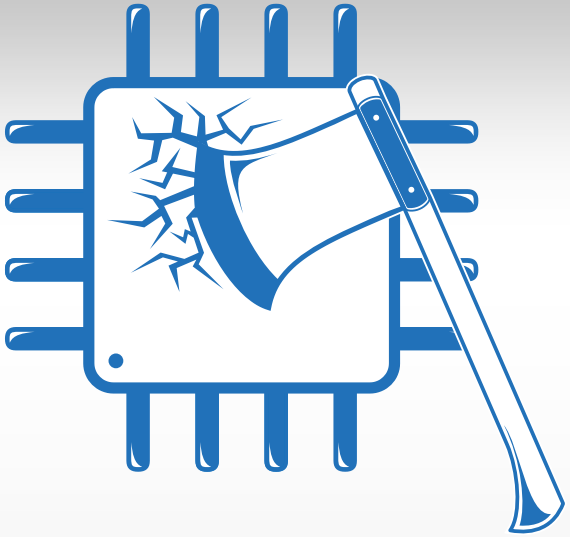
CacheOut and SGAxe: How SGX Fails in Practice



Stephan van Schaik, Marina Minkin, Andrew Kwong
Daniel Genkin and Yuval Yarom

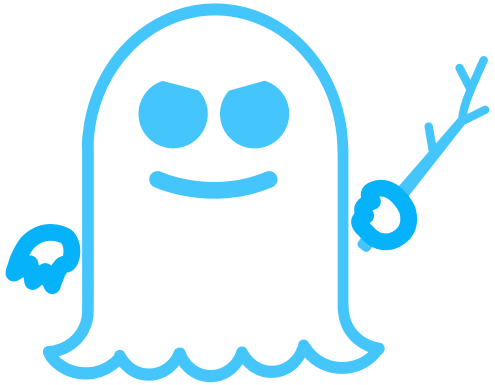
CSE COMPUTER SCIENCE AND ENGINEERING UNIVERSITY OF MICHIGAN

DATA 61

THE UNIVERSITY of ADELAIDE

Recap (2018)

MELTDOWN
CVE-2017-5754

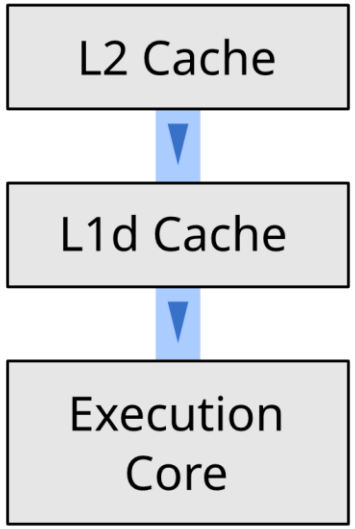


SPECTRE
CVE-2017-5715
CVE-2017-5753

FORESHADOW
CVE-2018-3615
CVE-2018-3620
CVE-2018-3646

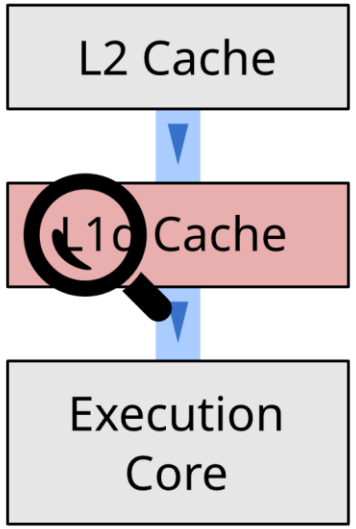
Disclosed in 2018: speculative- and transient-execution attacks

Recap (2018) - Meltdown & Foreshadow



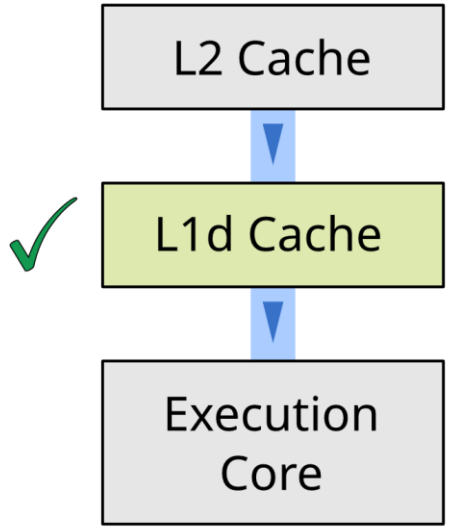
Basic cache hierarchy

Recap (2018) - Meltdown & Foreshadow



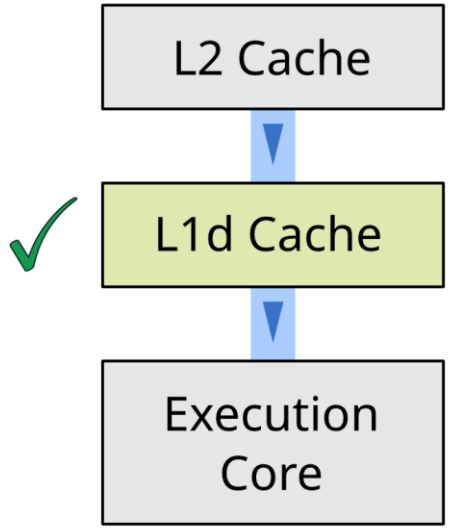
Meltdown and Foreshadow leak data from caches

