Survey of Hardware Security in CPU and GPGPU

Meet Udeshi
14D070007
Supervisor: Prof. Virendra Singh

CADSL - IIT Bombay

February 22, 2019

Contents

Introduction

- Multi-thread, multi-core hardware focus on sharing computing resources among different programs.
- In cloud, VM and sandbox technology is used as software level security measure. But they share the same hardware.
- Programs running on current hardware inevitably leave fingerprints of their execution in power traces, EMI spectrum, caches, etc.[1]
- Attackers with knowledge of hardware can obtain these fingerprints and extract sensitive information.

[1] [Chenglu Jin, Side Channel Attacks]

Side Channels

Shared resources of the processor can leak information about the tasks being performed in it. Measuring cache hit/miss, time of execution, power consumption, EMI spikes can determine what part of code is running or what data is being processed

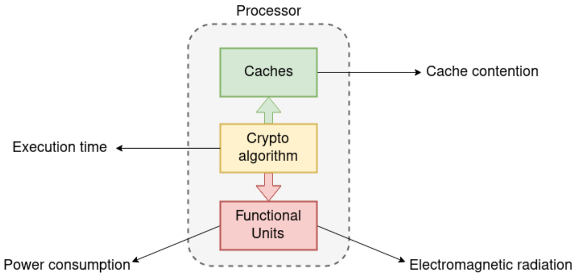

Figure: A number of side channels which are capable of leaking data.

Data dependent execution

Fast exponentiation algorithm used in RSA[2]

```
while(key > 0) {
    e = key % 2;

    Square();
    Reduce();
    if(e == 1) {
        Multiply();
        Reduce();
    }

    key >>= 1;
}
```

S-box access used in AES[3]

```
for(i=0; i<9; i++) {
    x = k;
    y = k ^ n;

    e = {S[x[1]]^1, S[x[2]],
            S[x[3]], S[x[0]]};
}
// Just for last round
y[0] = S[y[0]]^S[y[1]]^
         S[y[2]]^S[y[3]]^
          x[0];
```

[2][C. Percival, Cache missing for fun and profit]
[3][D. J. Bernstien, Cache-timing attacks on AES]

Cache Side Channel

1. Determine the cache-line accessed by victim using contention in shared cache.
2. In set-associative cache design, that reveals a few bits of the address.
3. For publicly available encryption libraries (OpenSSL), determine memory region of victim accesses.
4. Using both these information, we can pinpoint exact address accessed by victim.

