prime+Scope (Purnal et al. CCS 2021)

- Idea: Exploit replacement policy and LLC/directory inclusiveness to improve attacks.

# A short digression: Tree PLRU

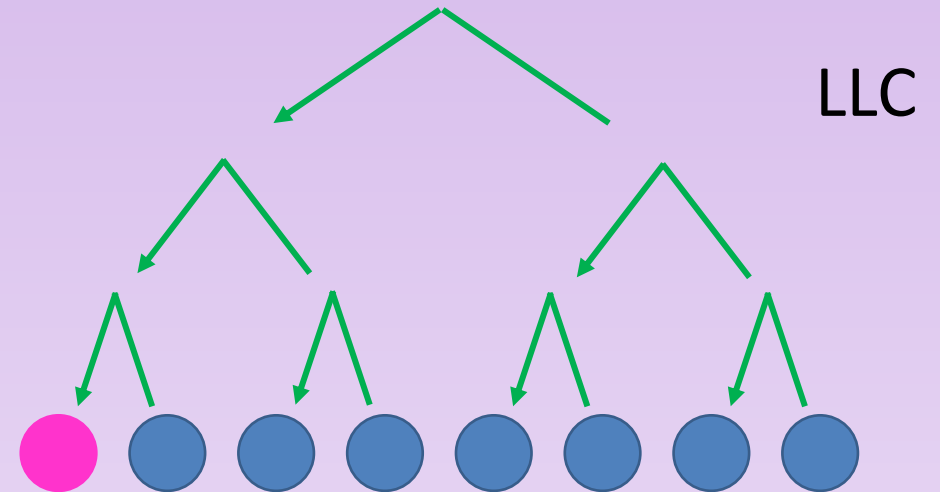
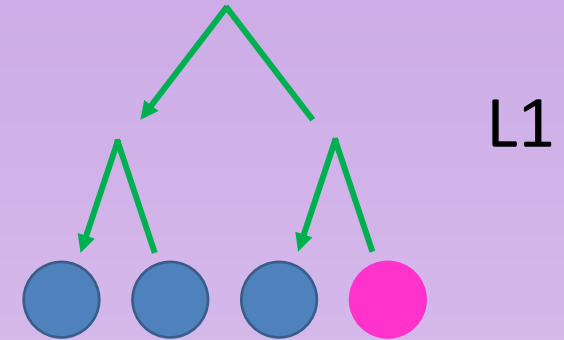
- Age data stored in nodes of a tree
- Each node points to the older child
- Accessing an entry switches older child in path
- Replacement follows oldest children chain
  - And updates path





# Prime+Scope (Purnal et al. CCS 2021)

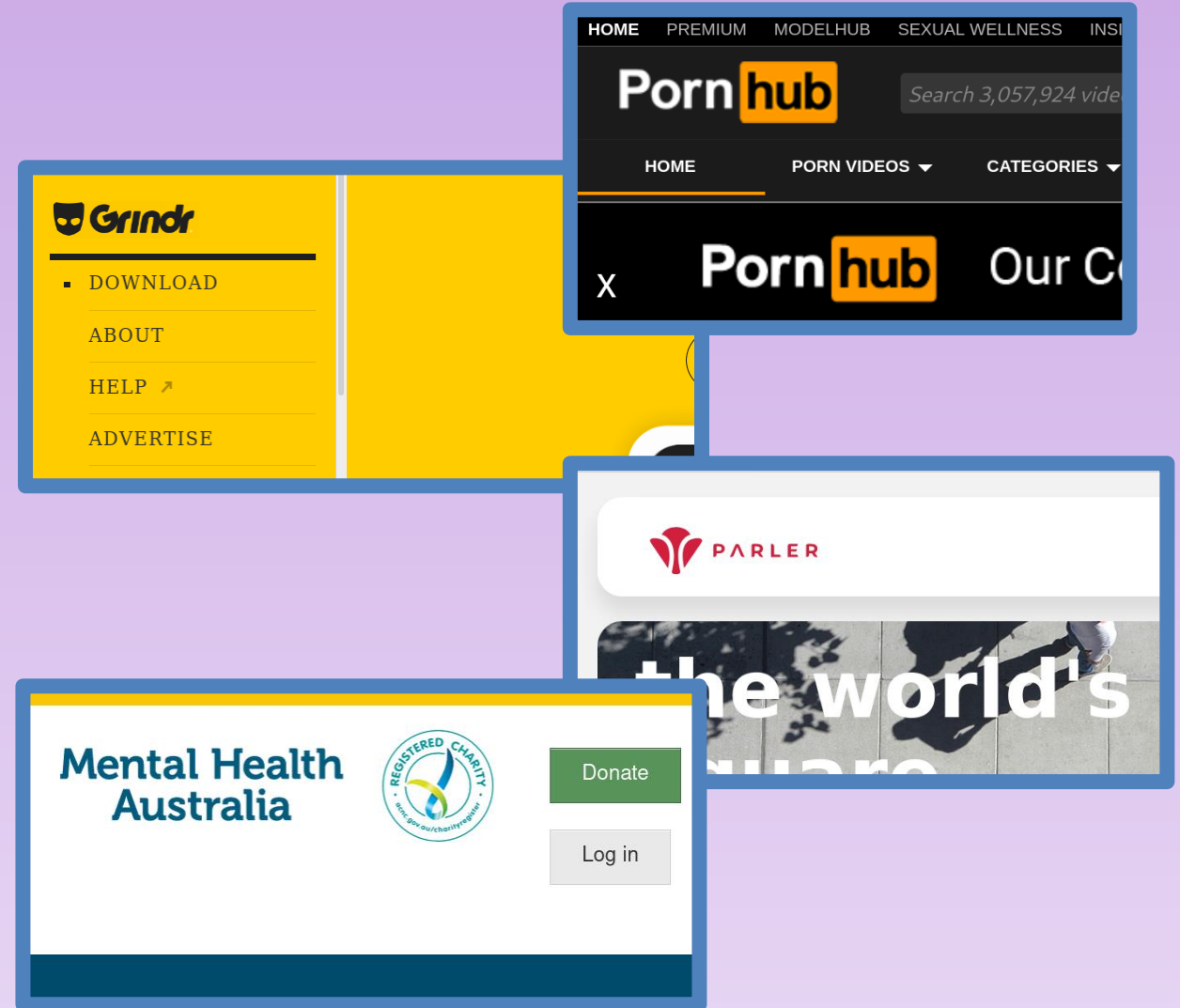
- Idea: Exploit replacement policy and LLC/directory inclusiveness to improve attacks.
- Insertion sequence ensures that the next LLC replacement candidate is in L1
- Probe only the LLC replacement candidate



