Prime+Scope: Overcoming the Observer Effect for High-Precision Cache Contention Attacks

spectredown (# **4**)
Prajeeth.S
(190050117)

I. Summary

In this paper, the authors have described a new cross core cache contention attack PRIME + SCOPE, that aims to improve the temporal resolution over the existing PRIME + PROBE attacks. They propose PRIMETIME, which gives an efficient way to implement accurate access patterns. Using this, they propose an improved attack on AES T-tables.

II. Details

The threat model assumes that the attacker can execute unprivileged code on the same processor as that of the victim. In particular, the PRIME+SCOPE attack has two additional requirements over the existing cache contention attacks - eviction candidate has to accurately predictable and reads on lower level cache don't affect higher level ones. The main idea is as follows - The attacker fixes the line (say S) that is going to be evicted in the shared cache using specific access patterns(given by PRIMETIME), while remaining in the private cache. Now, when the victim accesses the cache, S gets evicted from the shared cache. This leads to a sudden hike in the access time, which goes beyond a particular threshold, thus allowing the attacker to identify the event of victim's access.

The clever idea proposed to fix the line(S) for eviction is as follows - The PRIME access pattern contains regular accesses to S, while accesses to other lines are sparse. This leads to S used frequently in the private cache, while it becomes less recently used in the shared cache. So, after some time, S is evicted from the shared cache, while it remains intact in the private cache.

III. Strengths

The idea proposed leads to a preserving and concurrent measurements, hence allowing for a windowless technique. This method improves the time resolution to the order of about 25 ns. This attack does not assume any shared memory between the attacker and victim, nor the ability to interrupt or control the victim's program. Unlike other attacks where the cache has to be prepared before every measurement, here, the PRIMETIME algorithm allows one to make subsequent measurements with just one repeatable cache access.

IV. Weakness

The two main weaknesses come from the two main assumptions. This attack doesn't work if the replacement policy is not deterministic, as one cannot predict the eviction candidate accurately. Again, this doesn't work if the reads on lower level cache affect the higher ones, as the set to be evicted will not be the least recently used one in the higher level cache. The comparison between the PRIME+PROBE(PP) and the PRIME+SCOPE(PS) attacks seem unfair as they've used a pattern specific of PS to PP. This requires some sort of inclusiveness - LLC in inclusive caches and CD in non-inclusive caches.

V. Possible Extensions

One can try to extend this attack to random cache replacement policies, without affecting the multi-level cache performance.

PRIME + SCOPE

→ PRIME + PROBE – attacks that exploit cache contention.

→ PRIME + SCOPE – proposed attack, claiming to <u>increase time resolution</u> over PRIME + PROBE

⇓

→ based on deterministic nature of modern replacement policies (?), interaction across cache lines.

→ claims of achieving some resolution

→ new attack on AES Ttables

→ simple routine to construct eviction sets

Temporal resolution – metric of cache attack techniques – more the resolution, lesser the time interval – more information?

~~Solves~~ 2 challenges → observer effect
↳ time to access the lines.

claims PRIME + Scope → can be repeatable, optimal

FLUSH + RELOAD
↓
shared memory
cache line level granularity

PRIME + PROBE
↓
→ "inclusive" caches are most susceptible
→ offers only spatial granularity of sets.

vict from
but
LLC

n LLC.
PPWA,
air ?
n Ps) ?

this

t policies
ont
nes.

Why ~~windows~~ windows ?

Blind spots - period before Reload, that cannot be detected.

⇒ Introduce wait time that can possibly remove some

larger blind spot → lesser detection
→ lower resolution

How to avoid?
→ Why blind spots arise?
2 reasons - from observer effect
1) Non-preserving - reload doesn't preserve cache state
2) Non-concurrent ⇒ again, reload is non-concurrent, as the event occuring during it is missed.

PRIME + SCOPE Idea:

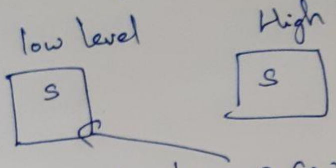
2 additional reqs:
→ Eviction candidate can be accurately predicted (⇒ explains the basis of deterministic replacement policy)
→ Reads on low level cache dont affect higher ones.

Idea:
Attacker fixes the set going to be evicted in the shared cache ~~[illegible]~~ with. a specific pattern.
Now, when victim accesses some data not in cache, ~~&~~ the set that attacker fixed would be evicted. ⇒ Now, it takes more time to get that Data. ⇒ Attacker has found out there is a victim access

R_A : high eviction rate

R_B - Install a specific line as EVC in C_S

R_B - Keep S in C_P.



regular access to line 0 (S)
less freq. accesses to lines 1 -(W-1), ⇒ evict from ~~low~~ low level, but recently used in LLC

⇒ S eventually gets old in LLC.

Question on (6)(iii) figure - Prepare for PPwA, PPwB uses Prime patterns - Is it fair ? (as this is developed from Ps) ?

Disadvantages:

→ If the 2 reqs dont hold, then this wont work

→ Wont work for random replacement policies

Wont work if low level caches dont act as filter for higher ones.

Why ~~wi~~

Blind

⇒ I

large

How

→ (

2

1)

2)

PRIME +

2 a

→

of

→

Idea:

A

evic

Spe

N

no

fix

mo