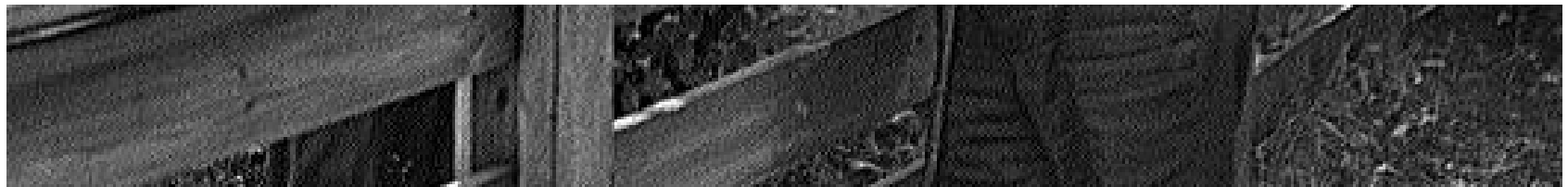




The Bad Neighbor



Hardware Side Channels in Virtualized Environments



Who I am

- **Sophia D'Antoine**
- **Security Researcher at Trail of Bits**
- **Masters in Computer Science from RPI**
 - **Thesis on Hardware Side Channels**
- **I play CTF**



Overview

- **Side channel attacks**
- Side channel mediums: Hardware
- Vulnerabilities in the cloud
- Demo
- Defenses
- The future

What are side channel attacks?

Attacks:

Leak information from or alter behavior of a system or process.

Side channel:

Attack does not alter or introspect into the program itself.

What are side channel attacks?

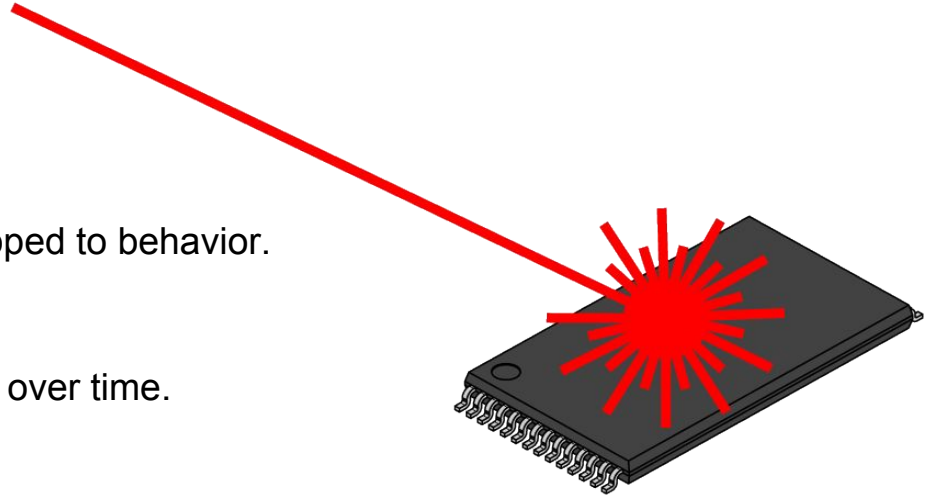
- Attacker can observe the target system. Must be 'neighboring' or co-located.
- Ability to *repeatedly* query the system for leaked artifacts.

Artifacts: changes in how a process interacts with the computer

Variety of Side Channels

Different target systems implies different methods for observing.

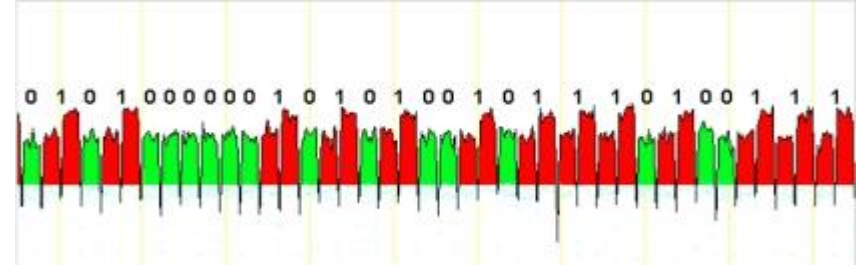
- **Fault attacks**
 - Requires access to the hardware.
- **Simple power analysis**
 - Requires proximity to the system.
 - Power consumption measurement mapped to behavior.
- **Different power analysis**
 - Requires proximity to the system.
 - Statistics and error correction gathered over time.
- **Timing attacks**
 - Requires same process co-location.
 - Network packet delivery, cache misses, resource contention.



Example targeting Cryptography

“In cryptography, a side-channel attack is any attack based on information gained from the physical implementation of a cryptosystem”

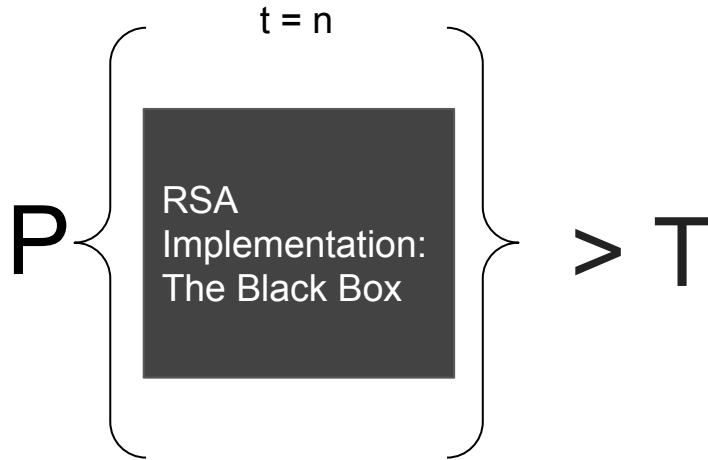
- Medium matters
- Attackers have ability to measure system as black box



RSA bits leaked through Power Analysis

What happens

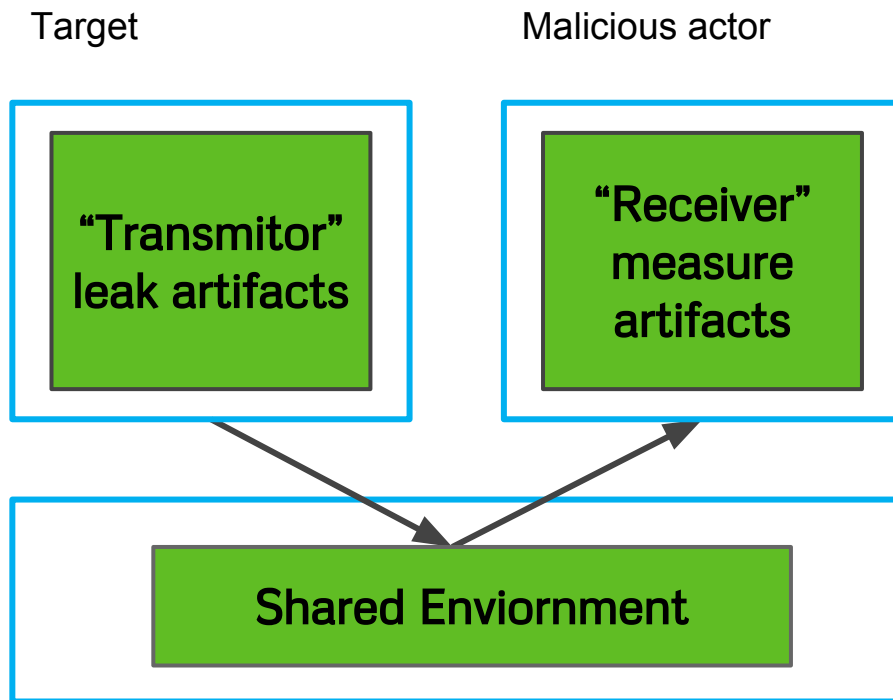
Information gained through recordable changes in the system



Powered sampled
at even intervals
across time.