# Cache Occupancy Attack

# Spy in the Sandbox

- Websites
  - Execute (sandboxed) third-party code
  - Process personal and potentially sensitive information
- Cache behaviour can identify the rendered web site (Oren et al. CCS 2015)



# Requirements for Cache Attacks

## Requirement

Execute code

Shared Cache

Arrays

Timers

DeterFox: Cao et al. (CCS 2017)

JavaScript Zero: Schwarz et al. (NDSS 2019)

# Reducing Timer Resolution








Measure this



With this

# Reducing Timer Resolution

Rank	Mark	WIND	Competitor	DOB	Nat
1	9.58	+0.9	Usain BOLT	21 AUG 1986	 JAM
2	9.69	+2.0	Tyson GAY	09 AUG 1982	 USA
2	9.69	-0.1	Yohan BLAKE	26 DEC 1989	 JAM
4	9.72	+0.2	Asafa POWELL	23 NOV 1982	 JAM
5	9.74	+0.9	Justin GATLIN	10 FEB 1982	 USA

58

+1.8

Patrick JOHNSON






26 SEP 1972



AUS

# Attack 1: Cache Occupancy

- Shusterman et al. USENIX Security 2019

Rank	Mark	Competitor	DOB	Nat
1	2:14:04	Brigid KOSGEI	20 FEB 1994	 KEN
2	2:15:25	Paula RADCLIFFE	17 DEC 1973	 GBR
3	2:17:01	Mary Jepkosgei KEITANY	18 JAN 1982	 KEN
4	2:17:08	Ruth CHEPNGETICH	08 AUG 1994	 KEN
5	2:17:16	Peres JEPCHIRCHIR	27 SEP 1993	 KEN

