Securing Systems with Insecure Hardware

Kaveh Razavi

About VUSec

~20 members

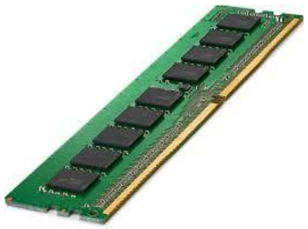
- Software protections
- Binary and malware analysis
- Fuzzing
- Network security
- **Hardware and OS security**

Assuming secure software,

what is still possible?

and what can we do about it?

General-purpose Hardware Attacks (2015-)



Drammer

A government entity in a certain country: “can we please have the Drammer exploit?”

Spectre/MDS

Immunity Inc.
@Immunityinc
Follow

Priv escalation: Leak of /etc/shadow's content using SPECTRE on Fedora 25 amd64. CANVAS Early Updates users will see the update soon and regular CANVAS users will see it on the next CANVAS release. #Spectre #Meltdown

What Is Different?

1) Attacks and their impact are not obvious
2) Problems are often structural
3) Cannot “update” hardware

Defending These New Classes of Attacks

DRAM-based corruptions (Rowhammer)

Hardware-based information leakage

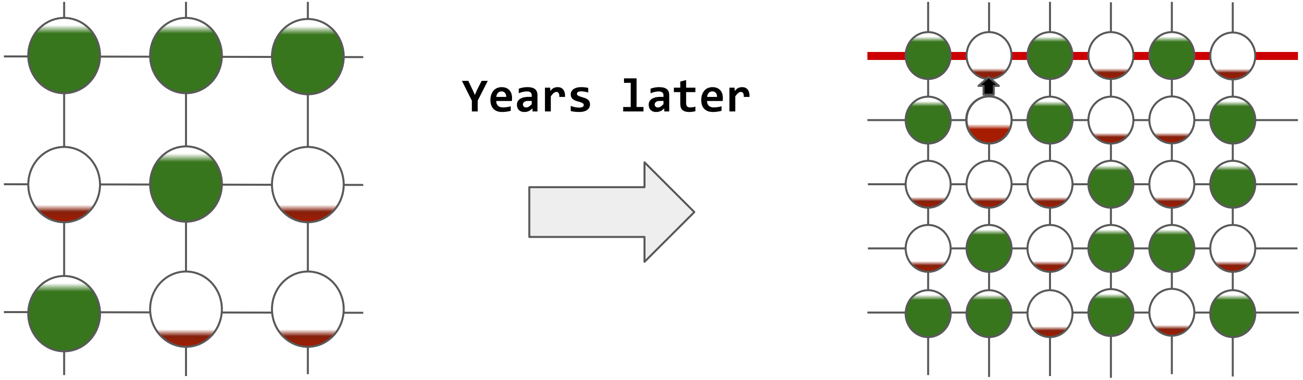
Defending These New Classes of Attacks

DRAM-based corruptions (Rowhammer)

Hardware-based information leakage

The Rowhammer Problem

We have reduced transistor without caring for reliability/security



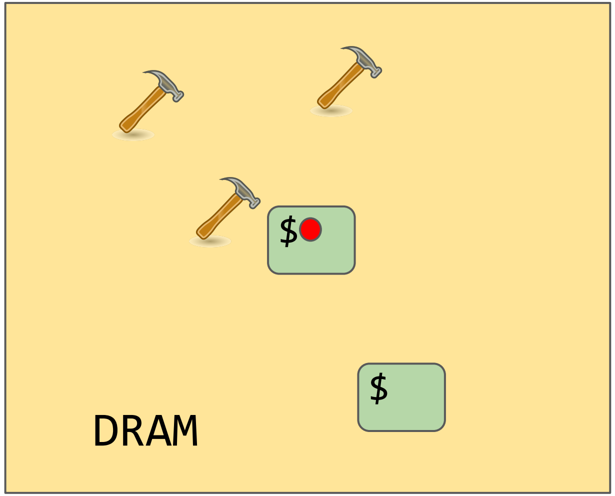
Rowhammer: affects 87% of deployed DDR3 memory, DDR4 as well.