Basic side channel requirements

- Medium agnostic
- Side channels require 3 primitives:
 - Transmit
 - Receive
 - Shared environment



Types of side channel attacks: Scenario 1

Receiver (Eavesdropper):

Record information from the shared environment.

Transmitter: Unaware target. Operating as normal.

Applications include:

- crypto key theft
- process monitoring
- environment keying
- broadcast signal

Types of side channel attacks: Scenario 2

Receiver (Unaware):

Non-existent. All other processes on system do not intentionally record artifacts.

Transmitter (Aware): Intentionally sends artifacts into the shared environment.

Applications include:

- DoS attack
- Proof of co-residency



Types of side channel attacks: Scenario 3

Receiver:

Records system artifacts and translates into a meaningful message.

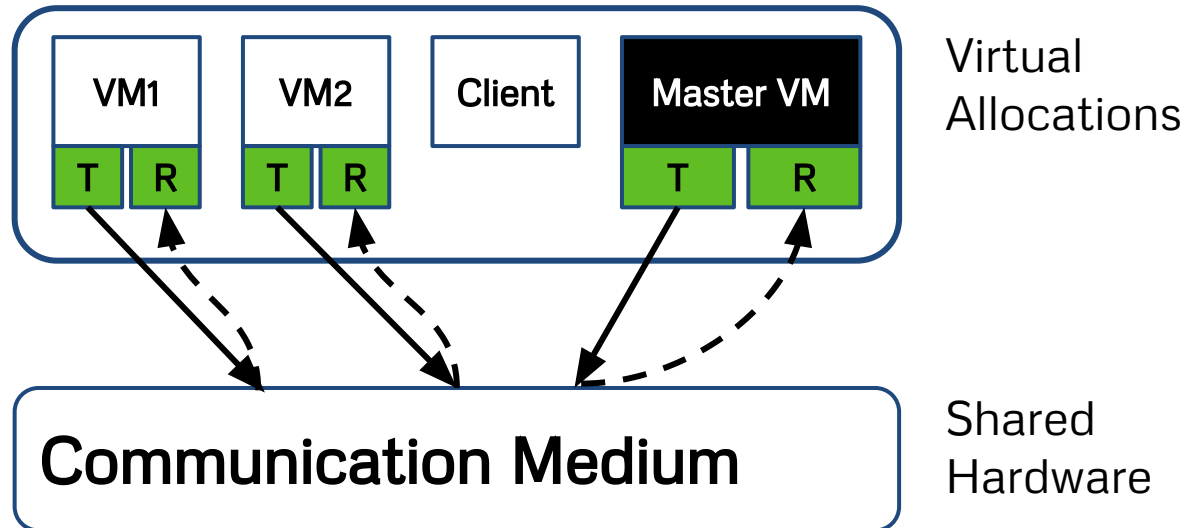
Transmitter:

Intentionally sends artifacts into the shared environment.

Caveats: All processes to have both and agree on time, pre-arrange message translation.

Types of side channel attacks: Scenario 3

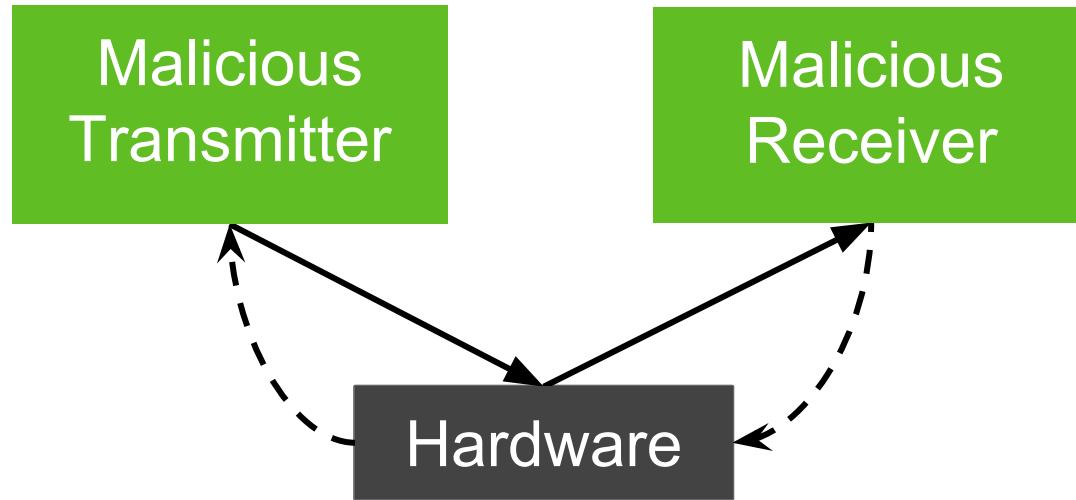
Applications Include: Communication channel.



Overview

- Side channel attacks
- **Side channel mediums: Hardware**
- Vulnerabilities in the cloud
- Demo
- Defenses
- The future

Communication Between Processes Using Hardware



Available Hardware

Shared environment on computers, accessible from software processes.
Hardware resources shared between processes.

- Processors (CPU/ GPU)
- Cache Tiers
- System Buses
- Main Memory
- Hard Disk Drive

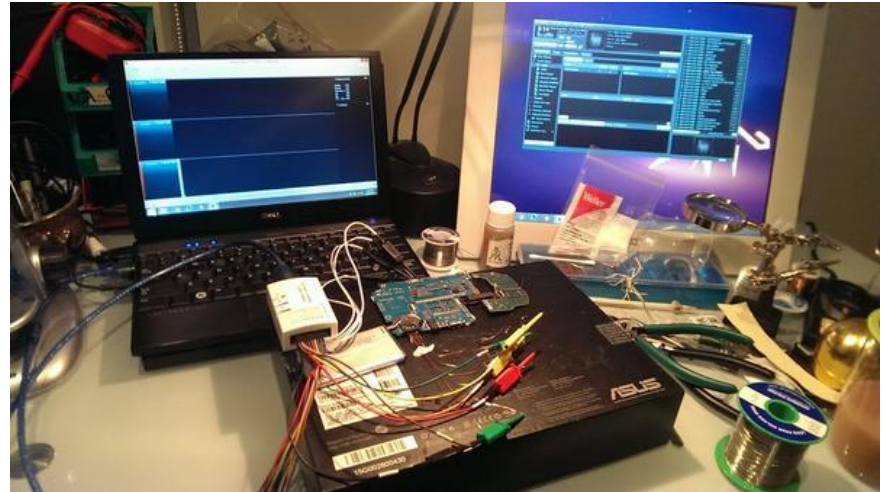
Hardware side channels compared to other types

Sender (Transmitter) process

- Affect the state of the shared hardware
- Must be observable/ recordable from other processes
- Repeatable

Receiver process

- Record the state of the shared hardware
- Must observe without affecting the transmitted state.