# Attack 1: Cache Occupancy

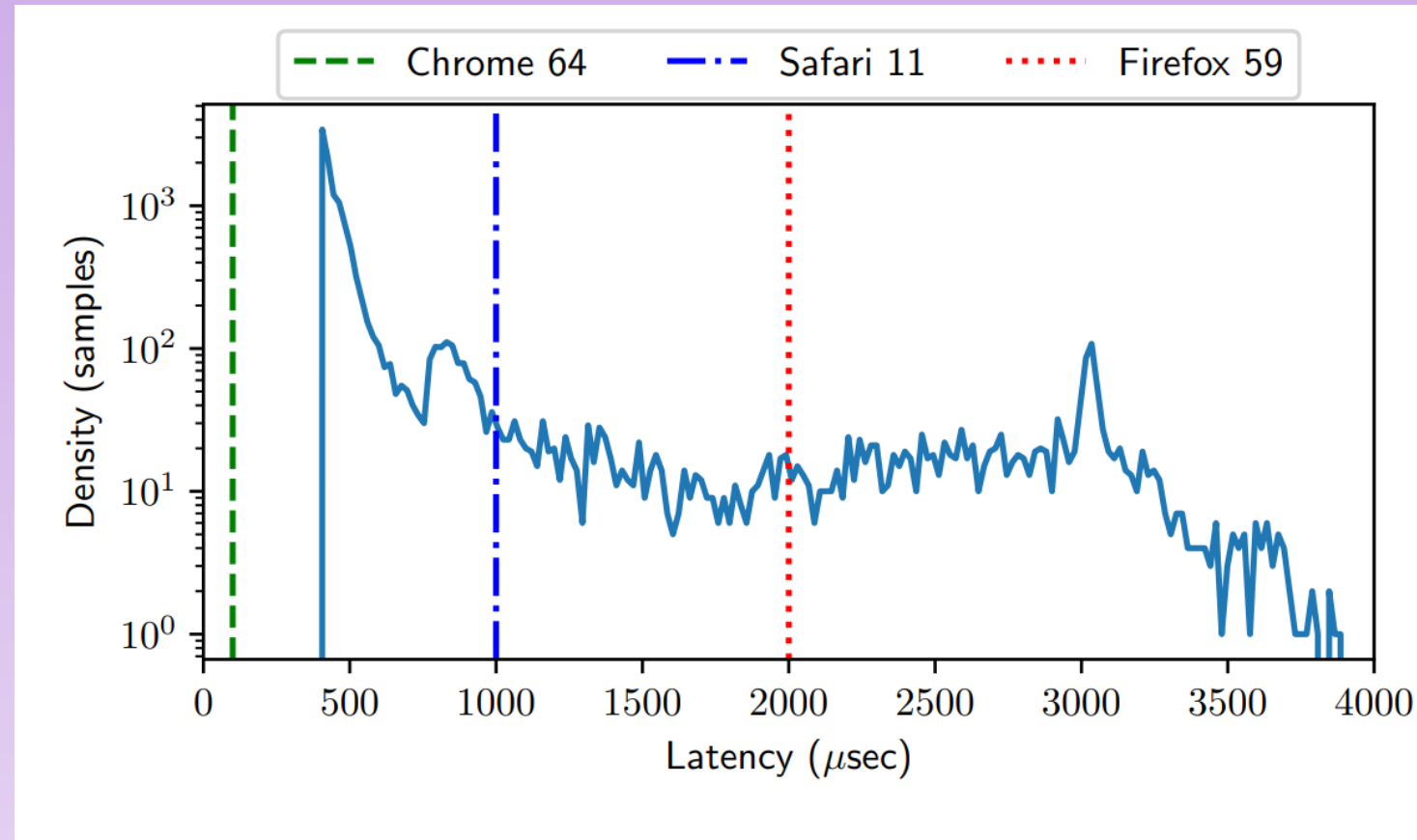
- Shusterman et al. USENIX Security 2019
- Allocate a cache-sized buffer
- Measure the time to access every cache line in the buffer
- Repeat