Recap (2018) - Meltdown & Foreshadow



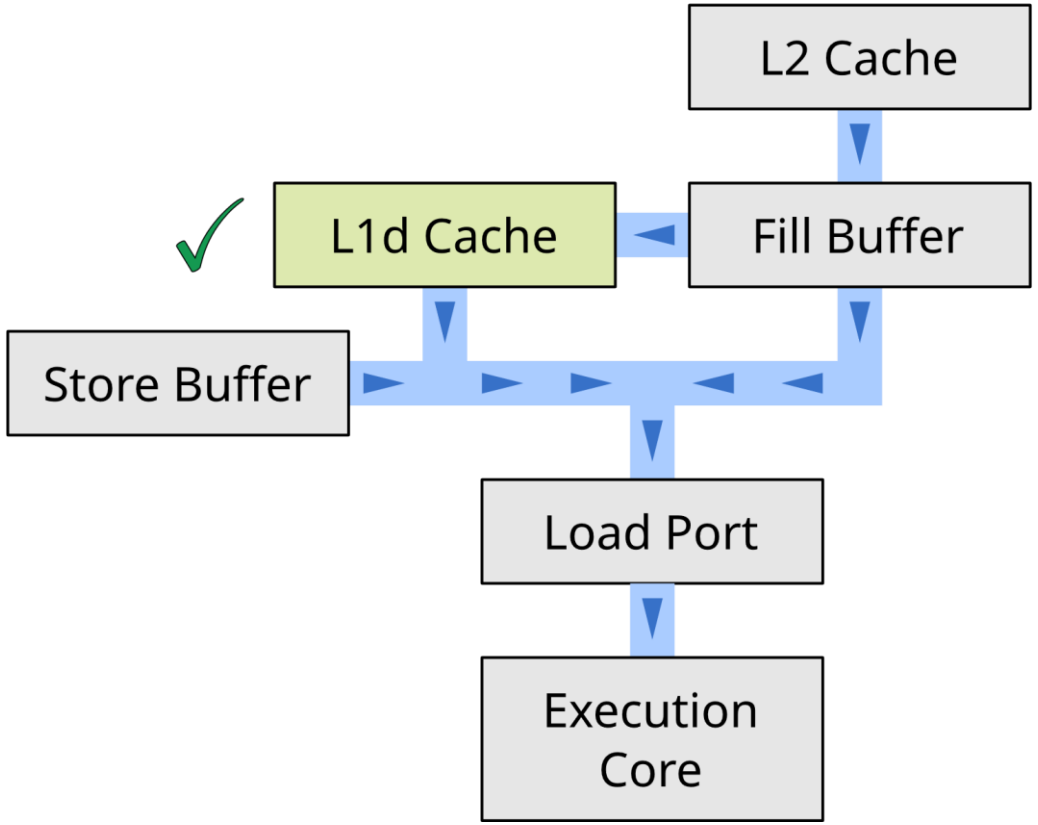
Prompted wide mitigation effort

Recap (2018) - Meltdown & Foreshadow



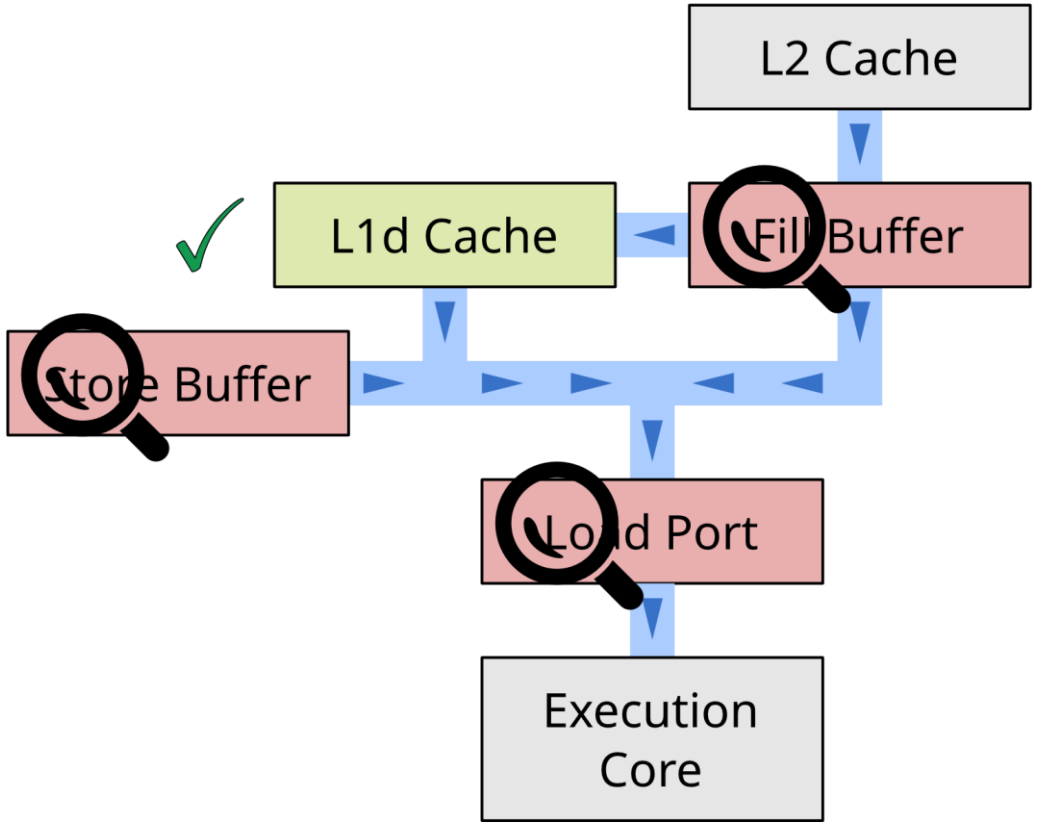
No longer possible to leak from caches

Recap (2019) - MDS Attacks



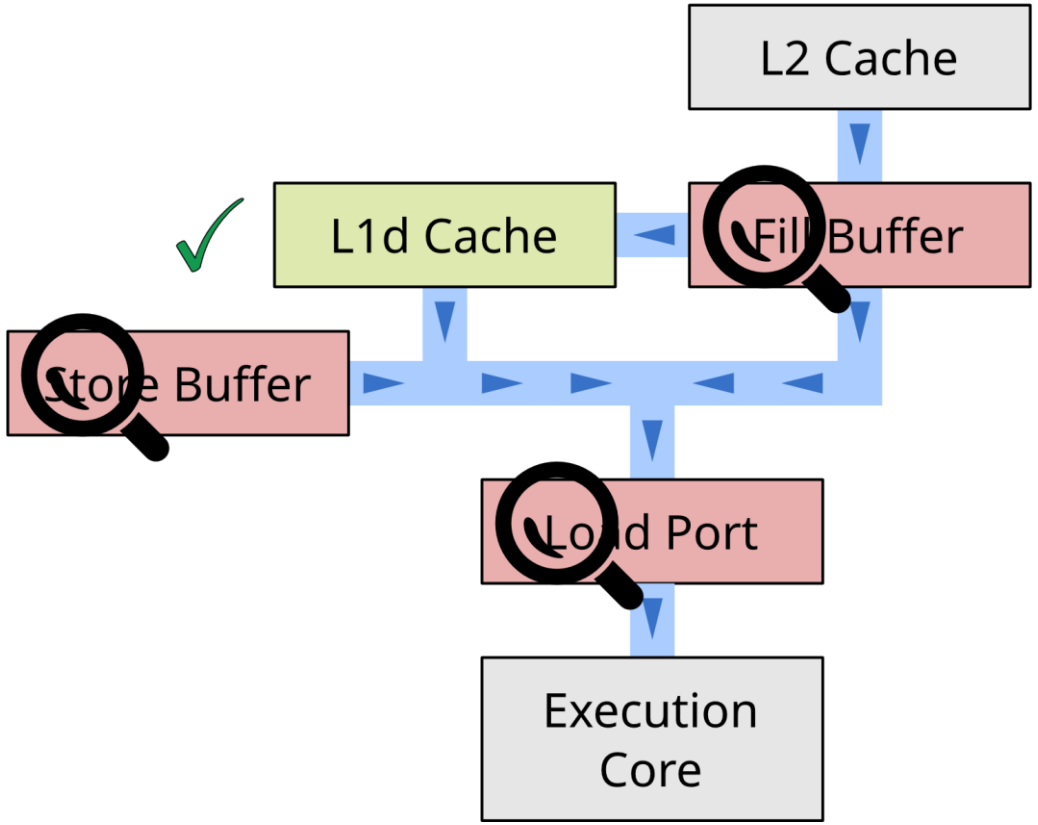
Turns out there are more buffers

Recap (2019) - MDS Attacks



Disclosed in 2019: **M**icro-architectural **D**ata **S**ampling

Recap (2019) - MDS Attacks

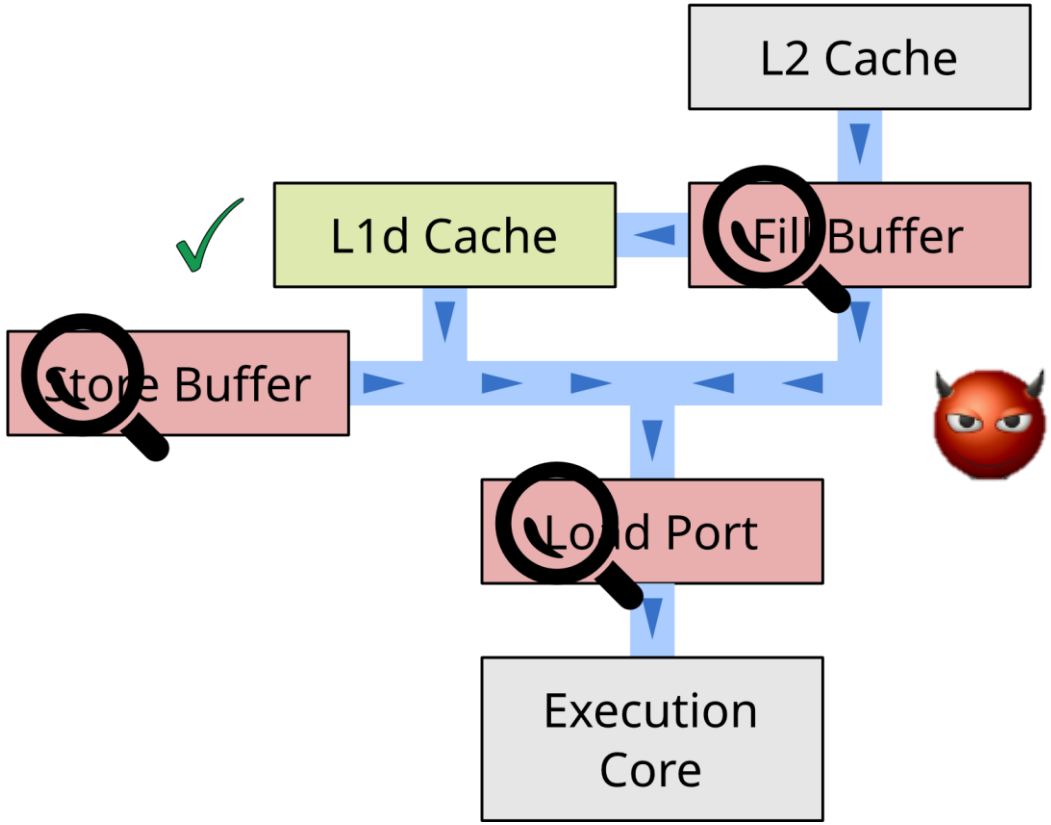


MDS targets internal CPU buffers

Micro-architectural Data Sampling

- Access other people's data through faulting or assisting loads
- Across processes, VMs and Intel SGX

Micro-architectural Data Sampling



MDS samples data passing through these buffers

Micro-architectural Data Sampling

MDS attacks are like drinking from a fire hose

Micro-architectural Data Sampling

You just get whatever data is in flight!

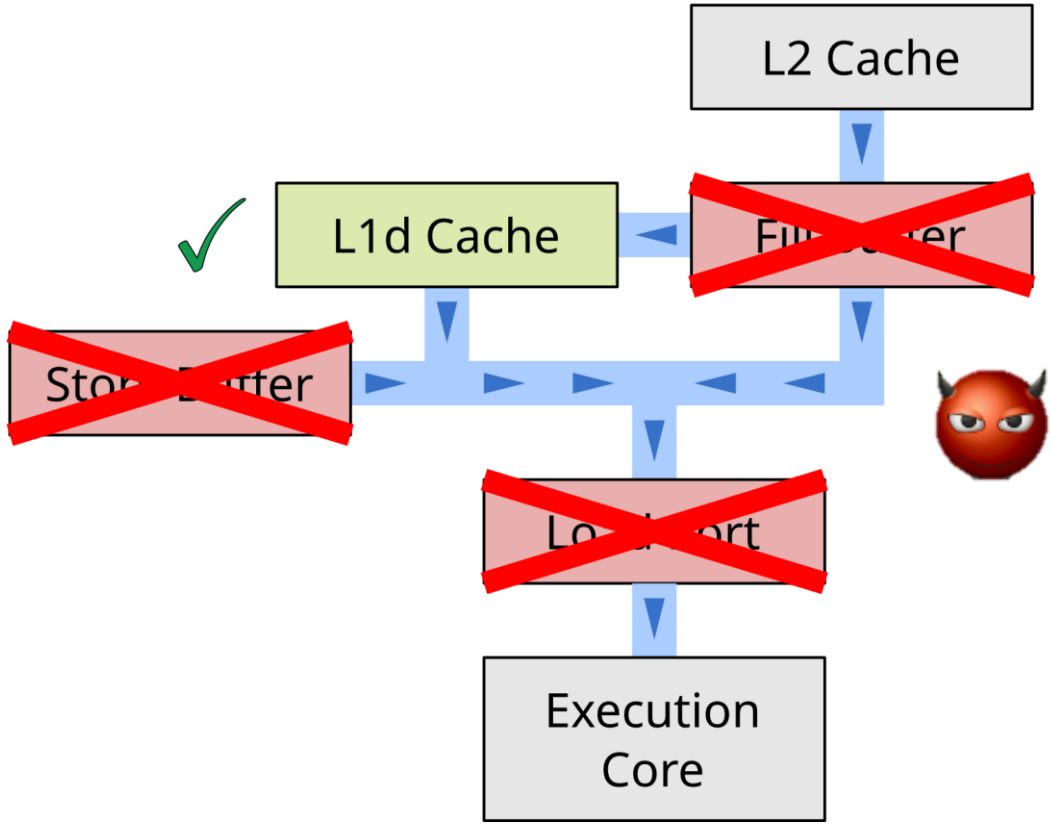
Micro-architectural Data Sampling

Hence the name: "**M**icro-architectural **D**ata **S**ampling"

Micro-architectural Data Sampling

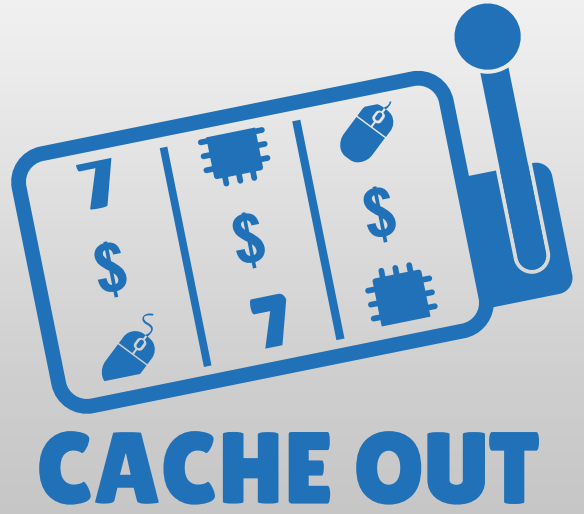
No control over what you are getting

Mitigating MDS



Mitigation: `verw` to flush the internal CPU buffers

Part 2: CacheOut



Mitigating MDS

- Flush internal CPU buffers
- Does not fix the **root cause** behind MDS!
- You can still leak data from those buffers
- Is flushing buffers sufficient as a mitigation?
- Bonus: can we get some more control?

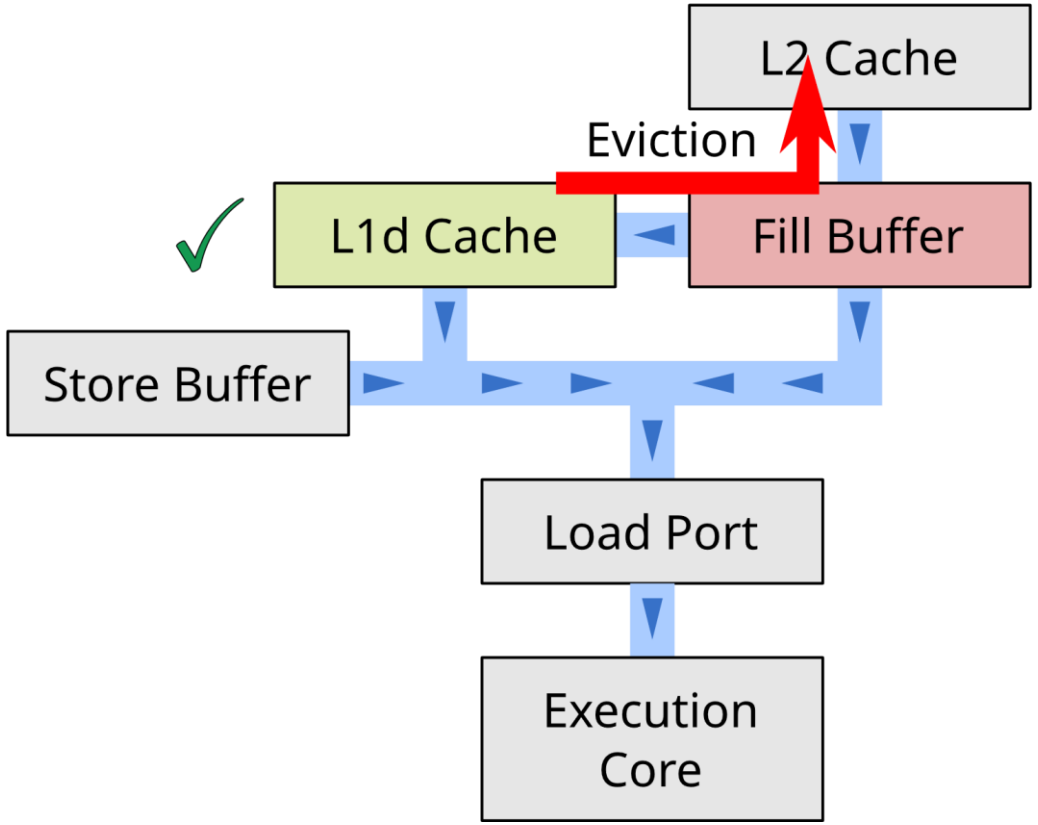
Observations

"… evicting the previous caches line from the L1d cache … the processor has to write them back through the memory hierarchy and will do this through the **LFB** ..."