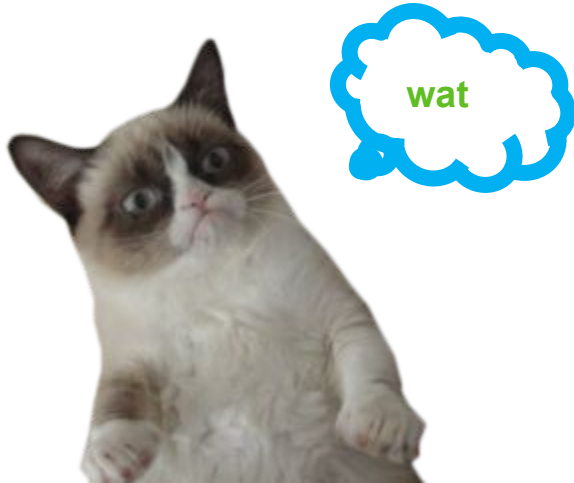


Side Channel attacks over Hardware

Primitives

- Processes share hardware resources
- Dynamic translation based on need
- Allocation causes contention

Physical co-location leads to side channel vulnerabilities.



The Cloud



Overview

- Side channel attacks
- Side channel mediums: Hardware
- **Vulnerabilities in the cloud**
- Demo
- Defenses
- The future

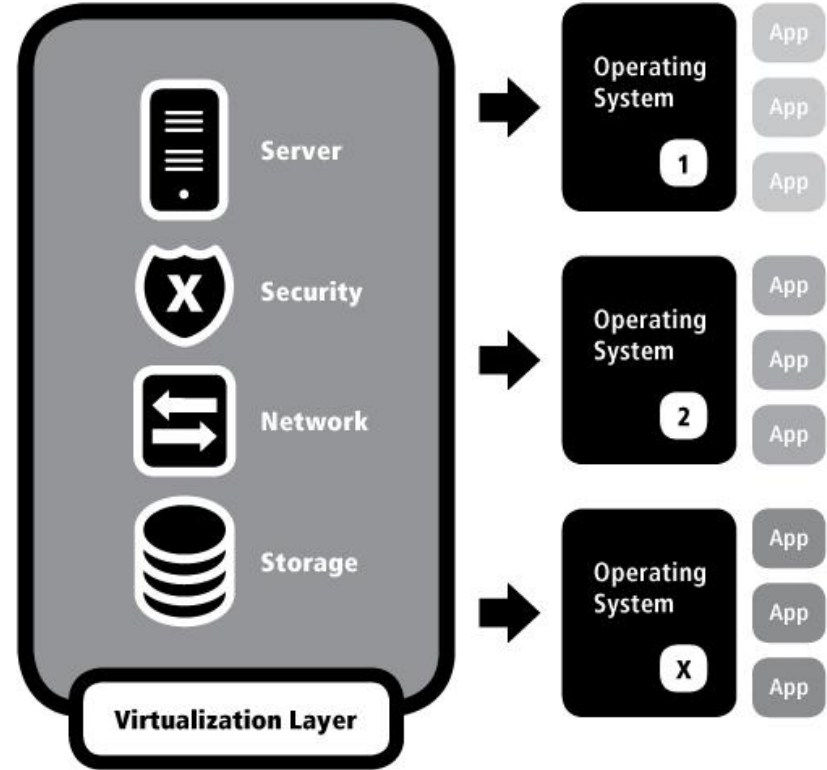
Cloud Computing (IaaS)

Perfect environment for hardware based side channels:

- Virtual instances
- Hypervisor schedules resources between all processors on a server