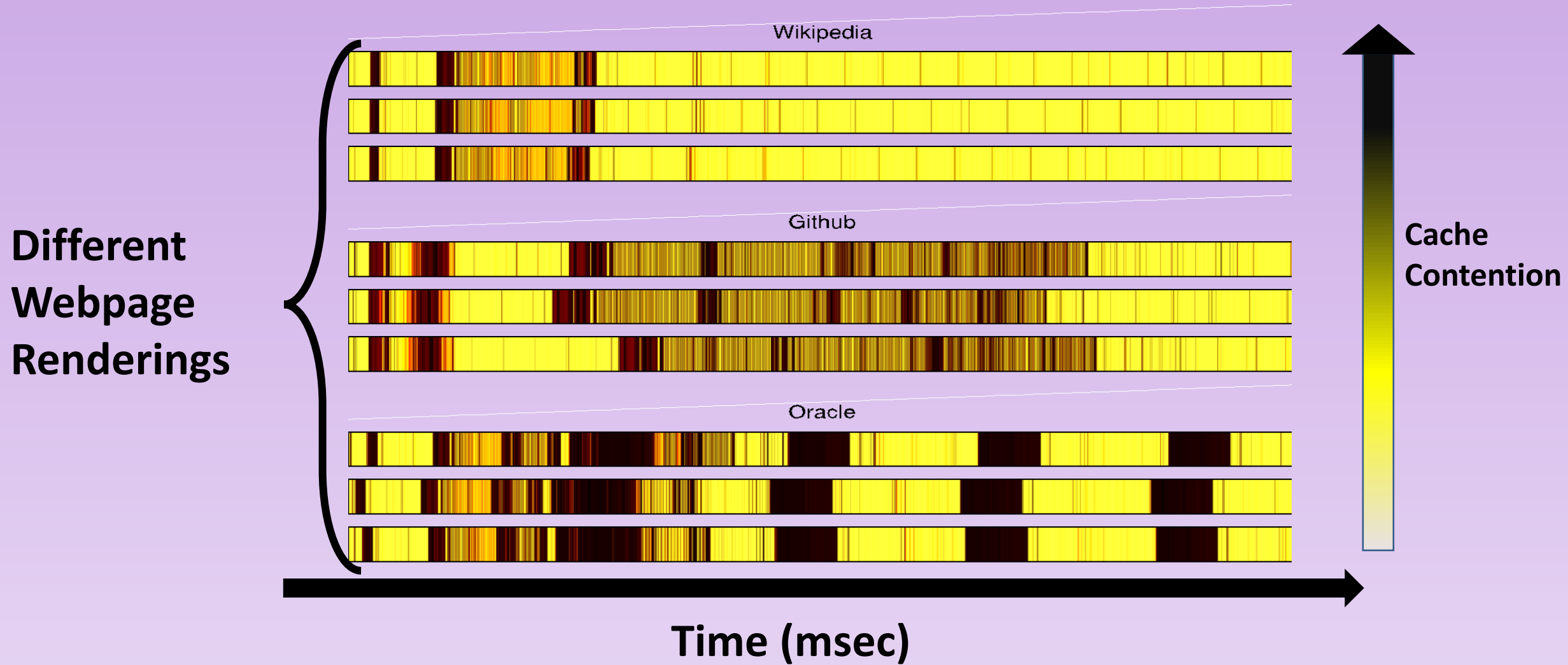
# Attack 1: Cache Occupancy

- Shusterman et al. USENIX Security 2019

- Allocate a cache-sized buffer
- Measure the time to access every cache line in the buffer
- Repeat

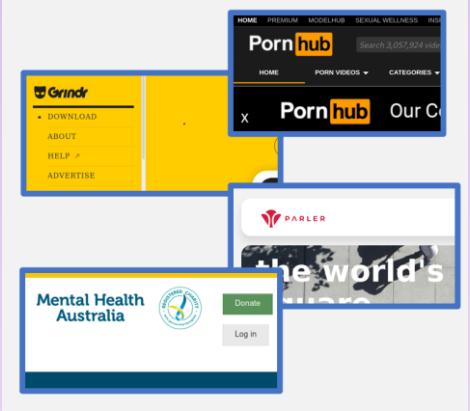
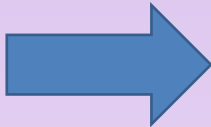
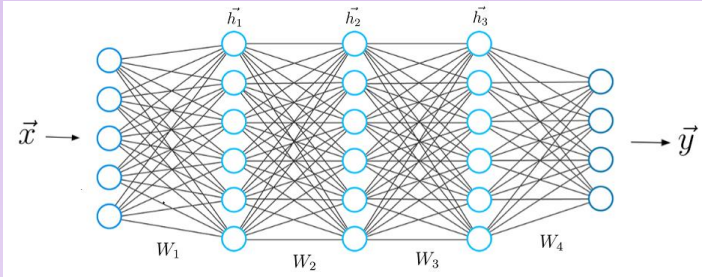
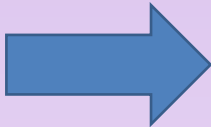
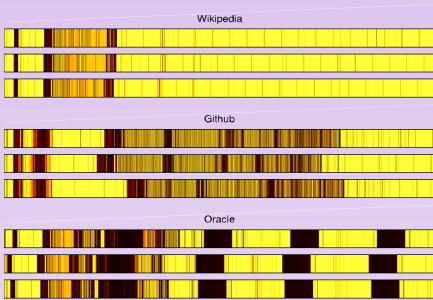


# Attack 1: Cache Occupancy



# Results

Attack Technique	Intel i5-3470	AMD Ryzen 9 3900X	Apple M1	Samsung Exynos 2100
Cache Occupancy	87.5	69.1	89.7	84.5

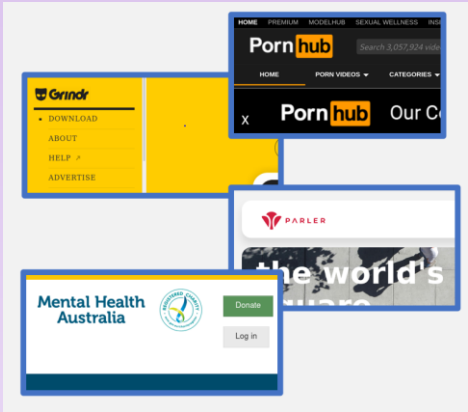
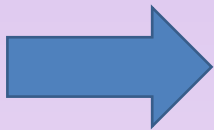
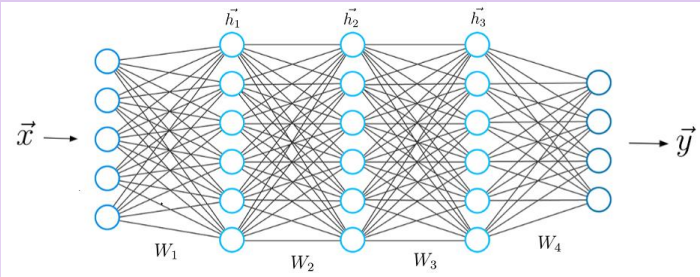
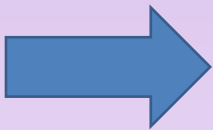
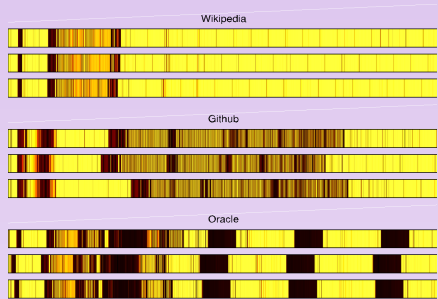