Kim et al., “Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors,” ISCA’14

Exploiting These Flips

Random, previously unknown locations, single flips.

1) Templating
2) Massaging
3) Exploitation



Razavi et al., “Flip Feng Shui: Hammering a Needle into the Software Stack,” SEC’16

Compromising Cloud Virtual Machines

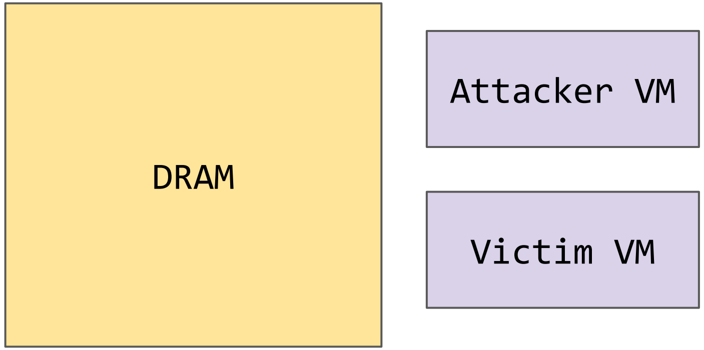
1) Templating

   Attacker’s own memory

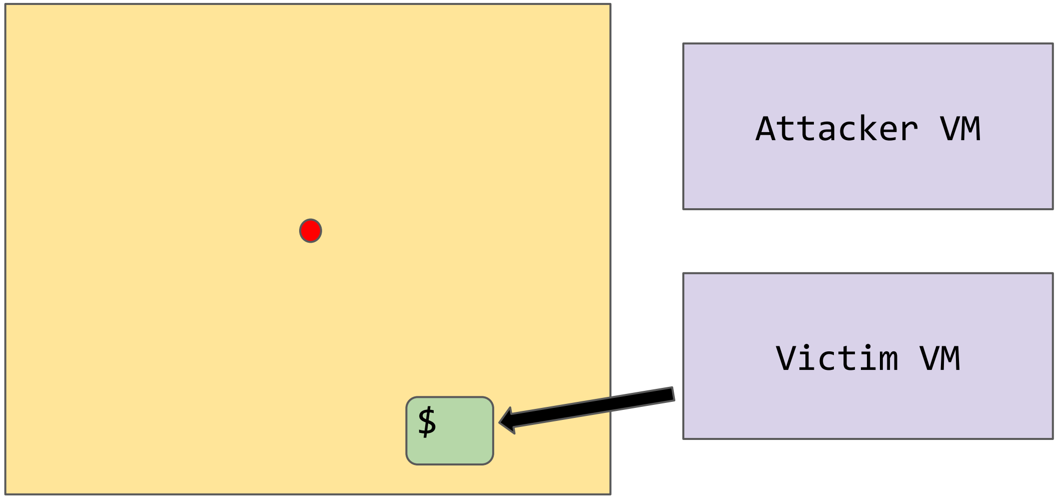