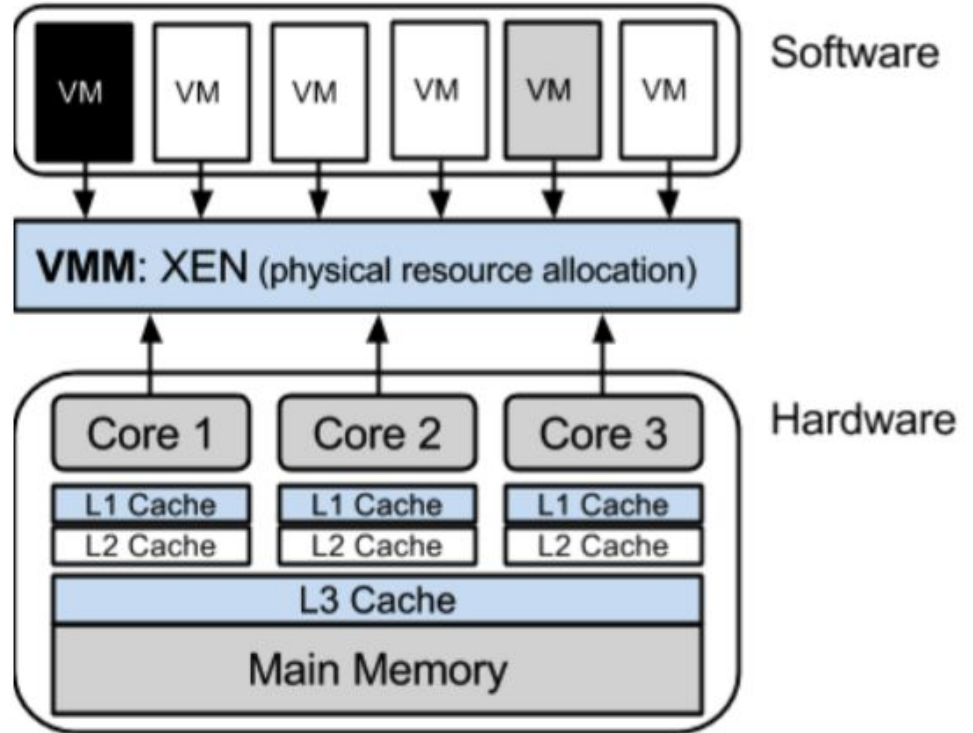
Dynamic allocation

- Reduces cost



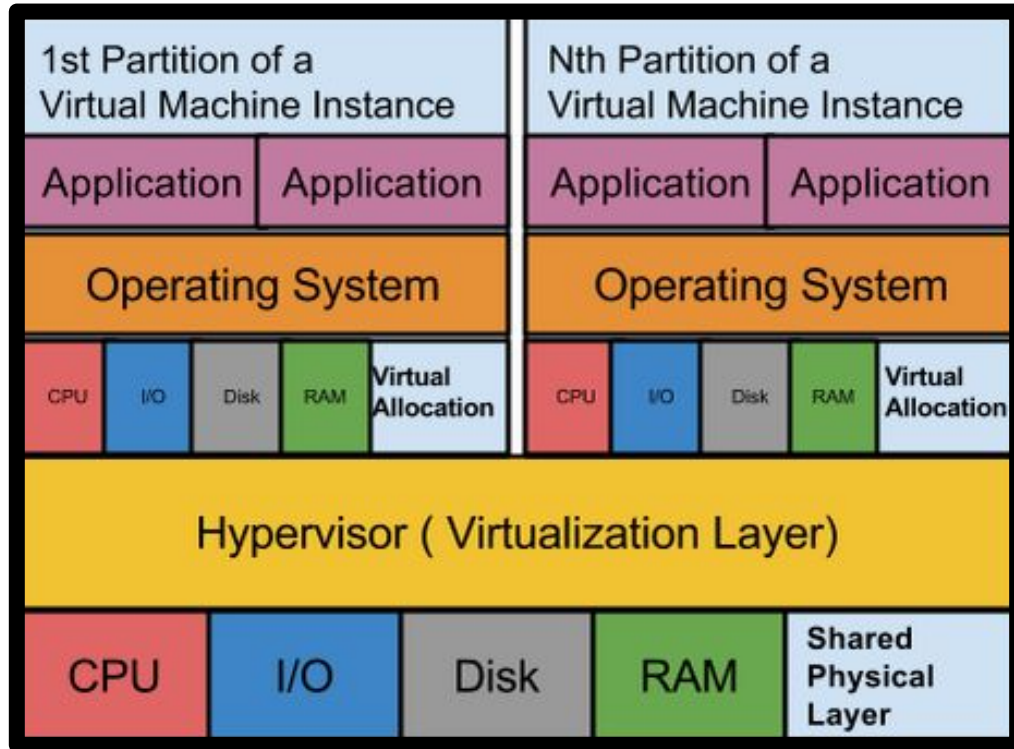
Vulnerable Scenarios in the Cloud

- Sensitive data stored remotely
- Vulnerable host
- Untrusted host
- **Co-located with a foreign VM**



Neighbor virtual machines

Sharing hardware allocations.



Hardware Side Channels in the Cloud

Cloud Computing Side Channel - Primitives

Medium: Shared artifact from a hardware unit

Cross VM: Virtual machine or process

Method: Information gained through recordable changes in the system

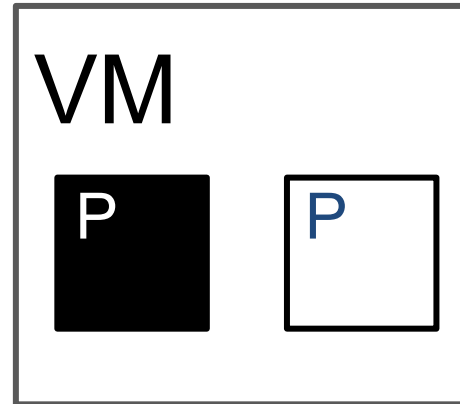
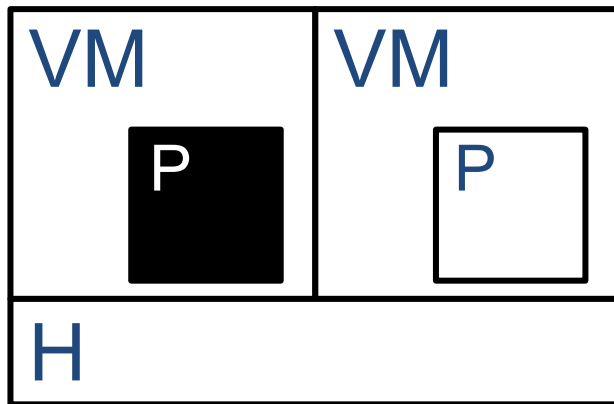
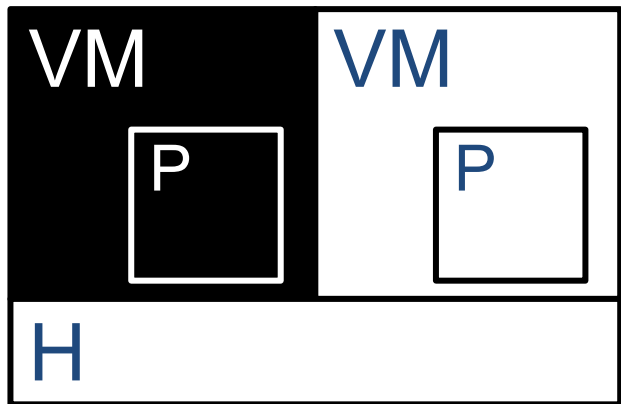
Vulnerability: Translation between physical and virtual, dynamic!

Cloud Computing Side Channel

Shared hardware

Dynamically allocated hardware resources

Co-Location with adversarial VMs, infected VMs, or Processes (requires SMT)



Build Your Own Side Channel: Hardware

Choose Medium: Measure shared hardware unit's changes over time

- Cache
- Processor
- System Bus
- Main Memory
- HDD



Build Your Own Side Channel: Measurement

Choose Vulnerability: Measure artifact of shared resource.

- Timing attacks (usually best choice)
 - Cache misses, stored value farther away in memory
- Value Errors
 - Computation returns unexpected result
- Resource contention
 - Locking the memory bus
- Other measurements recordable from inside a process, in a VM

Build Your Own Side Channel: Attack Model

Choose S/R Model: What processes are involved in creating the channel depend on intended use cases.

- Scenario 1: Transmit only
 - Application: DoS Attack
 - Sender only
- Scenario 2: Record Measurements
 - Application: Crypto key theft
 - Receiver only
- Scenario 3: Bi-way Channel
 - Application: Communication channel
 - Sender and Receiver

Some channels are easier than others....

Case Study 1: Locking the memory bus

- Pro: efficient, no noise, good bandwidth
- Con: highly noticeable

Case Study 2: Everyone loves Cache.

- Pro: hardware medium is 'static'
- Con: most common, mitigations are quickly developed

Some channels are easier than others....

Technical Difficulties:

- Querying the specific hardware unit
- Difficulty/ reliability unique to each hardware unit
- Number of repeated measurements possible
- Frequency of measurements allowed

Measurement methods for different hardware units

Hardware Medium	Transmitting Mechanism	Reception Mechanism
Processor	Processor Register and Functional Unit Resources Contention	Time Compared Against Threshold
Cache Tier	Prime-Probe, Shared Cache Functionality	Time Compared Against Threshold
System Bus	System Bus Restricted Access Contention	Measurement of Memory Access Capabilities
Main Memory	Prime-Probe, Shared Main Memory Storage	Measurement of Memory Access Capabilities
Hard Disk Drive	Prime-Probe, Shared Disk Drive Data Access	Time Compared Against Threshold