# Attack 2: Sweep Counting

- [illegible]

# Results

Attack Technique	Intel i5-3470	AMD Ryzen 9 3900X	Apple M1	Samsung Exynos 2100
Cache Occupancy	87.5	69.1	89.7	84.5
Sweep Counting	45.8	54.9	90.5	69.7



# Disabling timers

- JavaScript Zero (Schwartz et al. NDSS 2019)
  - Overwrite `performance.now()`
- DeterFox (Cao et al. CCS 2017)
  - Virtual timer
  - Deterministic timer behaviour

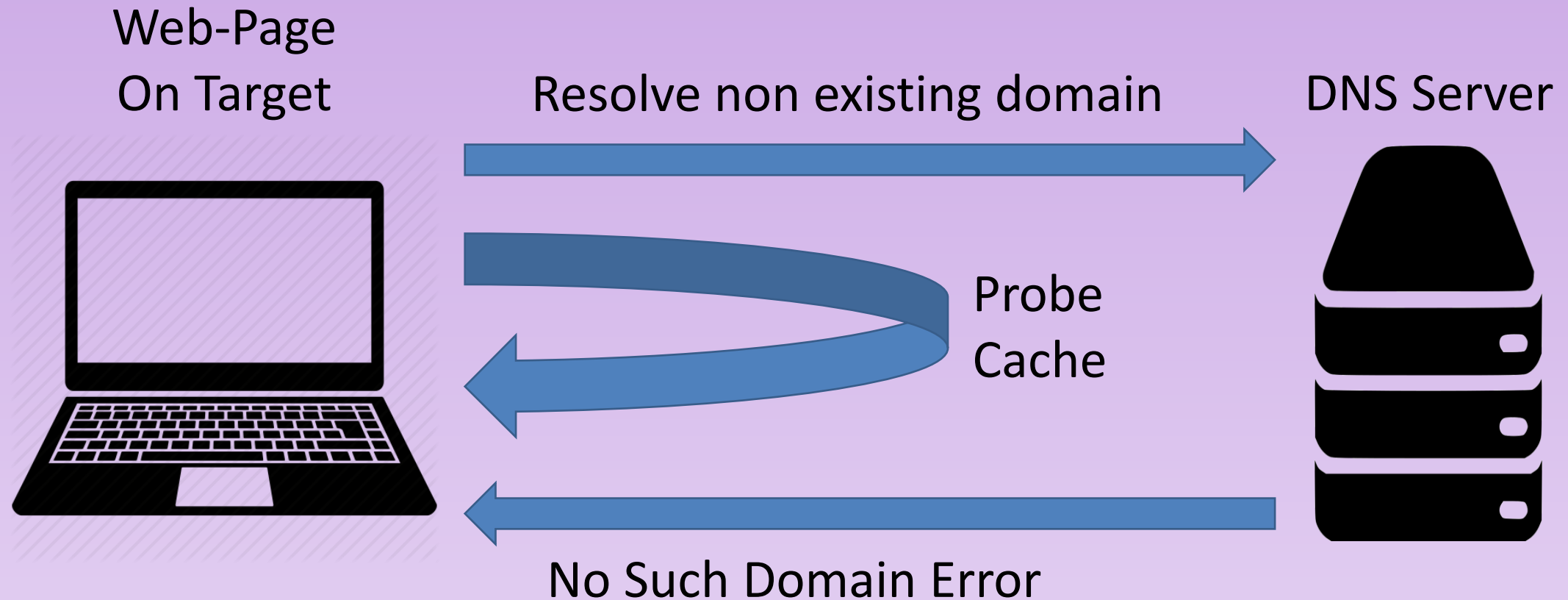


# Attack 3: DNS Racing

- Shusterman et al. USENIX Security 2021
- Use an external timer
- DNS resolution time
  - Local (Ethernet): 2ms
  - Local (Wifi): 9ms
  - Remote (cross continent): 70ms
- Low jitter – max 1ms on an intercontinental connection