The RIDL paper (https://mdsattacks.com/files/ridl.pdf)

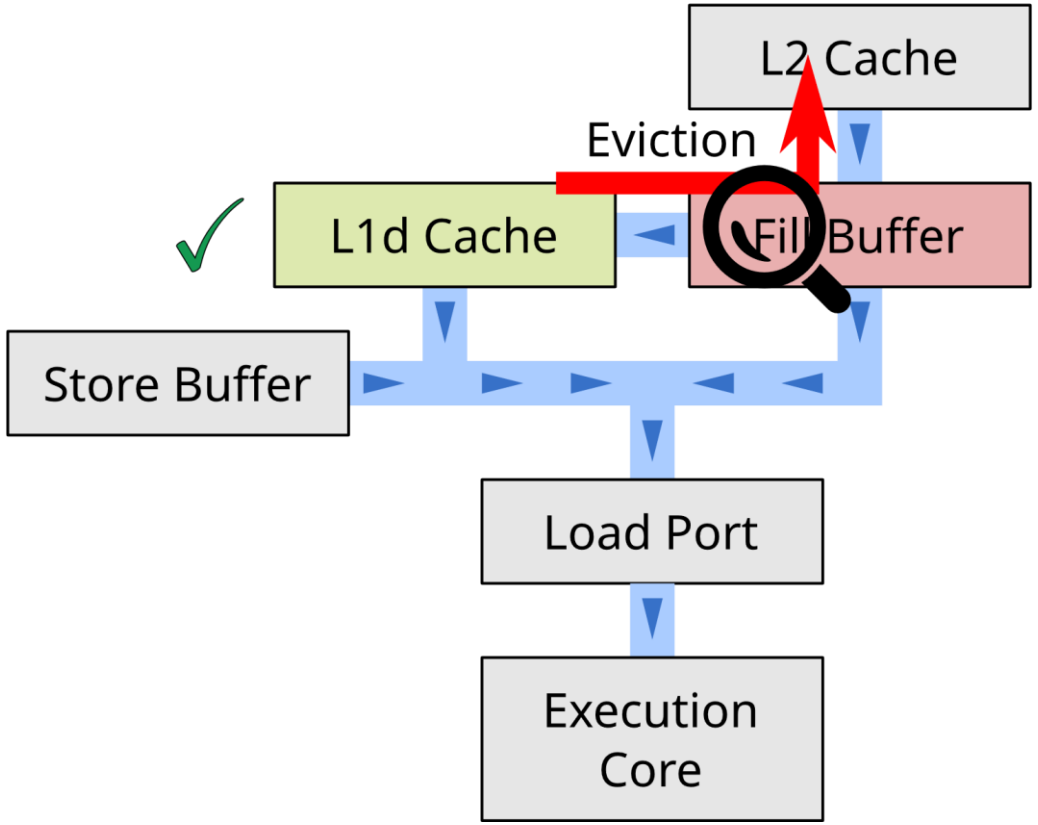
- Evicting the cache forces data into the fill buffer
- ZombieLoad reports 0.1 B/s leakage despite mitigations
- This **residual** leakage is worrisome

New Data Path



Data path for eviction through the LFB

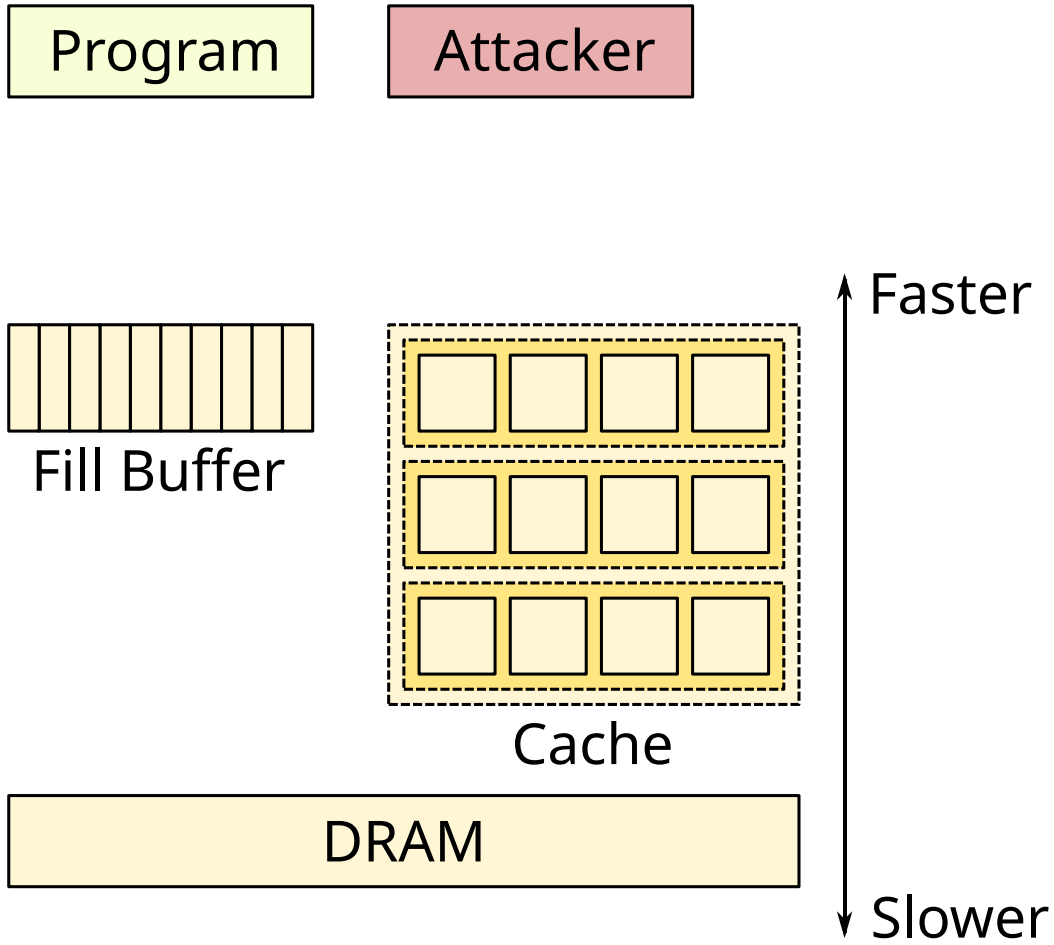
New Data Path



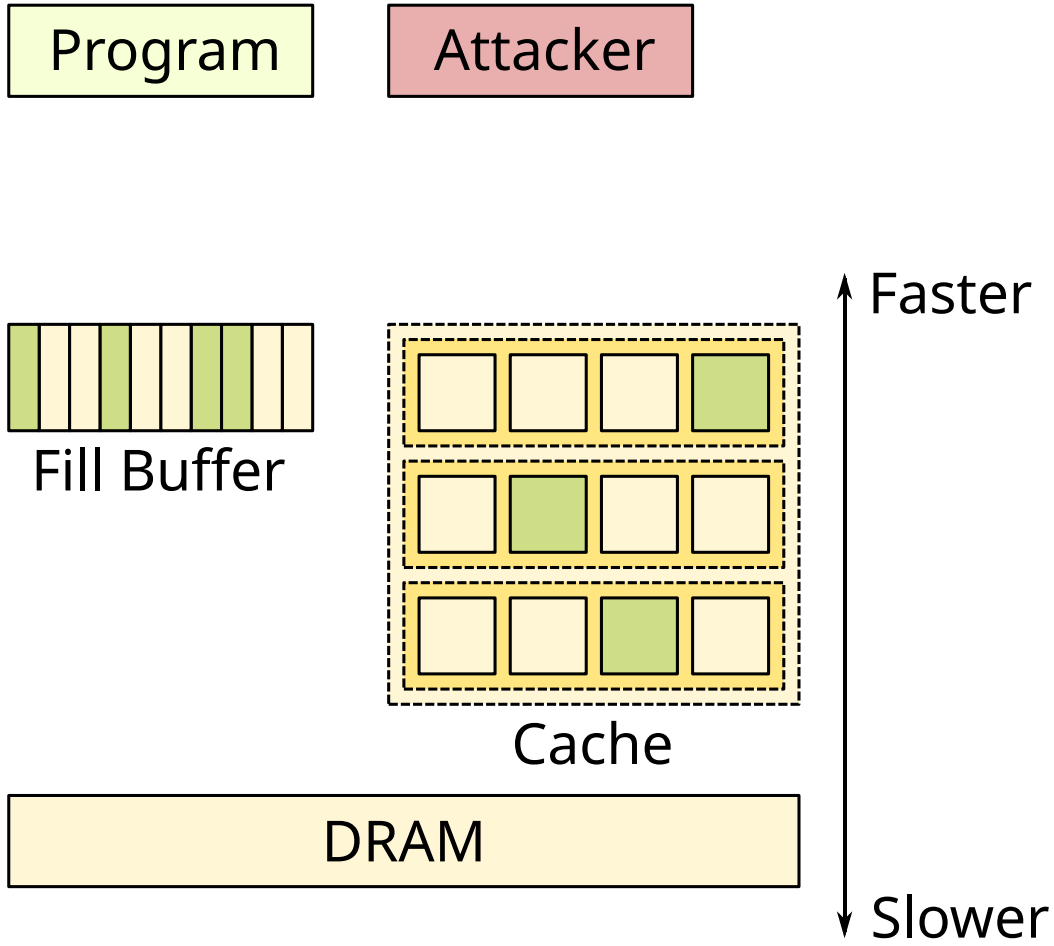
But the LFB is still leaky!

So how do we exploit evictions?

