Defeating the Recent AnC Attack by Simply Hashing the Cache Indexes
— Implemented in a BOOM SoC

Wei Song, Rui Hou, Dan Meng
Institute of Information Engineering, Chinese Academy of Sciences
89A Minzhuang Road, Haidian District, Beijing, China 100093
{songwei, hourui, mengdan}@iie.ac.cn

The AnC Attack

The AnC attack[1] is a principled way to bypass the address space layout randomization (ASLR) defense in all major browsers by utilizing existing side-channels on memory management unit (MMU) and caches.

Attack Scenario:

The attacker is a JITed thread running in a browser sandbox protected by ASLR.

- The attacker CAN access a large amount of vitual memory (data/code).
- The attacker runs in the SAME process with the browser (same ASID).
- The attacker runs on the SAME core with the browser (share L1/LLC).
- The attacker does NOT know the virtual address of her data/code.
- ALL virtual pages are randomized by the ASLR defense.

Attack Target:

Infer the virtual address (VA) of a target variable belonging to the attacker (bypassing ASLR).

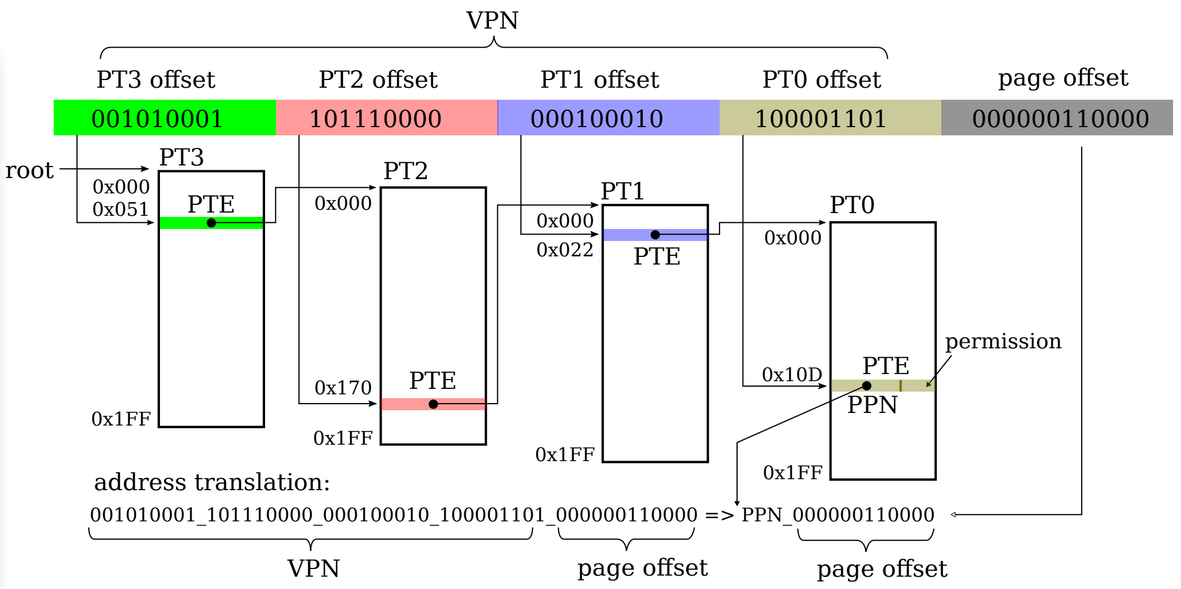
Attack Procedure:

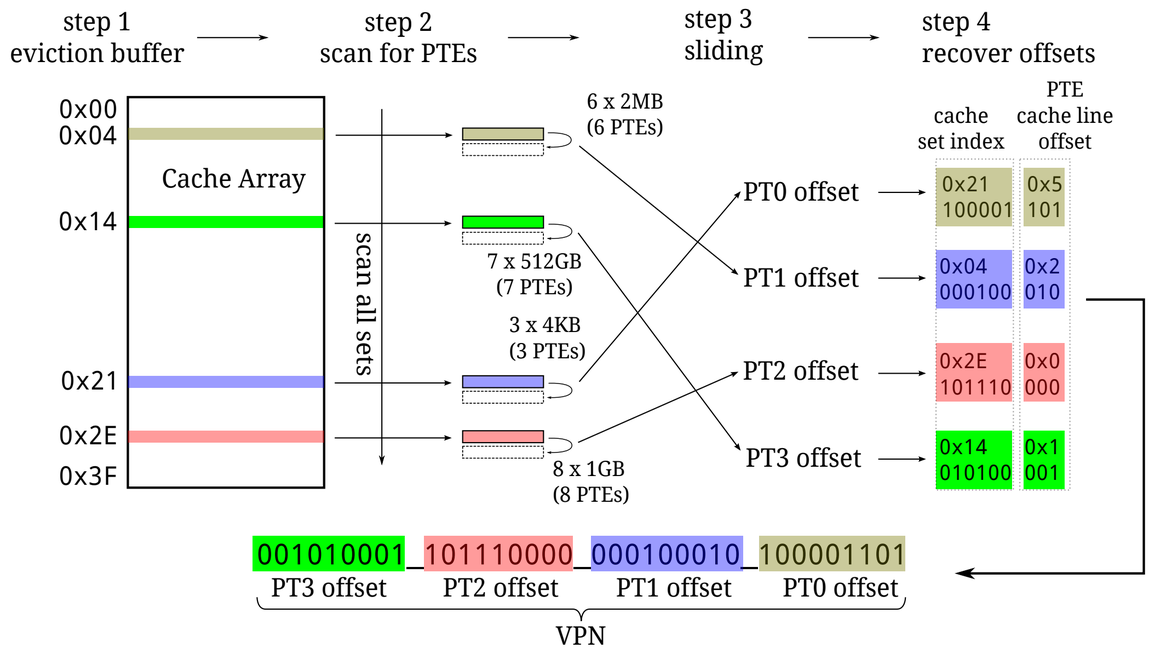
*For a target variable **v**, infer all PT (page table) offsets using cache side-channel attacks on the 4 cache PTEs (page table entries).*

Step 1: Construct an eviction buffer.
Step 2: Scan all cache sets to identify the 4 cached PTEs related to ***v***.
Step 3: Use the sliding technique to map PTEs to PT levels.
Step 4: Use the sliding technique to recover the PTE offsets inside cache lines.

Key Insights:

- Caching PTEs with data is a security vulnerability.
- Existing cache partitioning does not protect PTEs from side-channels.
- Knowning the cache set index is able to decipher page offsets.
- The direct mapping between VA and cache indexes is a prerequisite for the attack.





Solution

Break the direct mapping between VA and cache indexes using a simple hash.

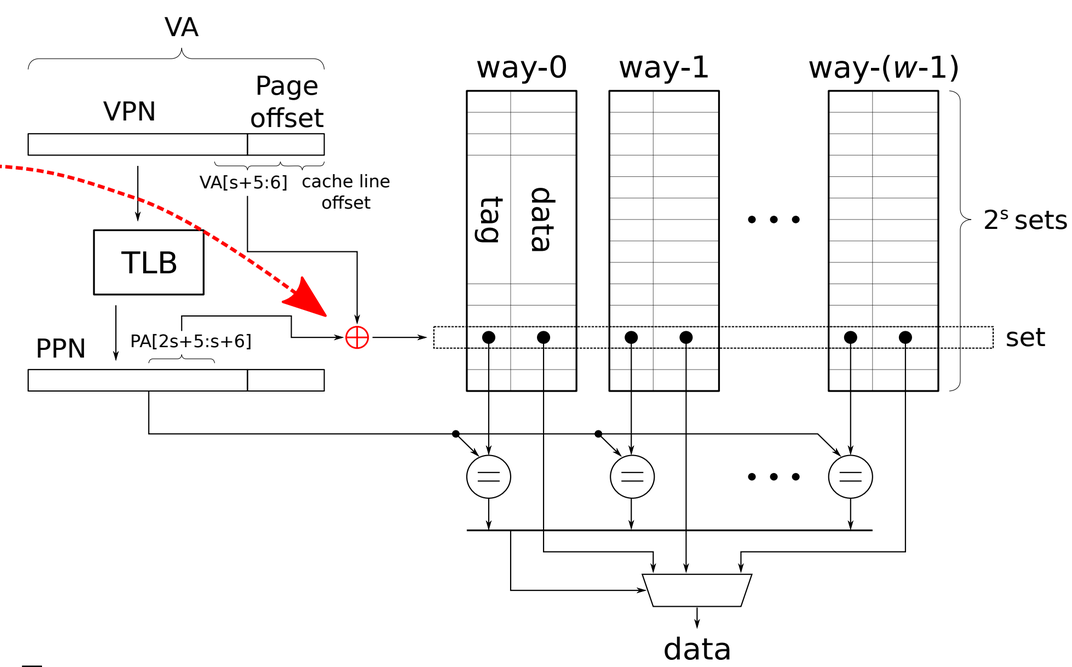
The original cache index:

CI = VA[s+5:6]

The hashed cache index:

CI = VA[s+5:6] ⊕ PA[2s+5:s+6]

CI: cache set index, 2^S: number of sets



Defense Methodology:

- Cache should be transparent to software; therefore, software should not infer or rely on the direct mapping.
- Without the direct mapping, it is difficult to constrcut an eviction buffer.
- Without knowing the PA, it is difficult to infer VA from cache set index.
- Cache lines with the same page offset are likely mapped to different sets.
- PA is normally unknown to user mode programs.

Prerequisites:

- PA is not exposed to user mode programs.
- OS constantly allocates random physical pages to consecutive virtual pages.
- OS disable the huge page support.

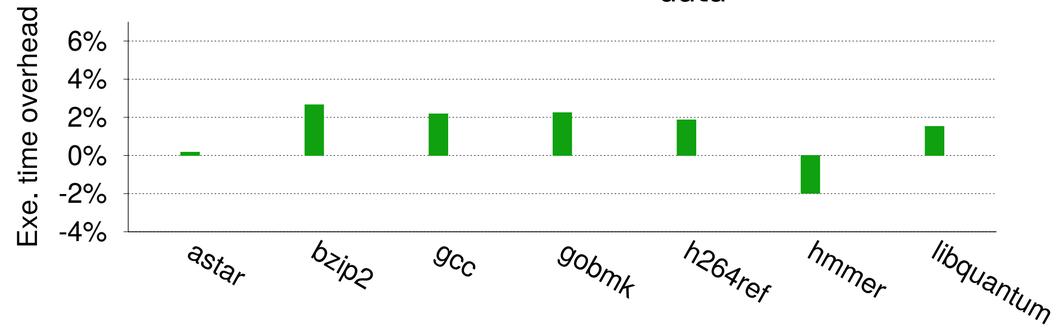
Implementation:

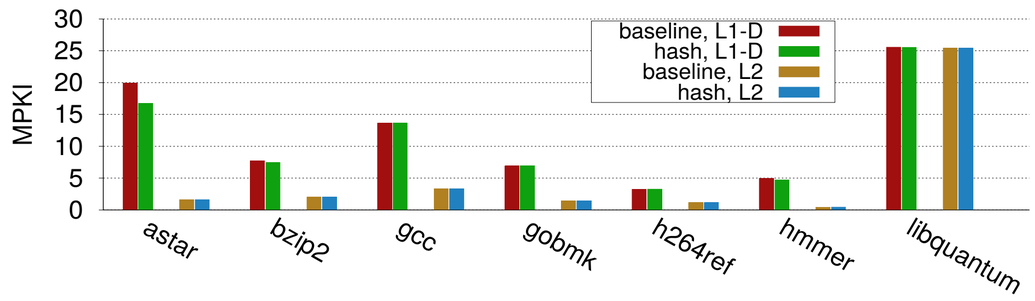
The proposed scheme is implemented in a BOOM SoC[2]. Both the L1 and L2 (LLC) caches are modified with the hash scheme.

Performance overhead:

Running the SPECInt 2006 benchmark on FPGA.

- Averge 1.0% increase in execution time.
- Marginal increases in the cache miss rates (L1 < 4%, L2 <0.5%).
- Marginal increases in cache area (<0.1%).





References:
[1] B. Gras, K. Razavi, E. Bosman, H. Bos, and C. Giuffrida. "ALSR on the line: Practical cache attacks on the MMU," In *Proc. of the Network and Distributed System Security Symposium*, 2017, p. 15.
[2] C. Celio, D. A. Patterson, and K. Asanovic, "The Berkeley out-of-order machine (BOOM): An industry-competitive, synthesizable, parameterized RISC-V processor." EECS Department, University of California, Tech. Rep. UCB/EECS-2015-167, 2015.