# Attack 3: DNS Racing

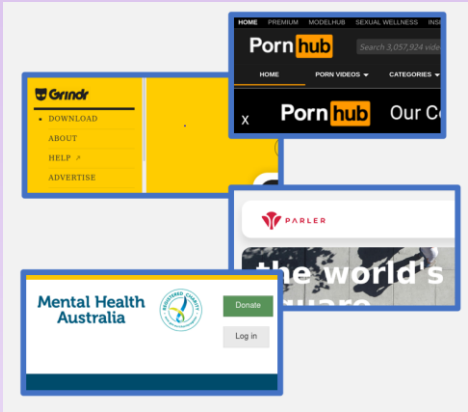
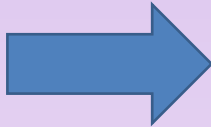
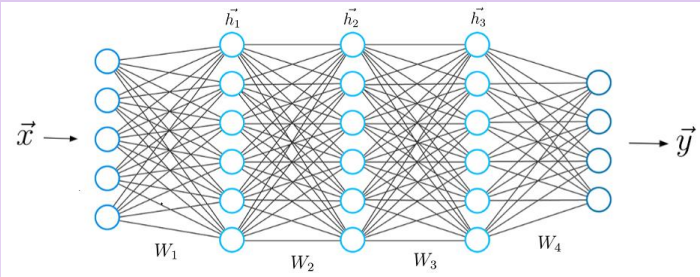
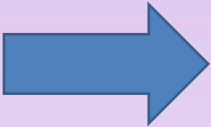
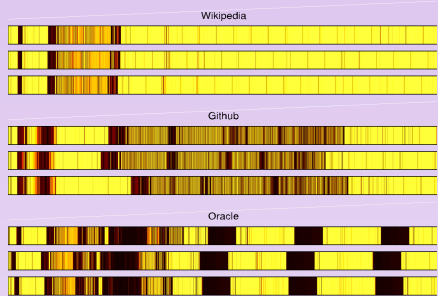


- No timers required!
- Resists jitter well enough to be used between two continents



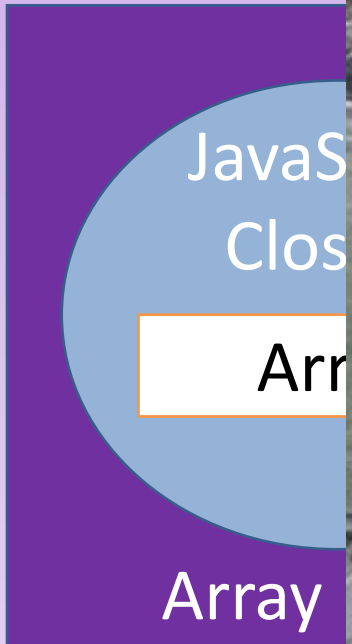
# Results

Attack Technique	Intel i5-3470	AMD Ryzen 9 3900X	Apple M1	Samsung Exynos 2100
Cache Occupancy	87.5	69.1	89.7	84.5
Sweep Counting	45.8	54.9	90.5	69.7
DNS Racing	50.8	5.4	48.2	5.8