Exploiting Evictions

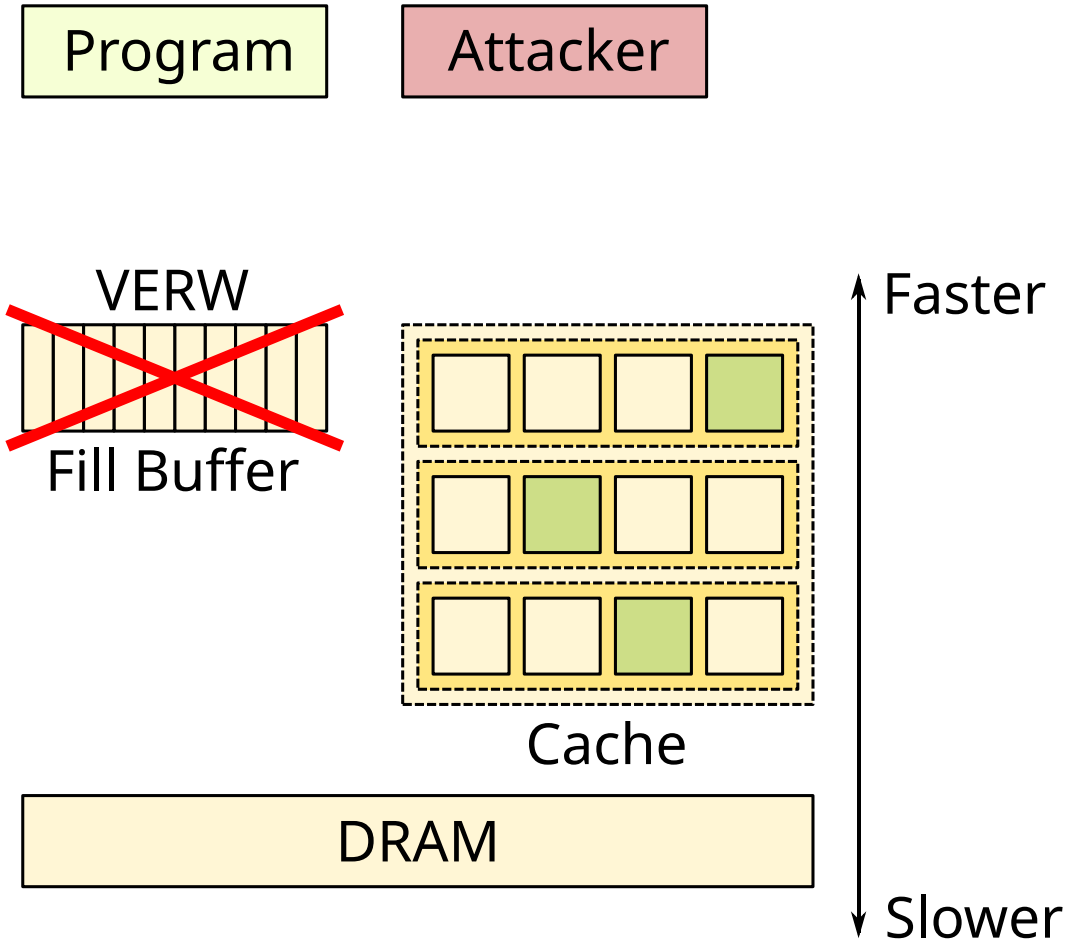


Exploiting Evictions



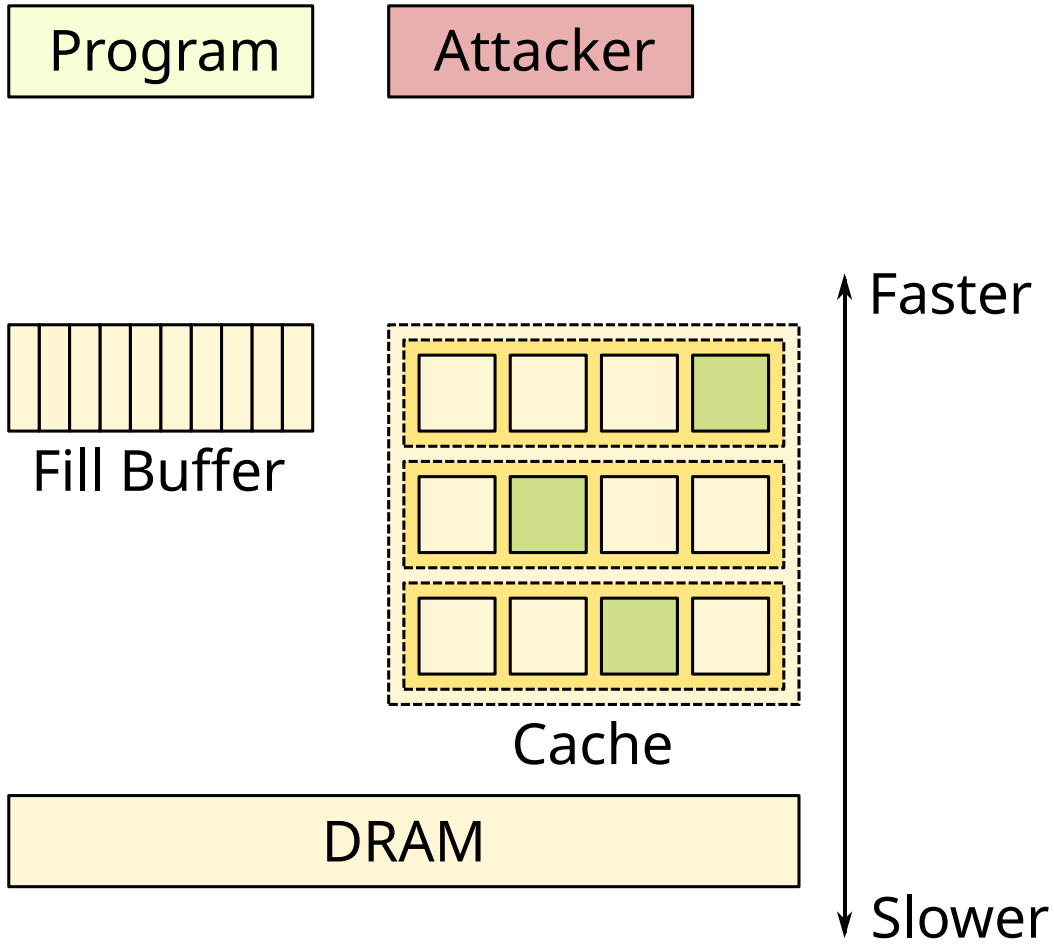
The victim's data is in the fill buffer and cache

Exploiting Evictions



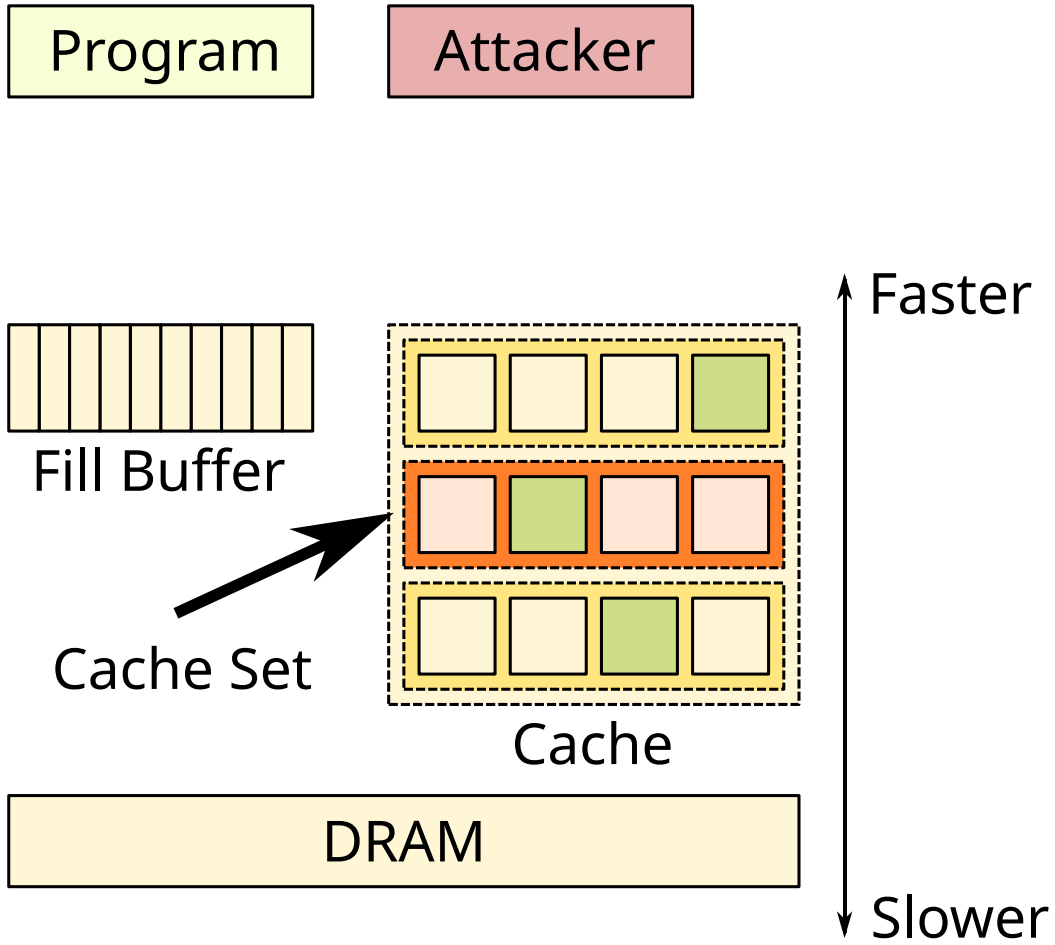
However, `verw` flushes that data from the fill buffer

Exploiting Evictions



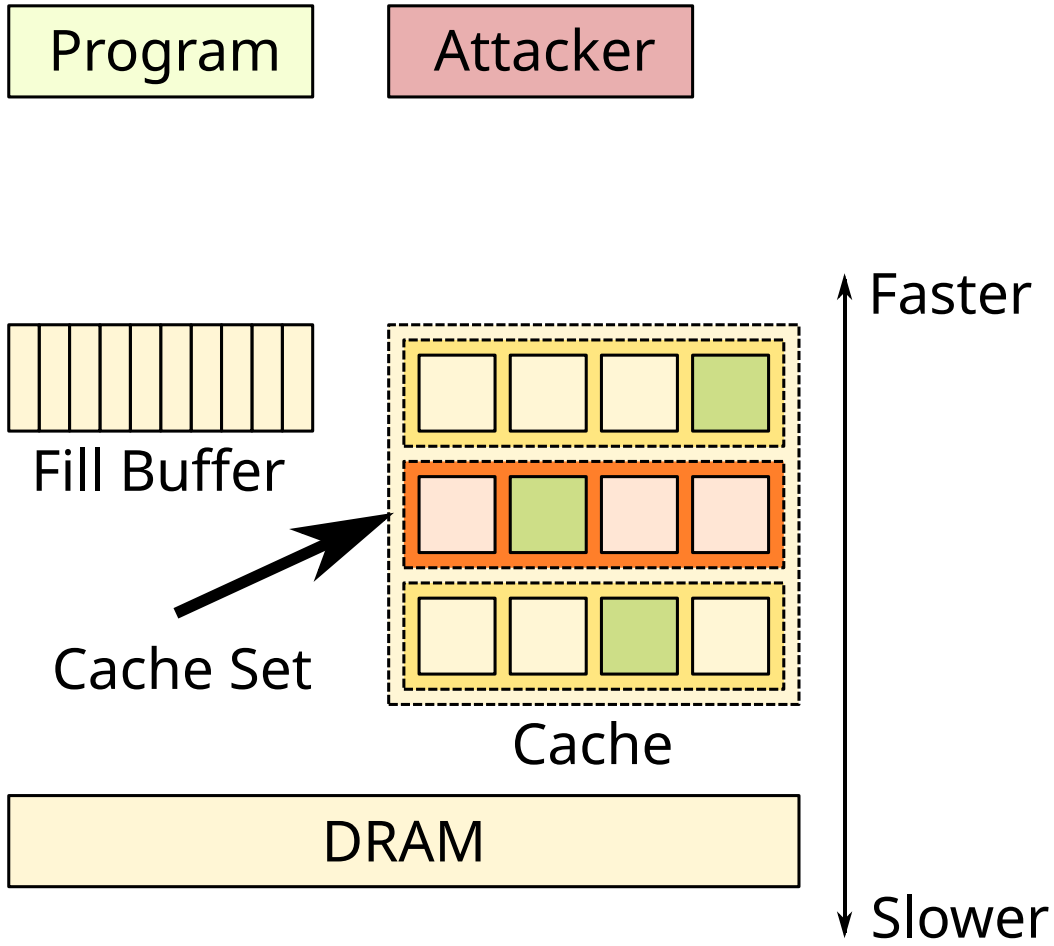
How do we leak the data from the cache?