Measurement methods for different hardware units

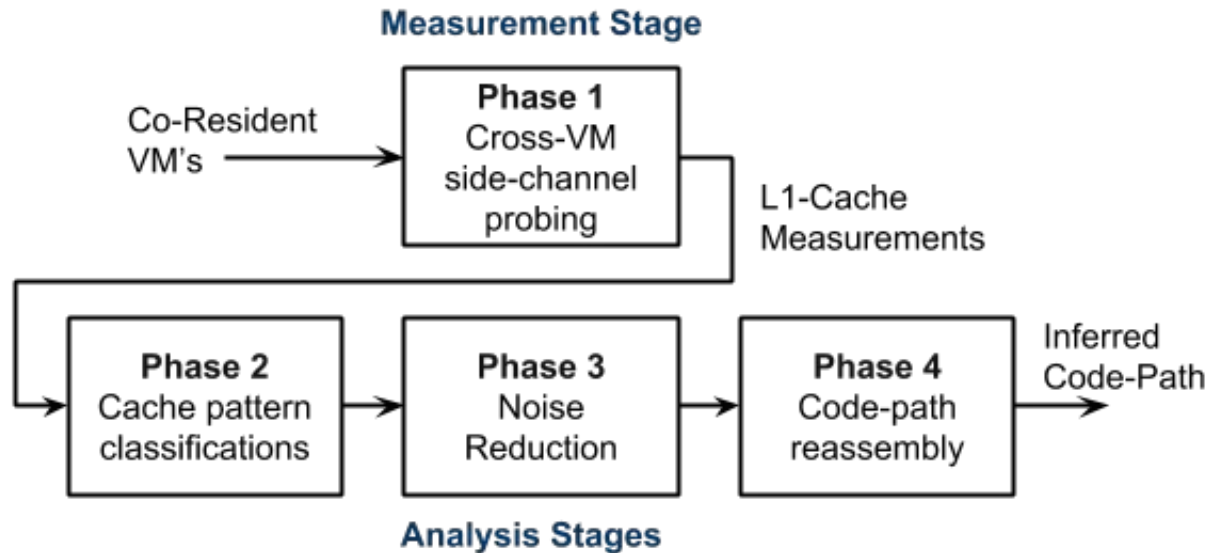
Medium	Transmission	Reception	Constraints
L1 Cache	Prime Probe	Timing	Need to Share Processor Space
L2 Cache	Prime Probe/ Preemption	Timing	Caches Missing Causes Noise
Main Memory	SMT Paging	Measure Address Space	Peripheral Threads Create Noise
Memory Bus	Lock & Unlock Memory Bus	Measure Access	Halts all Processes Requiring the Bus
CPU Functional Units	Resource Eviction & Usage	Timing	mo' Threads, mo' Problems
Hard drive	Hard Disc Contention - Access Files Frantically	Timing	Dependent on multiple readings of files

Overview

- Side channel attacks
- Side channel mediums: Hardware
- Vulnerabilities in the cloud
- **Demo**
- Review of discovery primitives
- Defenses
- The future

Example: Probing the Cache

Applied to a L1 cache side channel



Demo 1: Side channel setup

Medium: Shared L3 Cache Tier

Vulnerability: Timing attacks

- Cache misses, stored value farther away in memory

Model: Scenario 2 Record Measurements

- Application: Crypto key theft
- Receiver only

General setup: Cross processes.

Demo 1: Flush+Reload Attack [1]

Receiver: 'Attacking' Process

- forcing victim code out of the L3 Cache
- measuring time it takes to access it

Transmitter: Victim process

- performing RSA encryption
- uses target code between the flush and reload of adversary

Requires:

- Timing
- Knowledge of target code

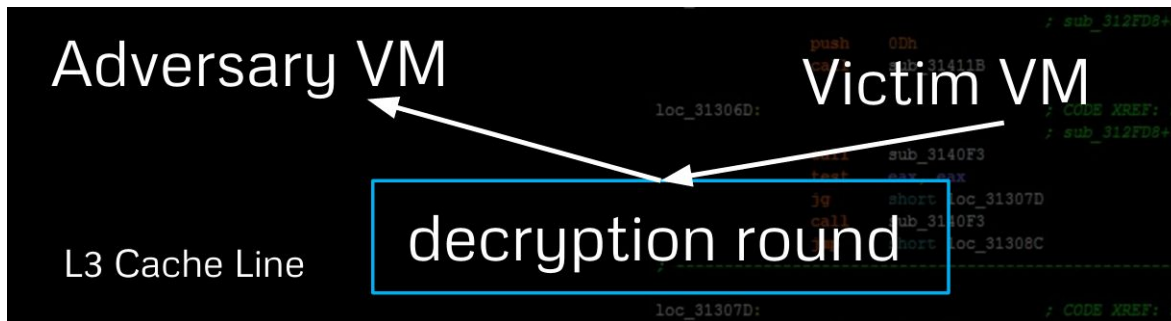
Demo 1: Measuring the L3 Cache Tier

Sources

- <https://defuse.ca/flush-reload-side-channel.htm>
- <https://github.com/DanGe42/flush-reload>

Demo 1: Results

- Successfully leaked the private key from the GnuPG
- Leaked 96.7% bits of the secret key



Demo 2: Side channel setup

Medium: CPU Pipeline

Vulnerability: Erroneous Values

- SMT optimizations, different values possible

Model: Scenario 3

- Application: Communicate a signal.
- Sender and receiver

General setup: Cross VM.

Out-of-Order-Execution

Demo 2: Out-of-Order Execution Attack

Receiver: Measuring Process

- Hardware medium must be measured dynamically unlike the cache.
- Instruction order, results from instruction sets

Transmitter: Sending process

- Force the pipeline state to optimize a certain way.....
- Or not optimize, memory fences

Requires:

- Timing
- Pre-arranged encoding

Demo 2: Transmitting out-of-order-executions

Force Deterministic Memory Reordering:

- Compile-time vs Runtime Reordering

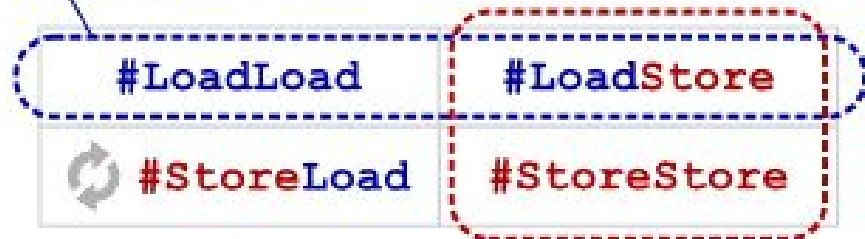
Runtime:

- Usually strong memory model: x86/64 (mostly sequentially consistent)
- Weaker models (data dependency re-ordering): arm, powerpc

Barriers:

- 4 types of run time reordering barriers

acquire semantics



release semantics

Demo 2: Transmitting out-of-order-executions

Force Out of Order Execution: Memory fences

Mfence:

- x86 instruction full memory barrier
- prevents memory reordering of any kind
- order of 100 cycles per operation

```
... mov dword ptr [_spin1], 0
```

```
... mfence
```

```
... mov dword ptr [_spin2], 0
```

```
... mfence
```

Lock-free programming on SMT multiprocessors

Load r1, [mem+arg_0]
mem



Demo 2: Receiving out-of-order-executions

8.2.3.4 Loads May Be Reordered with Earlier Stores to Different Locations

The Intel-64 memory-ordering model allows a load to be reordered with an earlier store to a different location. However, loads are not reordered with stores to the same location.