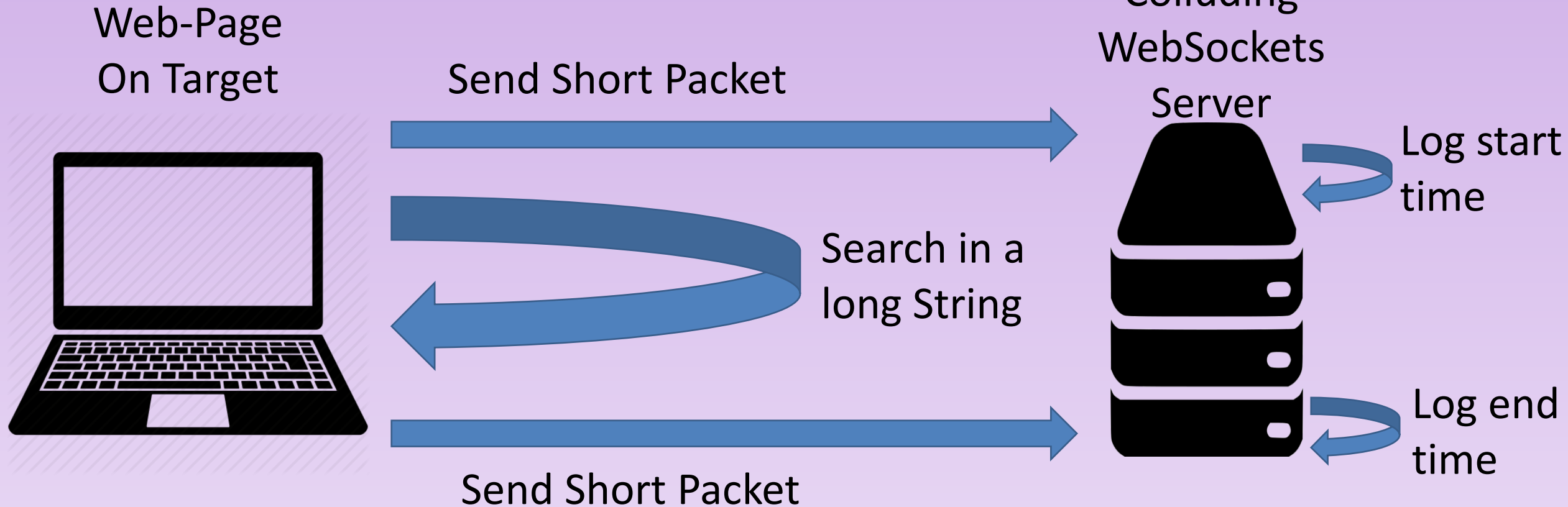
# Restricting arrays

- Schwarz et al. “JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks”, N



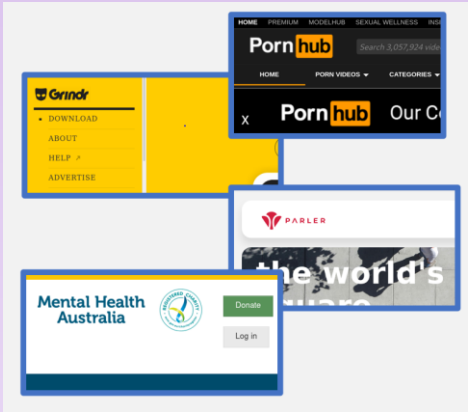
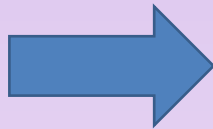
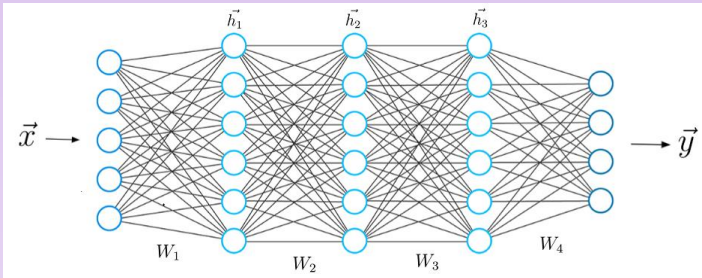
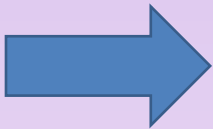
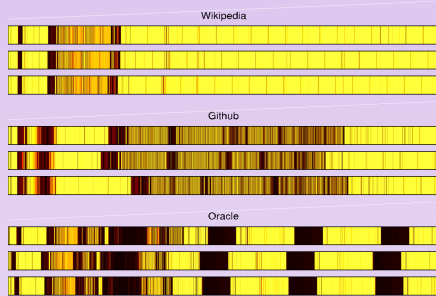
# Attack 4: String and Sock

- Shusterman et al. USENIX Security 2021
- Strings are arrays in disguise
- No timers or arrays required!



# Results

Attack Technique	Intel i5-3470	AMD Ryzen 9 3900X	Apple M1	Samsung Exynos 2100
Cache Occupancy	87.5	69.1	89.7	84.5
Sweep Counting	45.8	54.9	90.5	69.7
DNS Racing	50.8	5.4	48.2	5.8
String and Sock	72.0	53.9	90.6	60.2