Exploiting Evictions



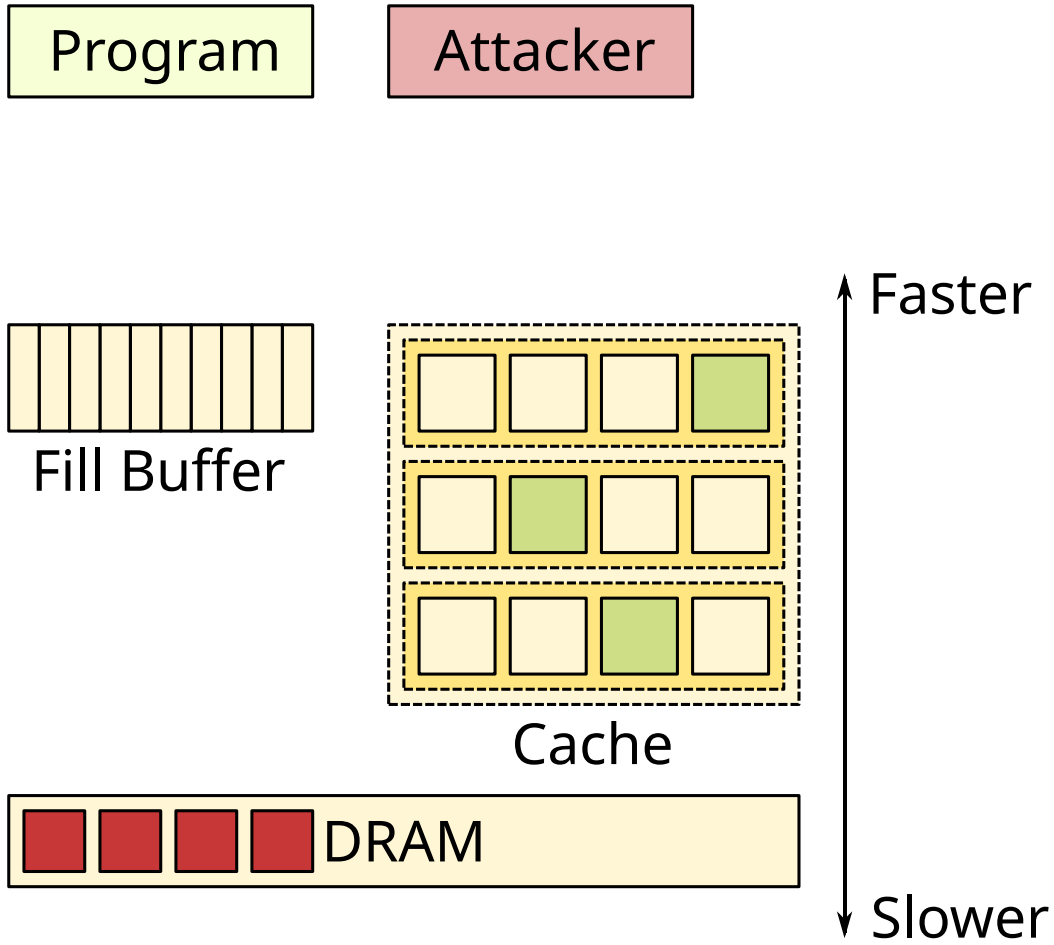
The cache is divided into sets

Exploiting Evictions



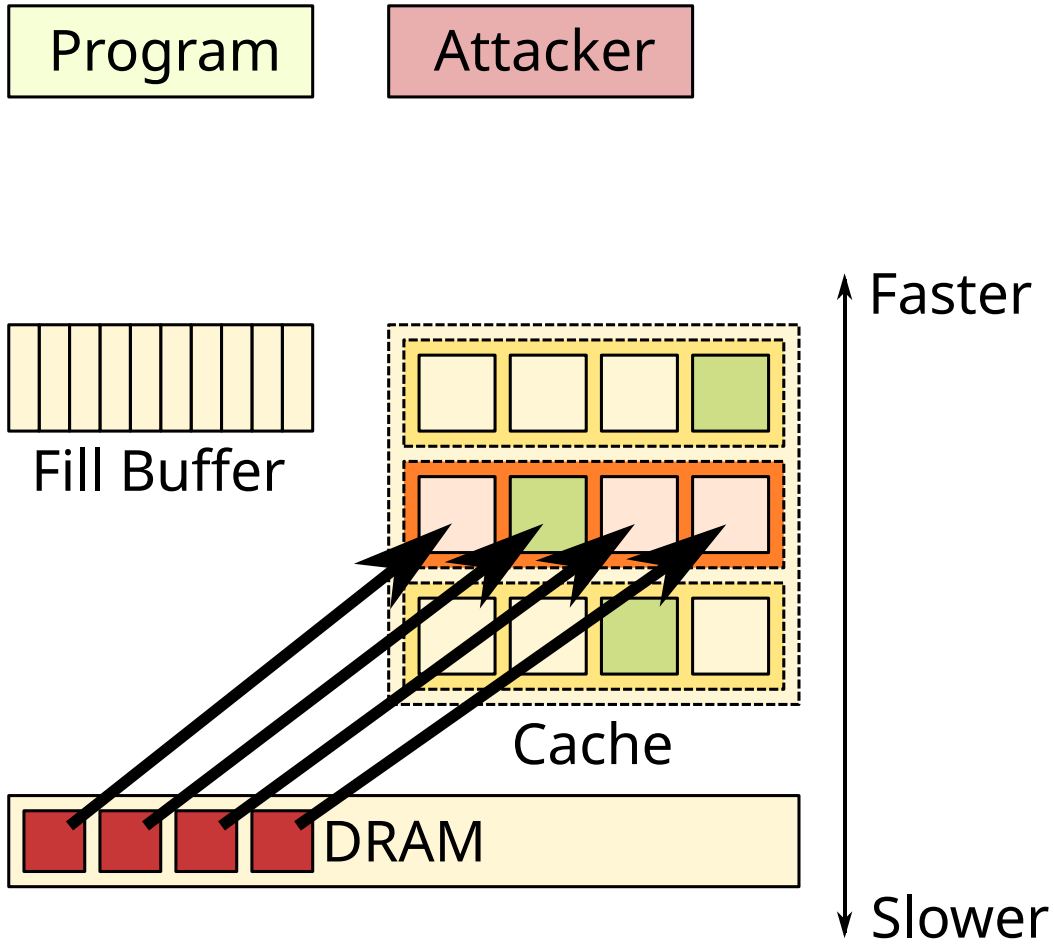
The address determines in which set the data is

Exploiting Evictions



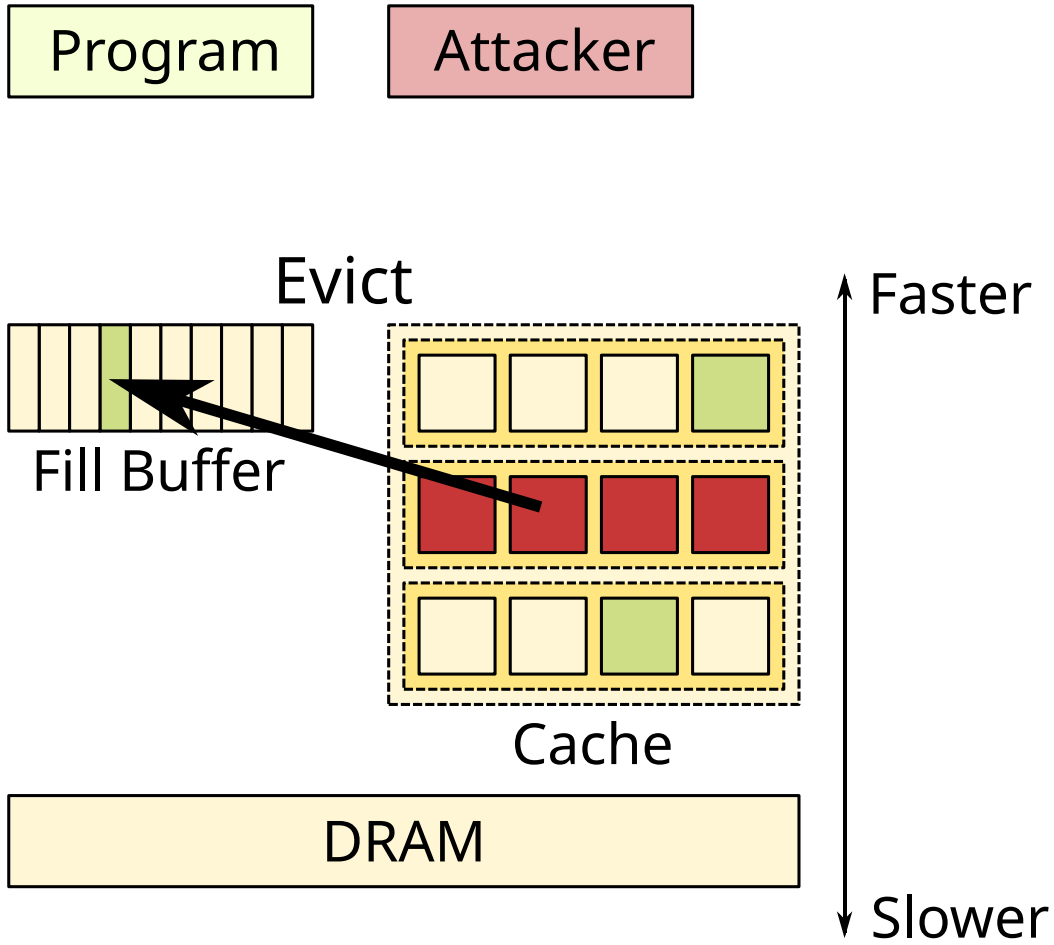
The attacker reads addresses mapping to the same set

Exploiting Evictions



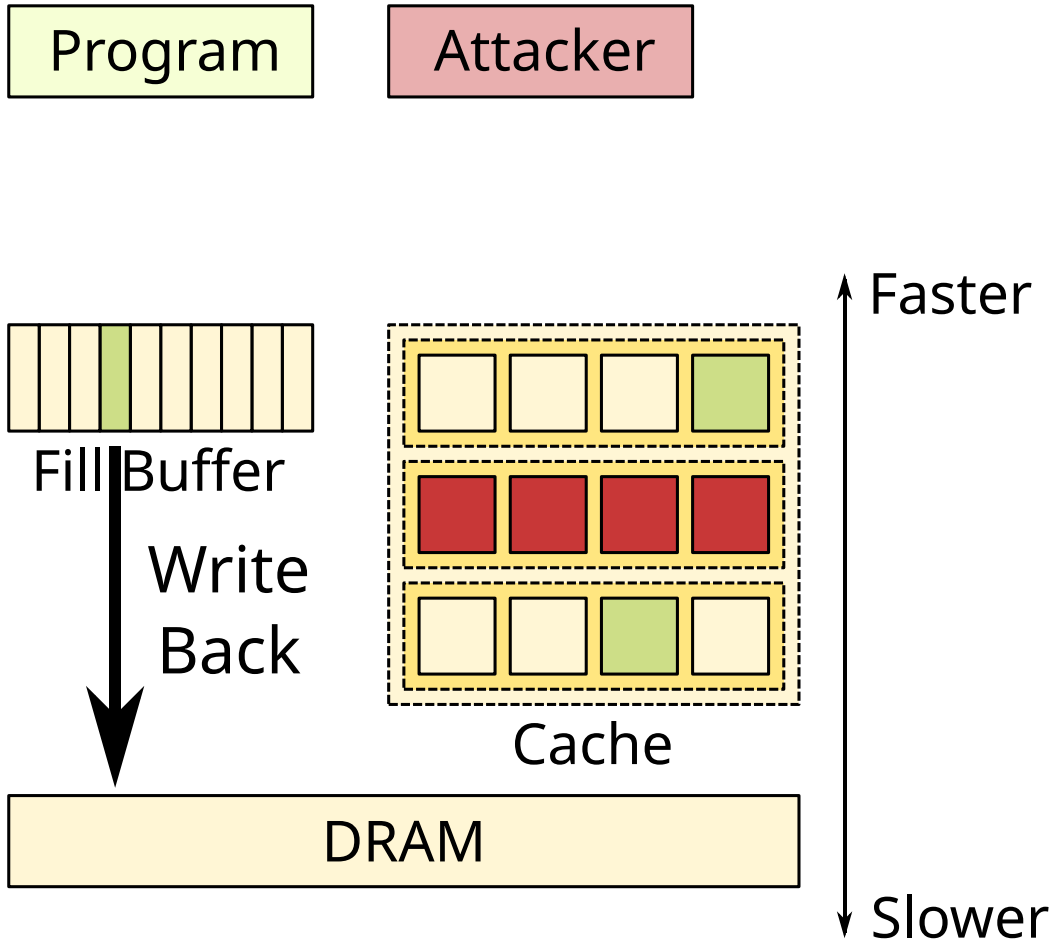
and fills the cache set with its own data