Prime+Probe

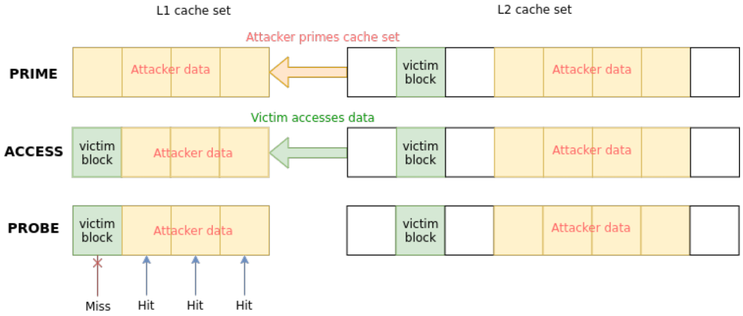

Figure: Working of Flush+Reload attack

Flush+Reload

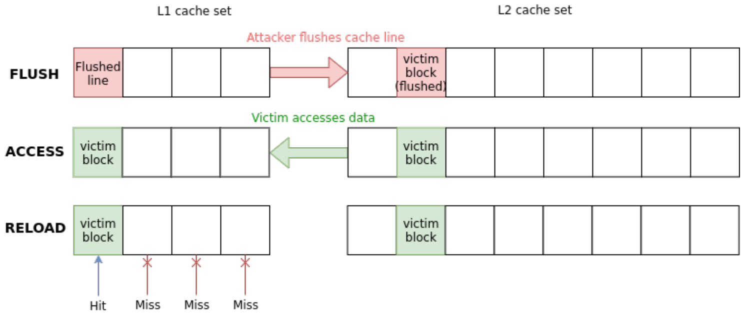

Figure: Working of Flush+Reload attack

Flush+Reload on LLC

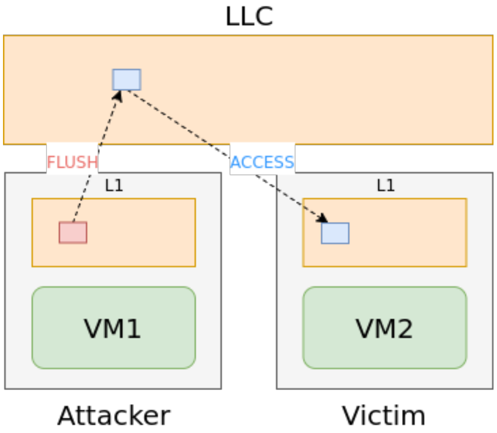

Figure: Cross-VM Flush+Reload Attack via LLC

Virtual Machine environments are extensively used in cloud-computing platforms to enable multiple different users share processors. VMs share a common host OS which tries to optimise memory usage by performing page de-duplication. This allows VMs to share memory pages and hence enables hardware based side channels[4].

[4][Wait a minute! A fast, Cross-VM attack on AES - RAID'14]

Cache side channels on GPGPU

Nvidia GPGPUs have recently started to support concurrent kernel execution at SM level, which allows multiple programs to simultaneously use the GPGPU resource. In this shared context, one must look at side channels which can be exploited.

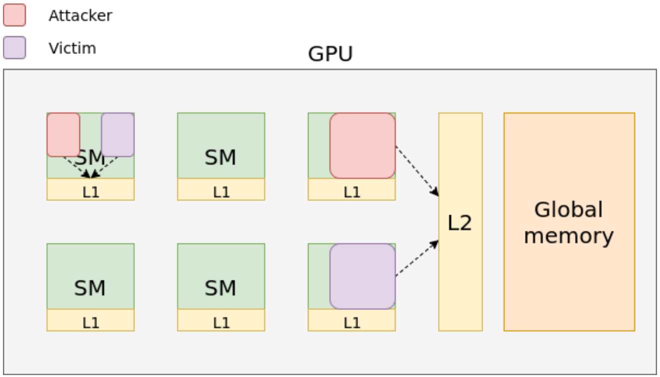

Figure: Memory layout of GPGPUs

GPGPU covert channel

Naghibijouybari et al in [5] achieve communication speed of over 4Mbps using a combination of L1 cache contention and SFU contention as covert channels on multiple Nvidia GPGPU architectures. They have used the inherent parallelism in GPGPUs to multiply the speed of the created covert channels by opening parallel communication channels on each SM.

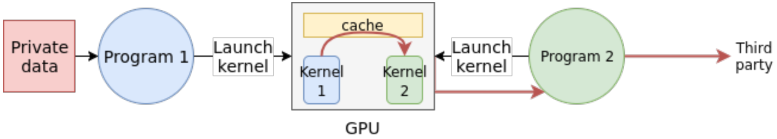

Figure: Covert channel implementation on GPGPU

[5][Constructing and characterizing covert channels on GPGPUs - MICRO-50]

Reverse engineer cache parameters

Knowing cache parameters is necessary for mounting any kind of side channel attacks on caches. This implementation of reverse engineering cache parameters is based on [6]. In that paper, Wong et al show how using a stride access pattern over an array to trigger a predictable number of cache-misses.

Listing 1: Generate array of given size with fixed stride pattern

```
size_t* array; // malloc beforehand
size_t t;

for (int i=0; i<array_size; i++) {
    t = i + STRIDE:
    if(t >= array_size) t %= STRIDE;
    array[i] = (size_t)array + sizeof(size_t)*t;
}
```

[6][Demystifying GPU microarchitecture through microbenchmark - ISPASS]

Reverse engineering cache parameters

For measuring timing of the array access, `rdtsc` instruction is used to get a reading of the Time Step Counter before and after accessing the array. The listing shows how to traverse the array using the stride access data stored in it.

Listing 2: Timing measurement of stride access over the entire array

```
long start = __rdtsc();
size_t* next_ptr = &array[0];
for(int i=0; i<MAX_ITERS; i++) {
    next_ptr = *((size_t**)next_ptr);
}
long time = __rdtsc() - start;
```

Result Plots

Setup: GEM5 x86 atomic CPU. L1 data cache 1KB, 2-way, 64B line size.