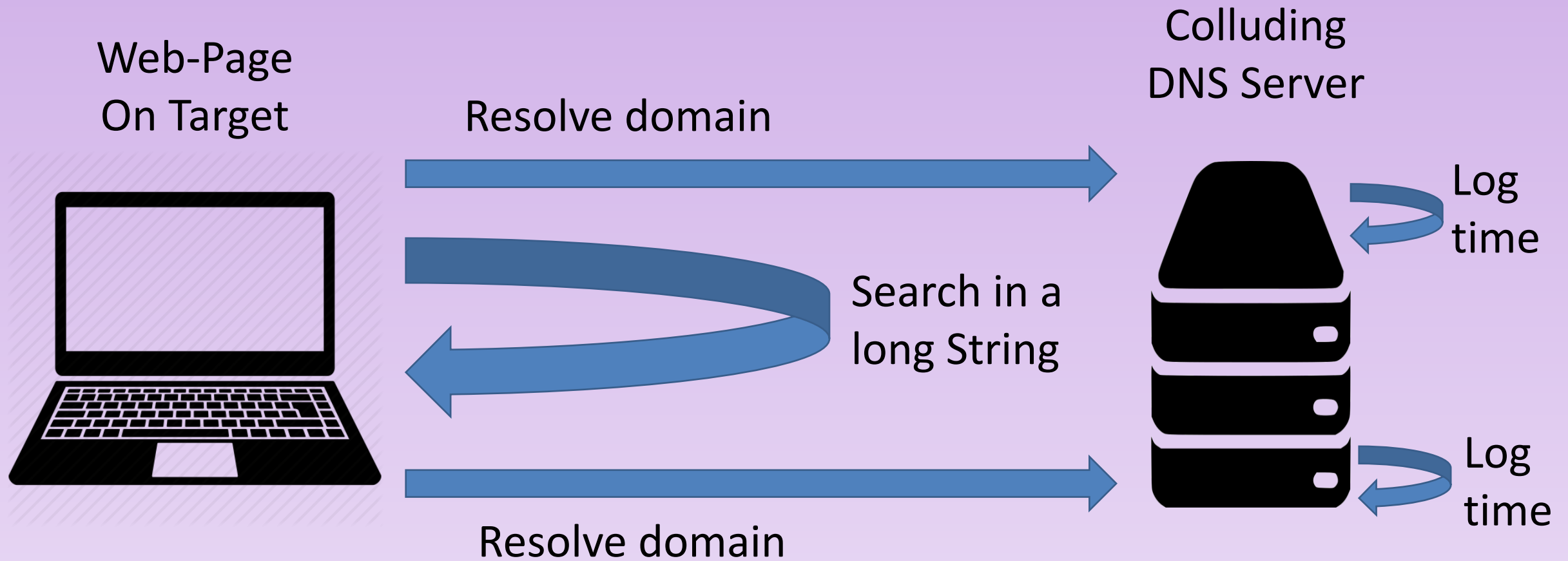


# Attack 5: CSS Prime+Probe

- Shusterman et al. USENIX Security 2021
- What if we do not have JavaScript?
- Combine DNS Racing and String and Sock:
  - CSS can search strings
  - Image downloads require DNS resolution

# Attack 5: CSS Prime+Probe

- Shusterman et al. USENIX Security 2021





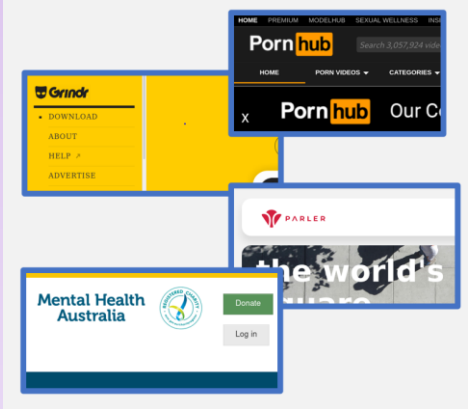
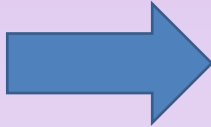
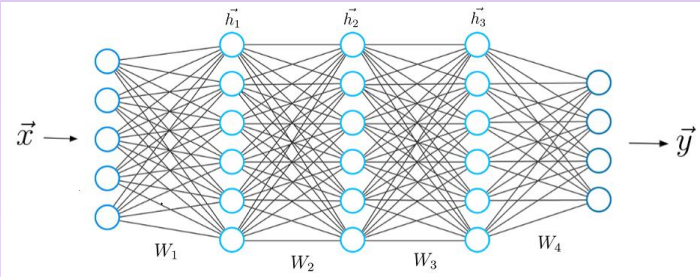
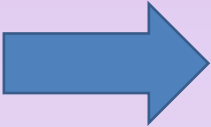
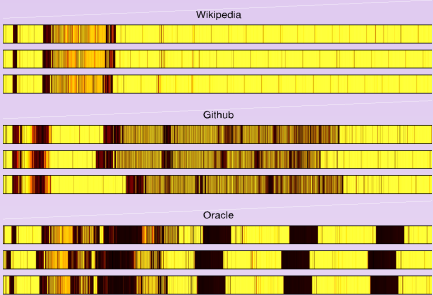
# Attack 5: CSS Prime+Probe

```
<div id="pp" class="AAA...AAA">
  <div id="s1">X</div>
  <div id="s2">X</div>
  <div id="s3">X</div>
  .
  .
  .
</div>
```

```
#pp:not([class*= 'jigbaa']) #s1 {
  background-image: url('https://knbdsd.badserver.com');
}
#pp:not([class*= 'akhevn']) #s2 {
  background-image: url('https://pjemh7.badserver.com');
}
```

# Results

Attack Technique	Intel i5-3470	AMD Ryzen 9 3900X	Apple M1	Samsung Exynos 2100
Cache Occupancy	87.5	69.1	89.7	84.5
Sweep Counting	45.8	54.9	90.5	69.7
DNS Racing	50.8	5.4	48.2	5.8
String and Sock	72.0	53.9	90.6	60.2
CSS Prime+Probe	50.1	—	15.7	—





# Summary

- Prime+Probe on the LLC
  - Allows cross-core attacks
  - Finding eviction sets is complex
- Prime+Abort and Prime+Scope improve temporal resolution
- Cache Occupancy attacks
  - Low temporal and spatial resolution
  - Extremely resilience against countermeasures
- Next lecture: Flush based attacks
  - Reading: Evtvushkin et al. “Jump overASLR: Attacking branch predictors to bypass ASLR”, MICRO 2016