Exploiting Evictions



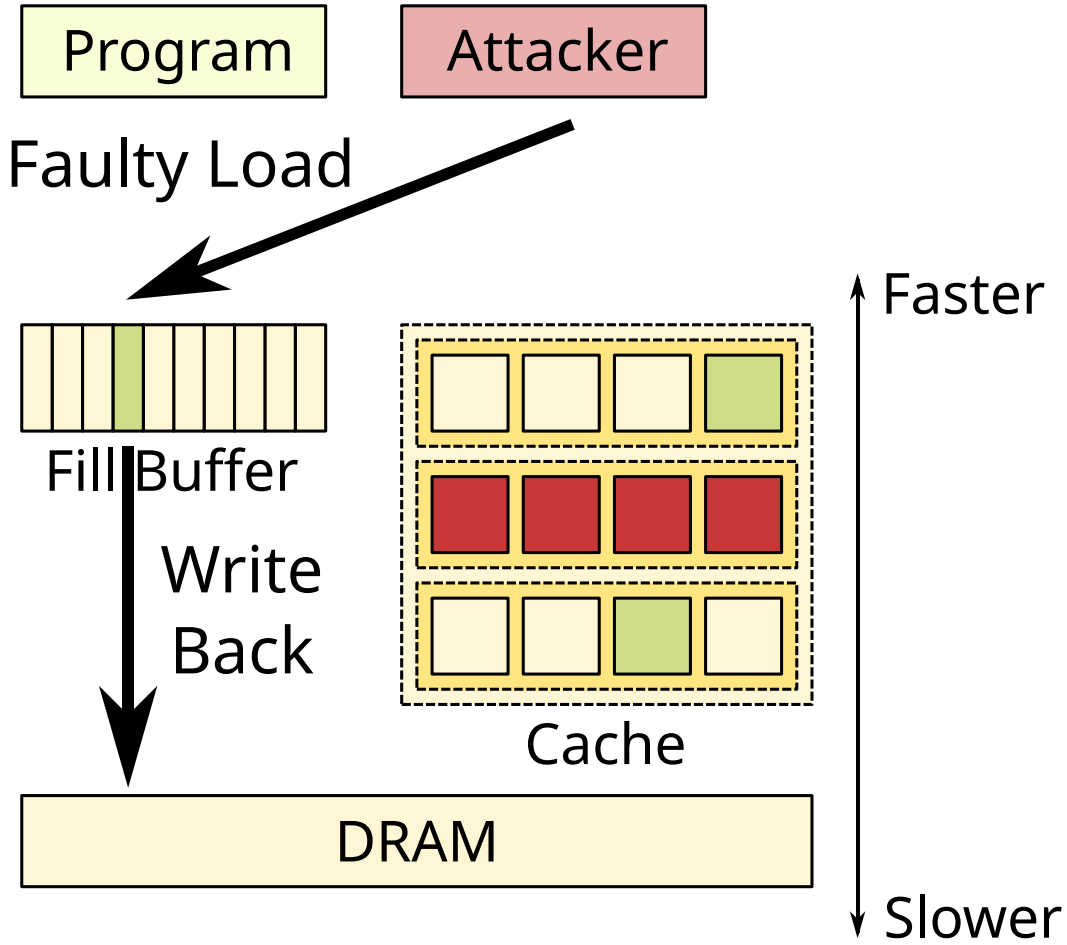
evicting the victim's data into the fill buffer

Exploiting Evictions



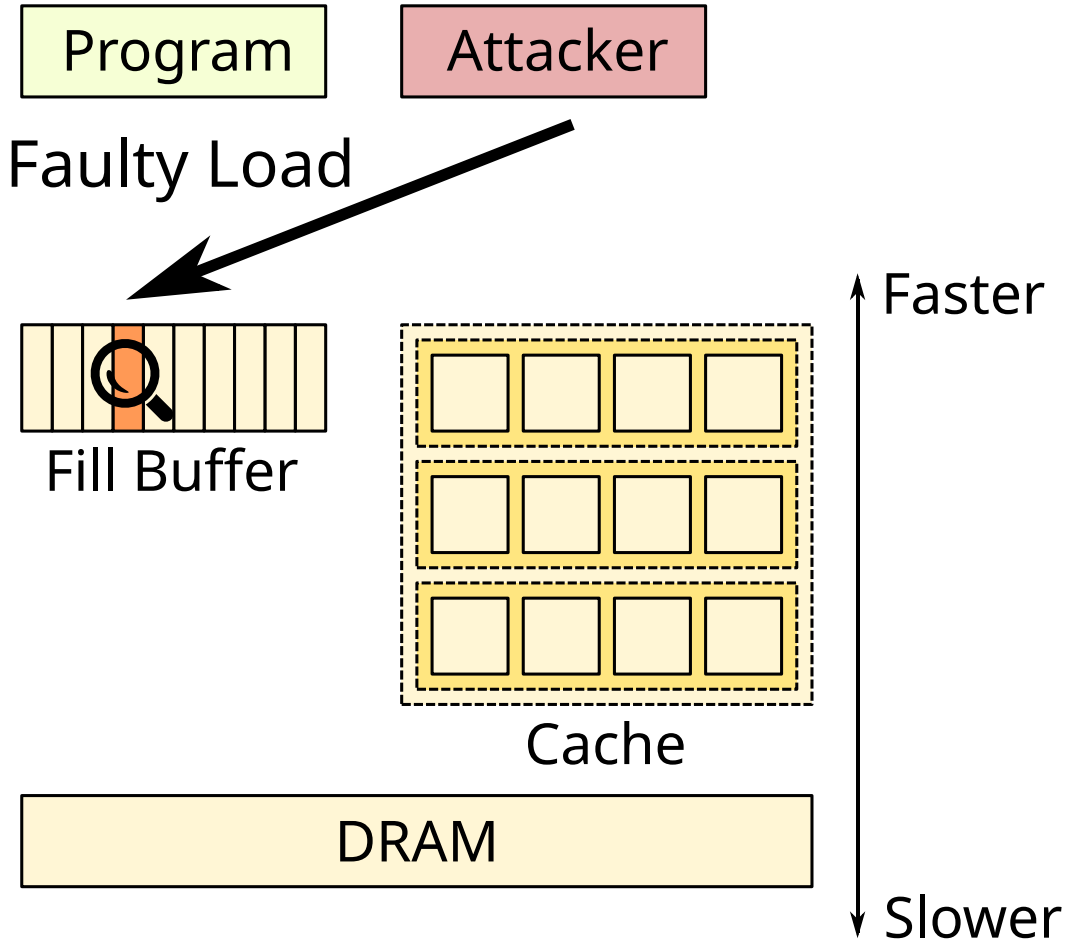
While the data gets written back to DRAM

Exploiting Evictions



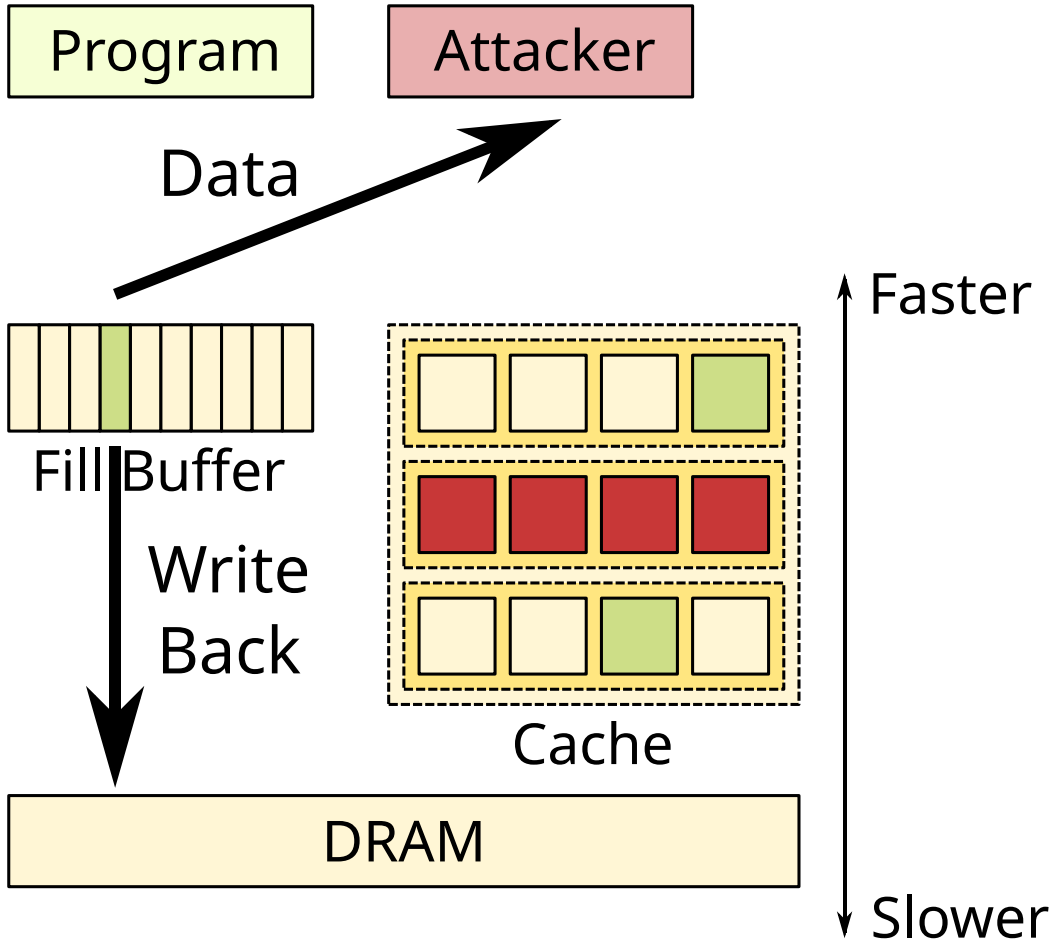
The attacker uses a faulty load to leak the data

Exploiting Evictions



The attacker uses a faulty load to leak the data

Exploiting Evictions

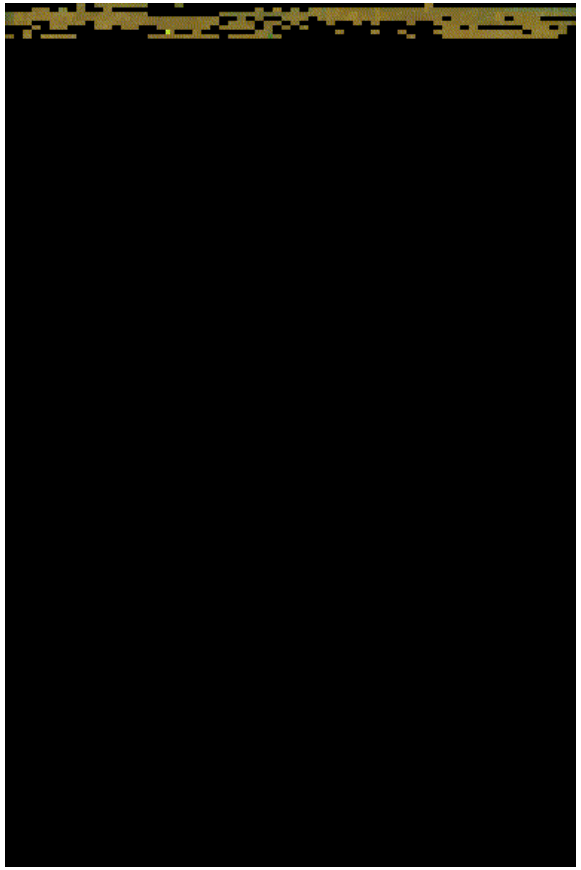


The attacker uses a faulty load to leak the data

Exploiting CacheOut

- CacheOut leaks at 2.85 KiB/s (rather than 0.1 B/s)
- Targets data at **rest**:
  - Control when we push the data into the LFB
  - Long after the victim accessed the data
- Leak AES and RSA keys
- Leak neural network weights
- Break KASLR and leak kernel data
- Even across VMs, including the hypervisor