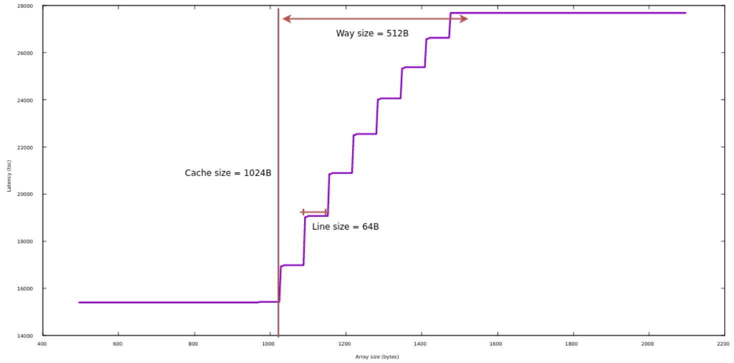

Figure: Latency vs Array Size plot

Result Plots

Setup: GEM5 x86 atomic CPU. L1 data cache 16KB, 4-way, 64B line size.

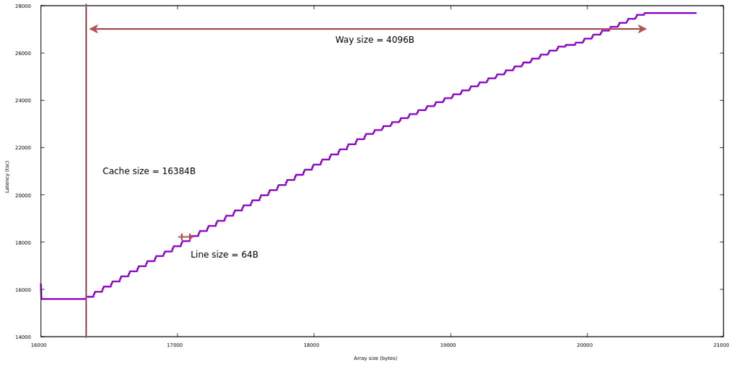

Figure: Latency vs Array Size plot

Result Plots

Setup: Intel Skylake x86_64 i5-6500. L1 data cache 32KB, 8-way, 64B line size.

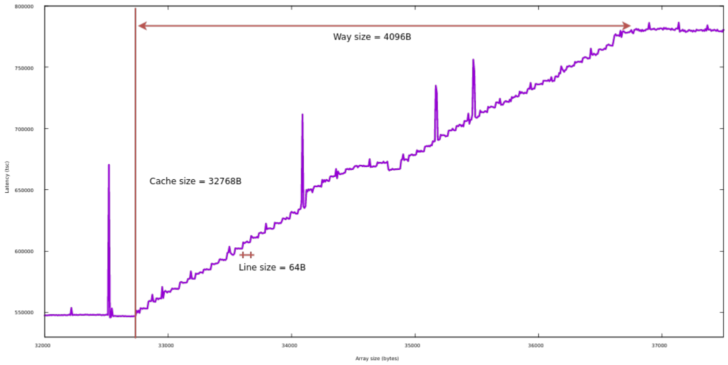

Figure: Latency vs Array Size plot

Mitigations against cache side channel

- Change the implementation of each encryption algorithm to avoid leaks.
- Only possible for known attacks and unavoidable for any software to not leave fingerprint in shared resources.
- Change hardware design of caches so that one process doesn't affect other processes via its cache accesses in a predictable way.
- Intentionally pollute cache to add noise in side channel measurements.

Partition-Locked cache

Naive cache partitioning would equally split cache for all threads. Such a partitioned cache will let only one process access a single partition at a time [7]. Wang et al have proposed a dynamically partitioned cache in [8].

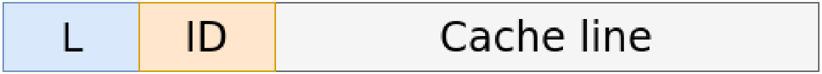

Figure: Partition-Locked cache - single cache line

PLCache requires addition of special locked-load/store instructions to the ISA. This also means OS has to look over which process gets to use them as a fairness measure.

[7][D. Page - Partitioned Cache Architecture as a Side-Channel Defence Mechanism]

[8][New Cache Designs for Thwarting Software Cache-based Side Channel Attacks - ISCA'07

Random-Permutation Cache

Random-permutation cache is another cache design proposed by Wang et al. They have added a redirection step in the address decoder of caches which uses a random permutation table.

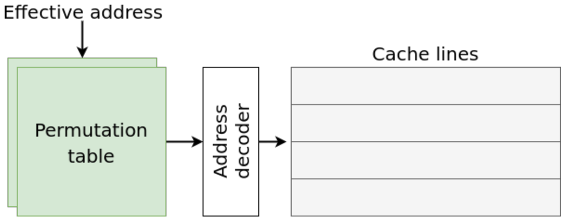

Figure: Random-permutation cache

RPcache is able to mark data using "Lock" bits by copying page protection bit. Drawback of RPCache is the added step in address decoding. 1 or 2 cycle latency increase will drastically change performance of L1.