The fact that a load may be reordered with an earlier store to a different location is illustrated by the following example:

Example 8-3. Loads May be Reordered with Older Stores

Processor 0	Processor 1
<code>mov [_x], 1</code> <code>mov r1, [_y]</code>	<code>mov [_y], 1</code> <code>mov r2, [_x]</code>
Initially $x = y = 0$ $r1 = 0$ and $r2 = 0$ is allowed	

THREAD 1

THREAD 2

Synched

store [X], 1

load r1, [Y]

store [Y], 1

load r2, [X]

=>

r1 = r2 = 1

Asynched

store [X], 1

load r1, [Y]

store [Y], 1

load r2, [X]

=>

r1 = 0 r2 = 1

Out of Order Execution

load r1, [Y]

store [X], 1

load r2, [X]

store [Y], 1

=>

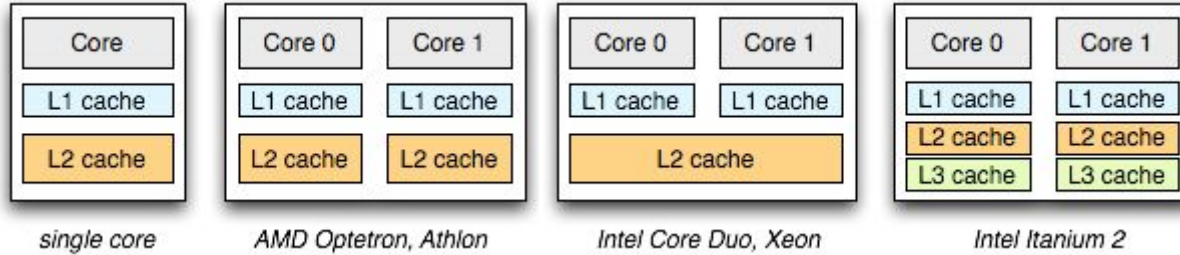
r1 = r2 = 0

```

push    eax
push    edi
push    edi
mov     [ebp+arg_0], eax
call   sub_31486A
test   eax, eax
jz     short loc_31306D
push   esi
mov     [ebp+arg_0], esi
call   sub_314623
test   eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], esi
jz     short loc_31306F
loc_31306D:
; CODE XREF: sub_312F
; sub_312FD8+49
call   sub_31411B
loc_31306D:
; CODE XREF: sub_312F
; sub_312FD8+49
call   sub_3140F3
call   sub_3140F3
loc_31307D:
; CODE XREF: sub_312F
; sub_312FD8+49
call   sub_3140F3

```


Demo 2: Hardware Architectures



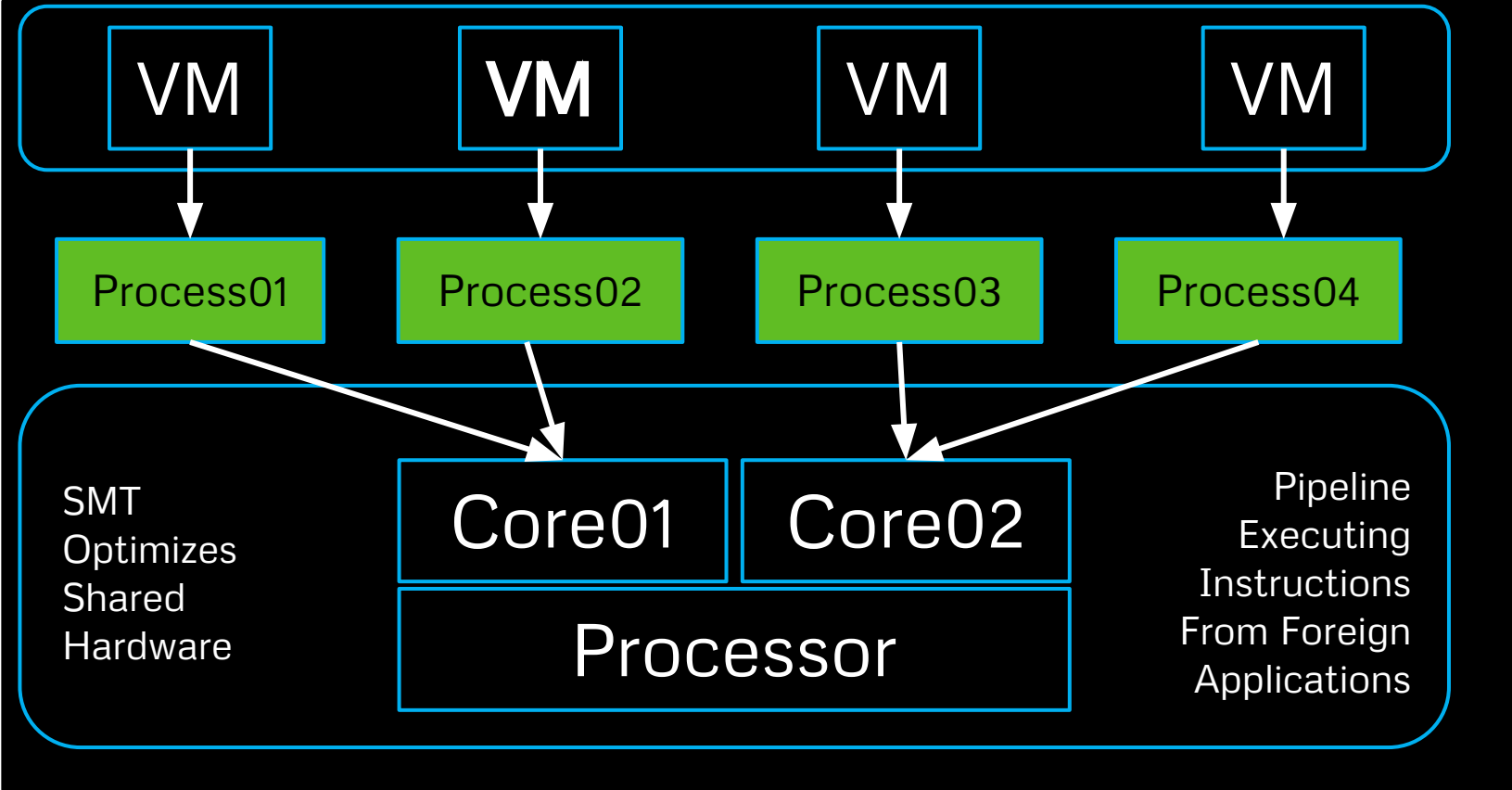
Lab Setup:

- Intel's Core Duo, Xeon Architecture
- Each processor has two cores
- The Xen hypervisor schedules between all processors on a server
- Each core then allocates processes on its pipeline

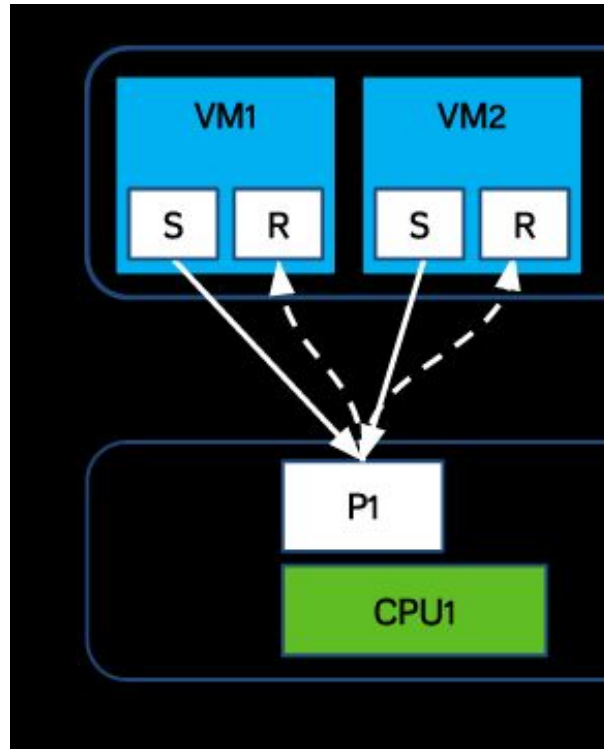
Notes:

- Multiple processes run on a single pipeline (SMT)
- Relaxed memory model

Demo 2: VM Processor Contention



Demo 2: VM Processor Contention



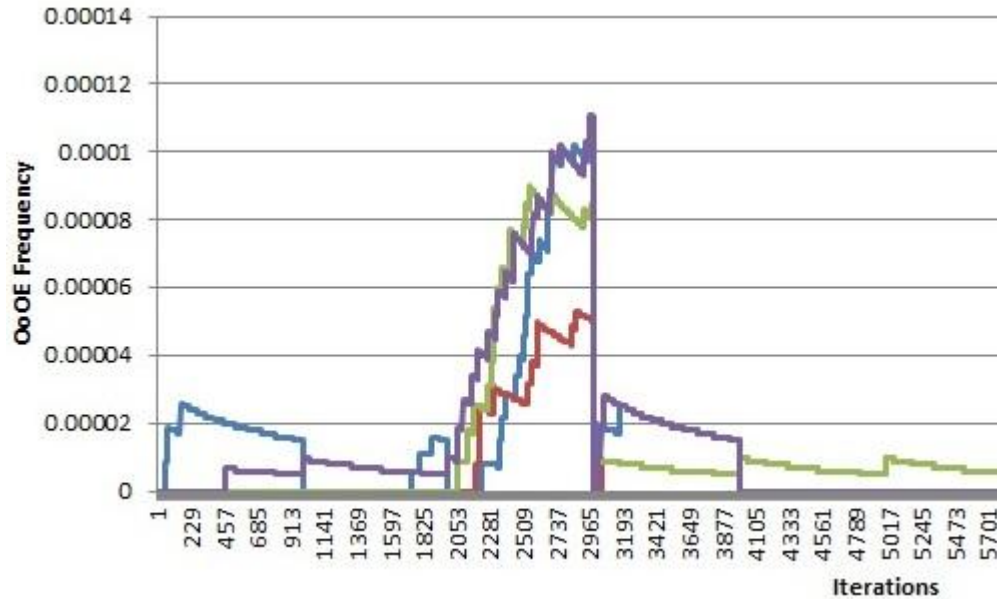
Demo 2: Measuring the Pipeline

Sources & Paper

- <http://www.sophia.re/SC>

Demo 2: Results

Process changes signature of queried hardware unit over time



Demo 2: Results

Benefits:

- Harder for a intelligent hypervisor to detect, quiet
- Eavesdropping sufficiently mutilates channel
- System artifacts sent and queried dynamically
- Not affected by cache misses
- Channel amplified with system noise

Overview

- Side channel attacks
- Side channel mediums: Hardware
- Vulnerabilities in the cloud
- Demo
- **Defenses**
- The future

Defensive mechanisms: Hardware

Protected Resource Ownership:

- Isolating VM's
- Turn off hyperthreading
- Blacklisting resources for concurrent threads
- Downside: removes optimizations or benefits of the cloud



Defensive mechanisms: Hypervisor

Anomaly detection:

- Specification
- Pattern recognition
- Records average OoOE patterns
- Predicts what to expect

Defensive mechanisms: Software

Control Flow Changes:

- Hardening software with *Noise*
- Force specific execution patterns (i.e. constant time loops, ...)
- Avoid using certain resources
- Downside: compiler, hardware optimizations lost



Overview

- Side channel attacks
- Side channel mediums: Hardware
- Vulnerabilities in the cloud
- Demo
- Defenses
- **The future**

Our Future in the Cloud

Side Channel Potential:

- More resource sharing
- More dynamic optimizations
- Virtualization more popular
- Malware

Things to Consider:

- Cloud Side Channels apply to anything with virtualization (i.e. VM's)
- Hypervisors are easy targets: Vulnerable host

i.e. “Xenpwn”, paravirtualized driver attack: INFILTRATECon 2016

Conclusion

- What is a side channel
- Primitives for hardware based side channels
 - Co-location
 - Medium
 - Transmit the artifact
 - Record/ observe the artifact
- Different mediums in the cloud
- Variety of possible attacks
- Cross VM and process