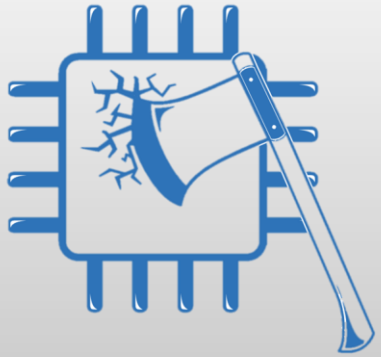
Exploiting CacheOut



Dump data from **SGX enclaves**

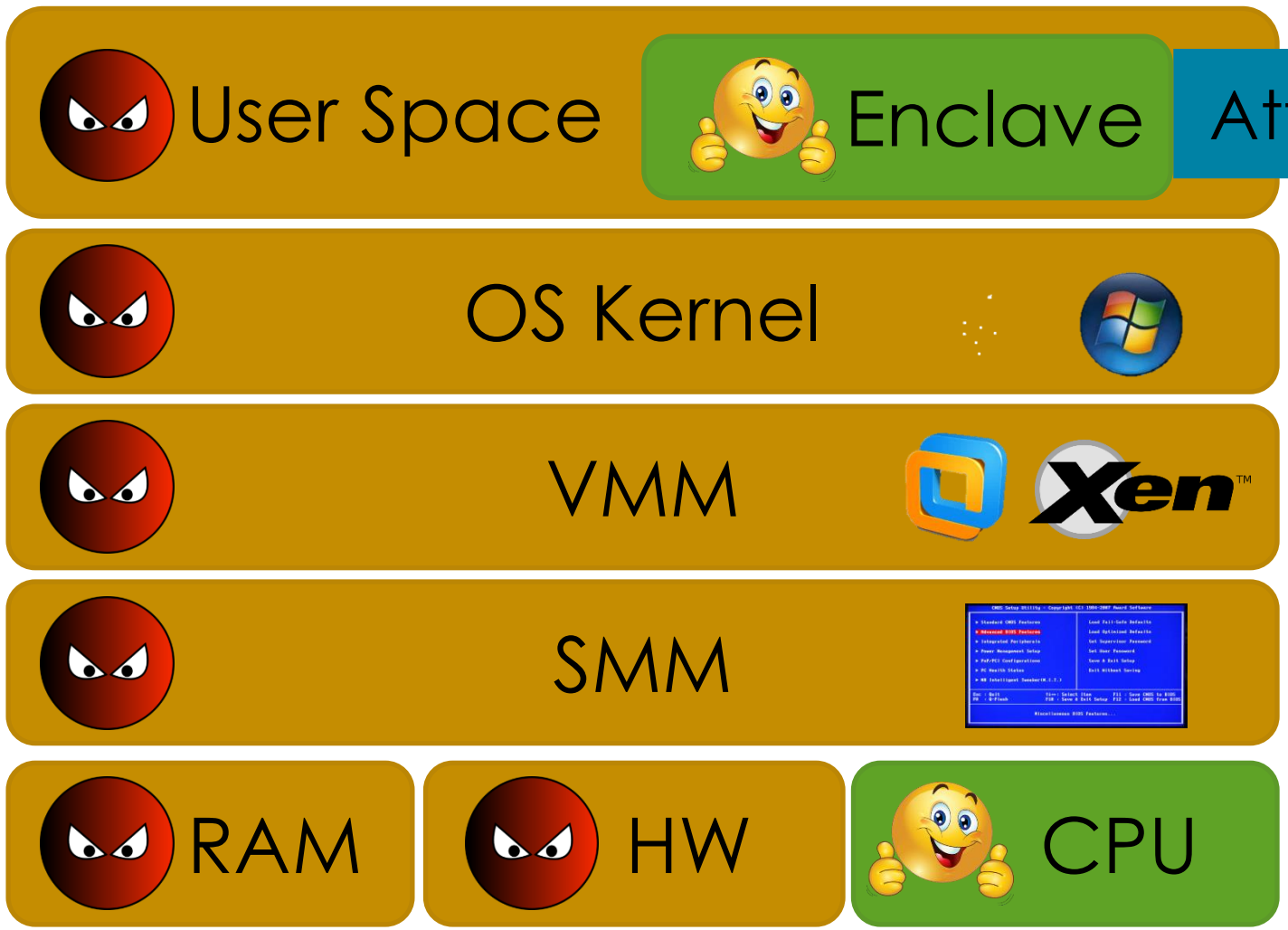
SGX based Applications

CacheOut



SGAxe

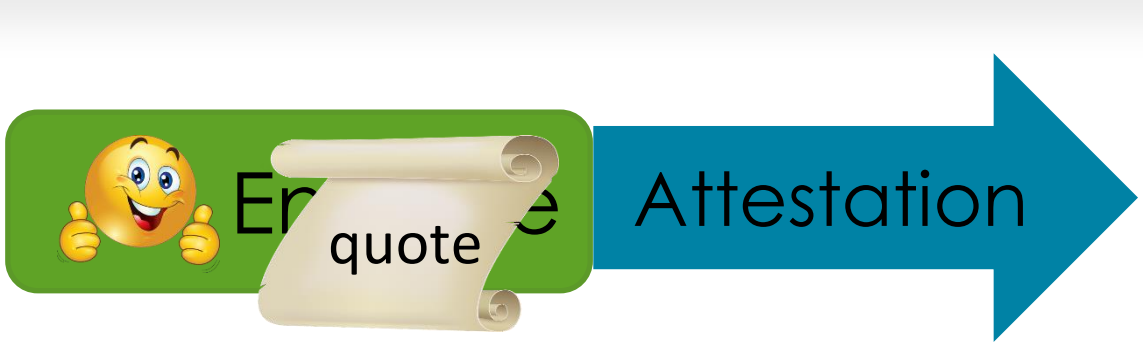
Software Guard Extensions (SGX) Security Model



Attestation

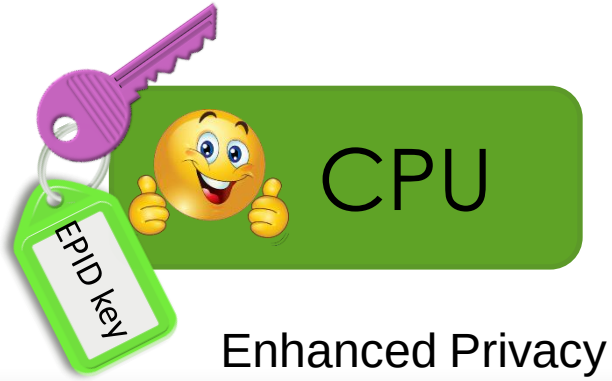
Remote Client

Software Guard Extensions (SGX) Security Model



Remote Client

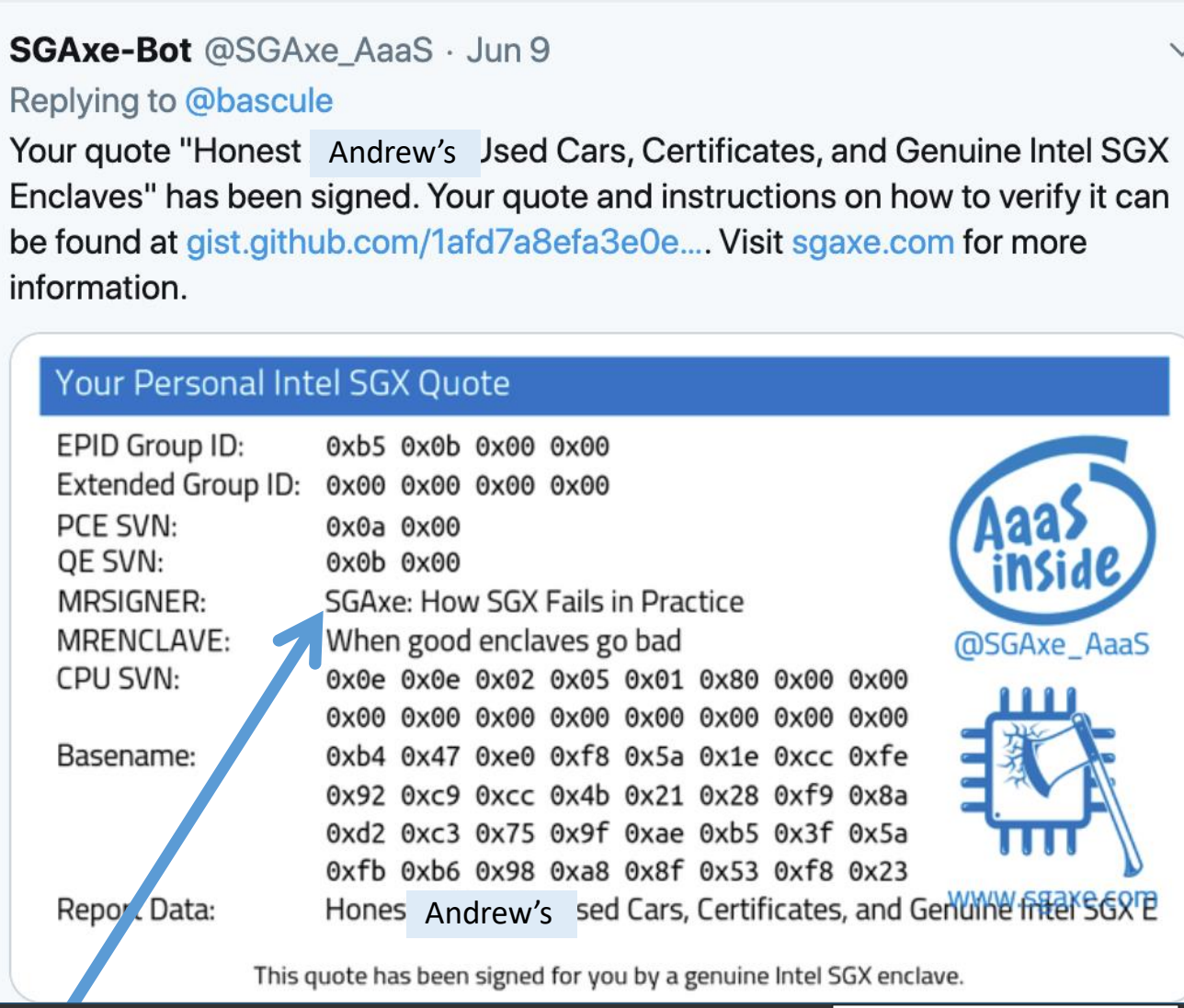
Takeaway: trust is based on the EPID key

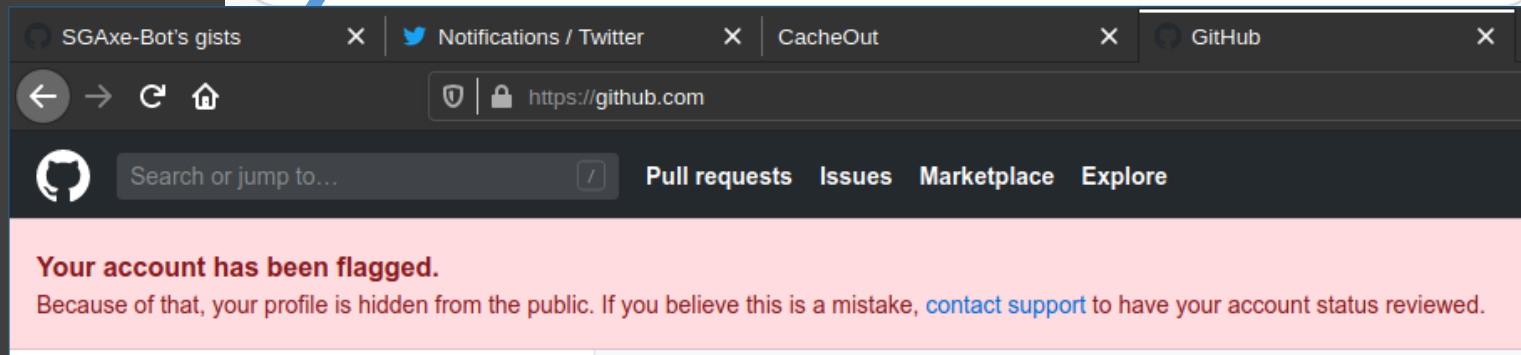


Enhanced Privacy ID

AaaS (Attestation as a Service)