Disruptive Pre-fetching

- Fuchs et al in [9] introduce additional steps to the pre-fetchers to increase the randomness in the loaded memory address.
- Randomise the pattern sequence and stride value to intentionally pollute the cache with unnecessary data.
- Slightly degrades the performance of non-malicious programs, but terribly disrupt side channel.
- Prime+Probe attack will not know whether the victim or the pre-fetcher evicted its block.
- Flush+Reload would get false cache hits which were not caused by the victim.

[9][Disruptive prefetching: impact on side-channel attacks and cache designs - SYSTOR'15]

Context Sensitive Decoding

A lot of modern processors use decoders to convert from ISA to an internal instruction representation. Taram et al explore in [10] if a custom decoder can insert decoy instructions to improve security.

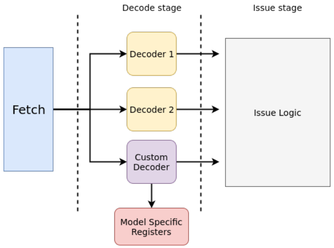

Figure: Custom decoder for Context Sensitive Decoding

Decoy instructions pollute the cache by running decoy loads and disrupt attackers attempting side channel attacks on the cache.

[10] [Mobilizing the Micro-Ops: Exploiting Context Sensitive Decoding for Security and Energy

Hypothetical ROB based Side Channel

- Reorder buffer is used by Out-of-Order cores for in-order retirement.
- In SMT cores, fairness policy ensures that roughly equal instructions are retired for all threads to maintain equal IPC.
- Simple implementation is to share retire pipeline width equally among threads.
- But when one thread is stalled, other thread will use full pipeline width to retire.
- IPC of other threads will see a slight increase.
- Data-dependent branch misses and cache misses can be infered from IPC information.

Hypothetical ROB based Side Channel

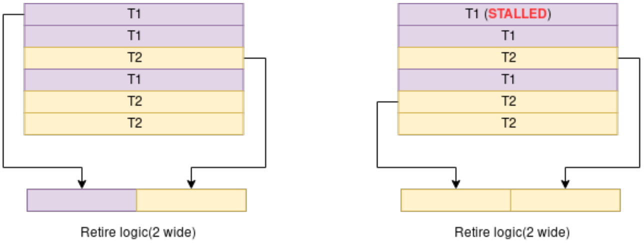

Figure: Reorder buffer for SMT. When T1 is stalled T2 retires twice as many instructions.

Conclusion and future work

- Modern hardware features have introduced leakages in the processor which is a security concern.
- GPGPUs are being targeted as a new domain of security leaks.
- Hardware design needs to take into account these kind of security leaks.
- Explore other shared resources apart from caches as potential leakage sources.

The End

For Further Reading I

- Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel , D. Page, Department of Computer Science, University of Bristol
- Cache Missing for Fun and Profit, Colin Percival
- Cache-timing attacks on AES, D.J. Bernstien
- Side Channel Attacks, Chenglu Jin, Department of Electrical & Computer Engineering, University of Connecticut
- Gorka Irazoqui, Mehmet Sinan Inci, Thomas Eisenbarth, Berk Sunar, Wait a minute! A fast, Cross-VM attack on AES
- D. Page, *Partitioned Cache Architecture as a Side-Channel Defence Mechanism*
- Z. Wang and R. B. Lee, New Cache Designs for Thwarting Software Cache-based Side Channel Attacks, ISCA'07

For Further Reading II

- Z. Wang and R. B. Lee. A novel cache architecture with enhanced performance and security. In Proceedings of the 41st Annual International Symposium on Microarchitecture, MICRO 2008
- Adi Fuchs and Ruby B. Lee. 2015. Disruptive prefetching: impact on side-channel attacks and cache designs. In Proceedings of the 8th ACM International Systems and Storage Conference (SYSTOR '15)
- Hoda Naghibijouybari, Khaled N. Khasawneh, and Nael Abu-Ghazaleh. Constructing and characterizing covert channels on GPGPUs - MICRO'50
- Henry Wong, M. M. Papadopoulou, Maryam Sadooghi-Alvandi, and Andreas Moshovos. 2010. Demystifying GPU microarchitecture through microbenchmark - ISPASS'10

For Further Reading III

Mohammadkazem Taram, Ashish Venkat, Dean Tullsen, Mobilizing the Micro-Ops: Exploiting Context Sensitive Decoding for Security and Energy Efficiency - ISCA'18