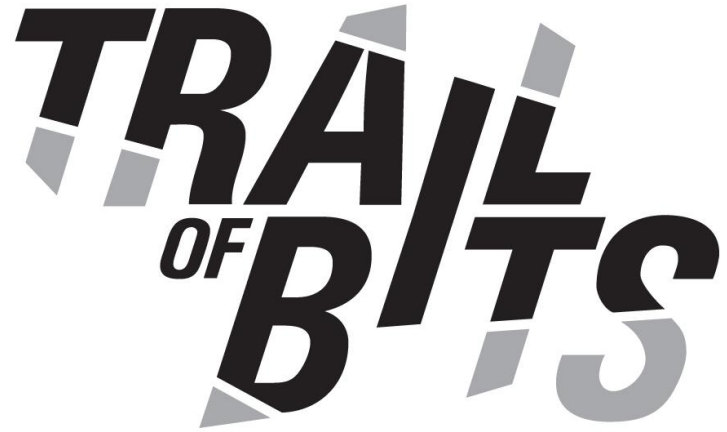


Acknowledgements

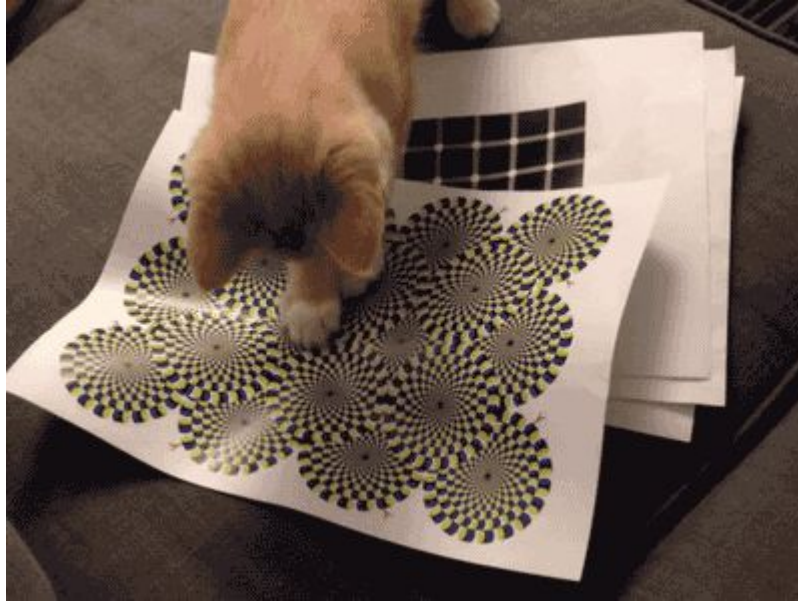
Jeremy Blackthorne

RPISEC

Trail of Bits

The logo for 'TRAIL OF BITS' is displayed in a bold, black, sans-serif font. The word 'TRAIL' is positioned above 'OF', which is smaller and centered between 'TRAIL' and 'BITS'. The word 'BITS' is the largest and is positioned below 'OF'. The letters in 'TRAIL' and 'BITS' have a slight 3D effect, with grey shadows on their right and bottom sides, giving them a sense of depth and movement.The logo for 'RPISEC' is shown in a bold, blue, sans-serif font with a black outline. It is set against a background of faint, light blue hexagonal characters arranged in a grid. The characters are a mix of letters and numbers, including '64', '65', '66', '67', '68', '69', '70', '71', '72', '73', '74', '75', '76', '77', '78', '79', '80', '81', '82', '83', '84', '85', '86', '87', '88', '89', '90', '91', '92', '93', '94', '95', '96', '97', '98', '99', '00', '01', '02', '03', '04', '05', '06', '07', '08', '09', '0a', '0b', '0c', '0d', '0e', '0f', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '1a', '1b', '1c', '1d', '1e', '1f', '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '2a', '2b', '2c', '2d', '2e', '2f', '30', '31', '32', '33', '34', '35', '36', '37', '38', '39', '3a', '3b', '3c', '3d', '3e', '3f', '40', '41', '42', '43', '44', '45', '46', '47', '48', '49', '4a', '4b', '4c', '4d', '4e', '4f', '50', '51', '52', '53', '54', '55', '56', '57', '58', '59', '5a', '5b', '5c', '5d', '5e', '5f', '60', '61', '62', '63', '64', '65', '66', '67', '68', '69', '6a', '6b', '6c', '6d', '6e', '6f', '70', '71', '72', '73', '74', '75', '76', '77', '78', '79', '7a', '7b', '7c', '7d', '7e', '7f', '80', '81', '82', '83', '84', '85', '86', '87', '88', '89', '8a', '8b', '8c', '8d', '8e', '8f', '90', '91', '92', '93', '94', '95', '96', '97', '98', '99', '9a', '9b', '9c', '9d', '9e', '9f', 'a0', 'a1', 'a2', 'a3', 'a4', 'a5', 'a6', 'a7', 'a8', 'a9', 'aa', 'ab', 'ac', 'ad', 'ae', 'af', 'b0', 'b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8', 'b9', 'ba', 'bb', 'bc', 'bd', 'be', 'bf', 'c0', 'c1', 'c2', 'c3', 'c4', 'c5', 'c6', 'c7', 'c8', 'c9', 'ca', 'cb', 'cc', 'cd', 'ce', 'cf', 'd0', 'd1', 'd2', 'd3', 'd4', 'd5', 'd6', 'd7', 'd8', 'd9', 'da', 'db', 'dc', 'dd', 'de', 'df', 'e0', 'e1', 'e2', 'e3', 'e4', 'e5', 'e6', 'e7', 'e8', 'e9', 'ea', 'eb', 'ec', 'ed', 'ee', 'ef', 'f0', 'f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'fa', 'fb', 'fc', 'fd', 'fe', 'ff'.

Any Questions?



IRC: quend

Email: sophia@trailofbits.com

Website: <http://sophia.re/SC>

References

- <https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-yarom.pdf>
- <http://bartoszmilewski.com/2008/11/05/who-ordered-memory-fences-on-an-x86/>
- <http://preshing.com/20120913/acquire-and-release-semantics/>
- http://www.intel.com/Assets/en_US/PDF/manual/253668.pdf
- <http://preshing.com/20120930/weak-vs-strong-memory-models/>
- http://en.wikipedia.org/wiki/Memory_barrier#An_illustrative_example
- <http://preshing.com/20120710/memory-barriers-are-like-source-control-operations/>
- <http://gauss.eecs.uc.edu/Courses/c653/lectures/SideC/intro.pdf>
- <https://www.ernw.de/download/xenpwn.pdf>

Extra Slides

Receiver: Record out of order executions

```
int X,Y,count_OoOE;
....initialize semaphores Sema1 & Sema2...
pthread_t thread1, thread2;
pthread_create(&threadN, NULL, threadNFunc, NULL);

for (int iterations = 1; ; iterations++)
    X,Y = 0;
    sem_post(beginSema1 & beginSema2);
    sem_wait(endSema1 & endSema2);

if (r1 == 0 && r2 == 0)
    count_OoOE ++;
```

Details of Demo 2: Pipeline Side Channel Setup

Scheduler Xen hypervisor: Popular commercial IaaS platforms

Xeon Processors

Shared multi-core/ multi-processor hardware

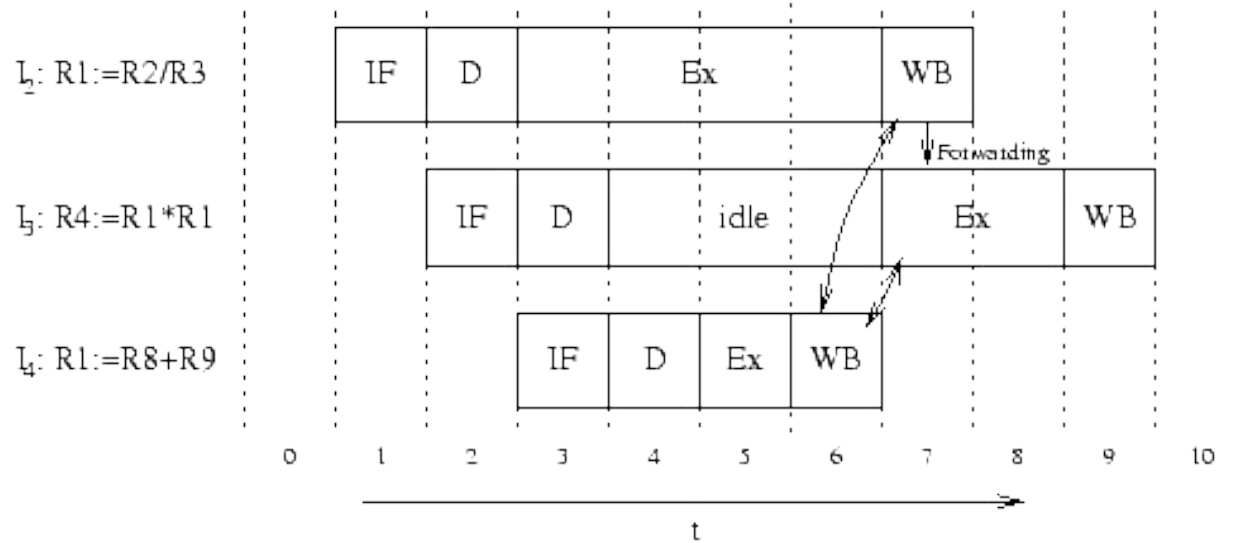
8 logical CPU's/ 4 cores

6 virtual machines (VM's)

Parallel Processing/ Simultaneous Multi-Threading On (SMT)



Details for Demo 2: Pipeline Reordering



Details for Demo 2: Pipeline Reordering