- @SGAxe_AaaS
  Will attest to anything tweeted at it
- Signed 100+ quotes within 2 hours
  - Blocked by Github
- After the public release of the paper, key was still trusted for a whole month
- Can't update TCB quickly because SGX users need to install BIOS updates
- Hardcoded MRSIGNER prevents abuse





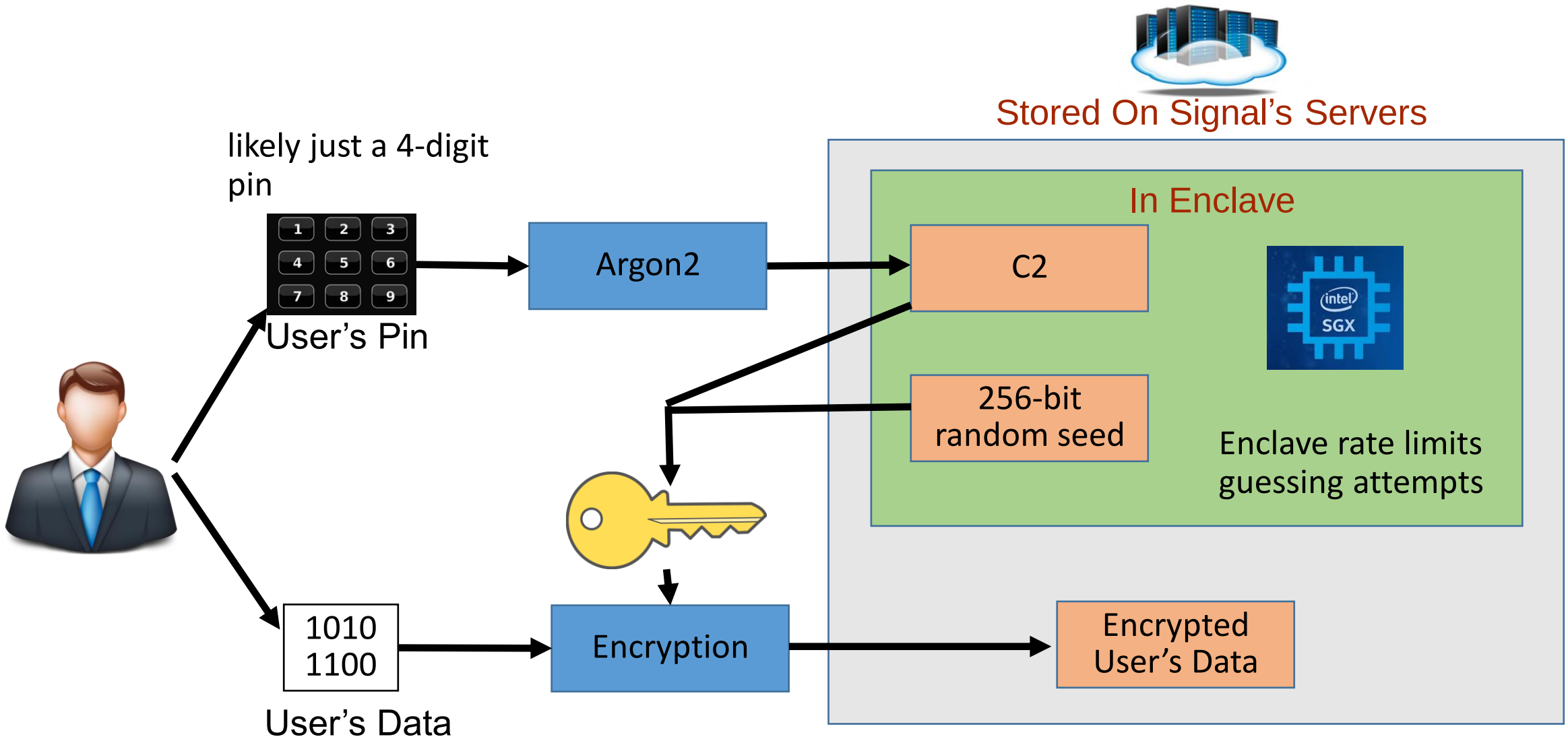
SGX in Signal

- In May, Signal started prompting users to add pins

- Secure Value Recovery (SVR)
  - Backup contacts in cloud
  - Non-phone # based addressing

- Isn’t selling point of signal is that they don’t store info about users?
  - Data is encrypted under a key derived from both the user’s pin and a random seed
  - Designed so that Signal themselves cannot decrypt the data
  - Problem: relies on SGX

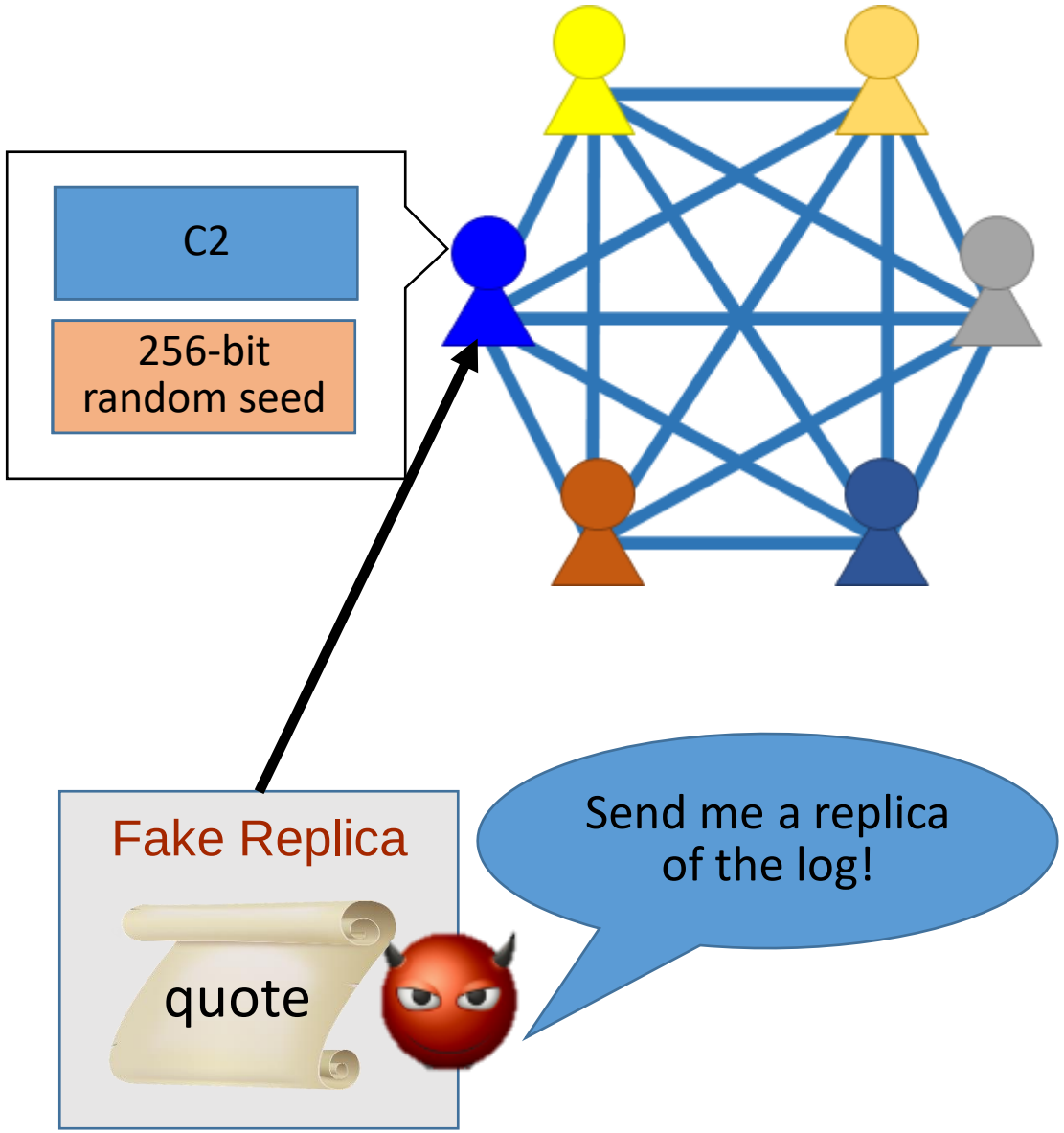
Secure Value Recovery



- Encryption key derived from both user's secret pin and a random seed
- User's data still secure even when the user's pin is short and memorable

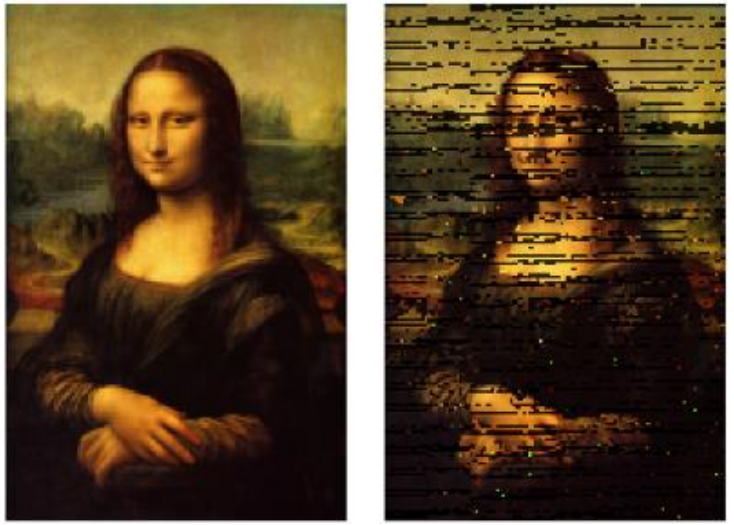
Leaking C2

- Attacker with access to the Signal servers storing C2:
  - Use CacheOut can recover C2 from L1-cache
  - Requires some side-channel knowledge
- Alternatively: exploit the trust placed in the EPID keys
- Values are replicated across a Raft cluster
  - Raft stores C2 and the random seed as a distributed log
- If cluster's network is compromised (subpoena, coercion, hacked machines, etc):
  - Attacker can then use the stolen EPID keys to forge quote proving that the malicious replica is running on SGX
    - Even though fake replica is not using SGX
  - request Raft network to replicate the entire log
  - Can then be read in plaintext



Summary

- CacheOut breaches SGX's confidentiality, allowing an attacker to masquerade as a legitimate SGX enclave
- Good that Signal is thinking about side-channels against SGX
  - insert LFENCE before each branch
  - also use retpolines to defend against speculation attacks
  - Doesn't mitigate CacheOut
- SGX cannot be relied upon to limit guessing attempts
  - Signal should still require strong passwords for SVR
- Attack assumes a malicious Signal Server with access to the cluster
  - Cannot do this at home
  - Subpoena, insider threat, hacked machines

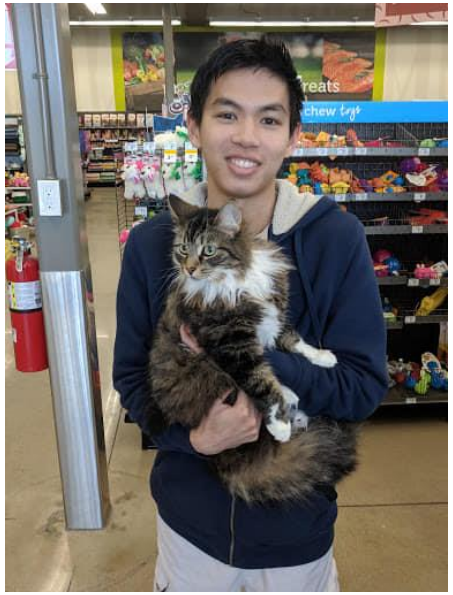


Countermeasures

- Dedicated Microcode update released on June 20th
  - Patch your machine
- More details at CacheOutAttack.com

Thank you!

Questions?



(ankwong@umich.edu)

(stephvs@umich.edu)