2) Massaging

   Memory deduplication

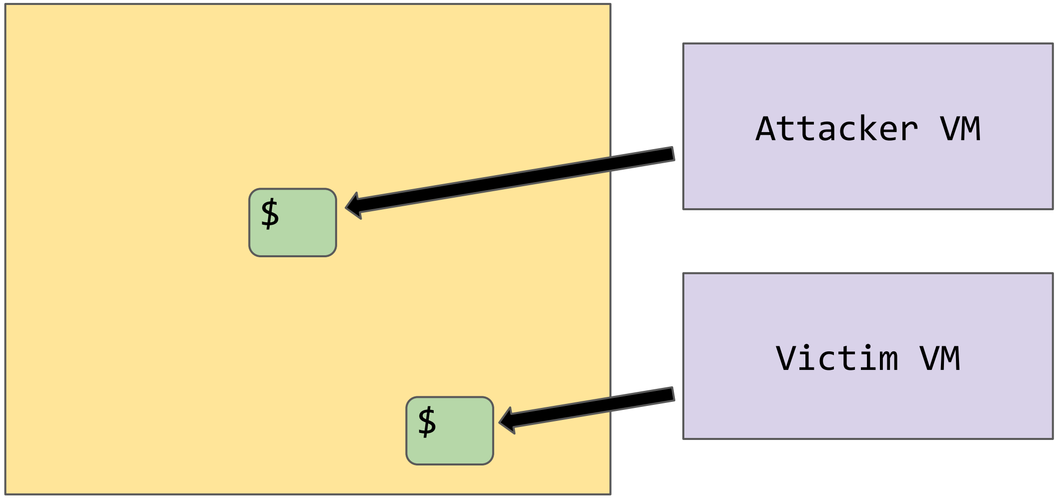
3) Exploitation



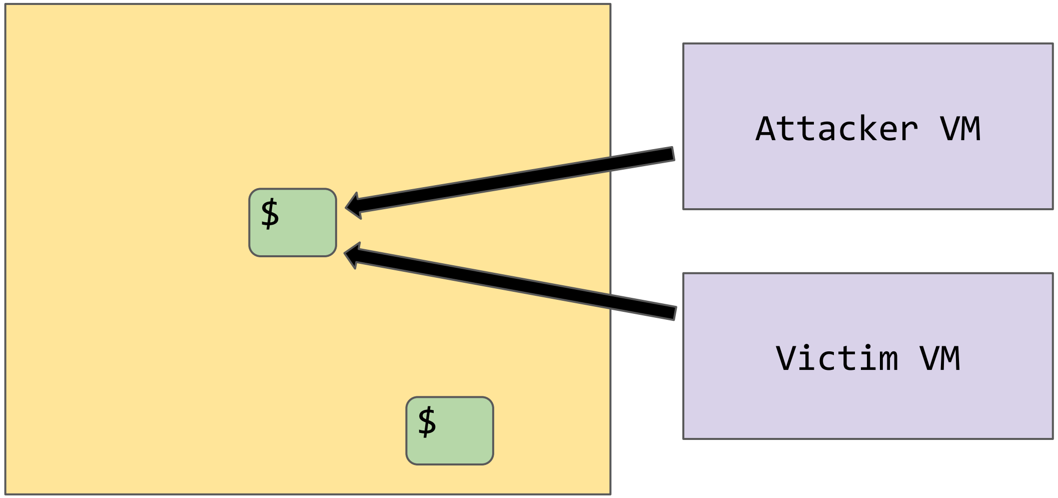
Memory Deduplication



Memory Deduplication



Memory Deduplication



Compromising Cloud Virtual Machines

1) Templating

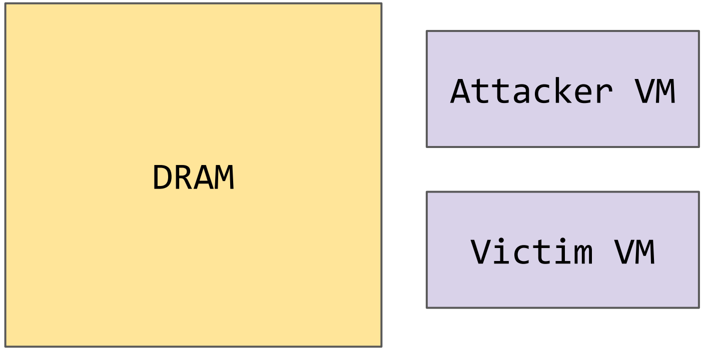
Attacker's own memory

2) Massaging

Memory deduplication

3) Exploitation

Corrupt RSA public keys (OpenSSH)



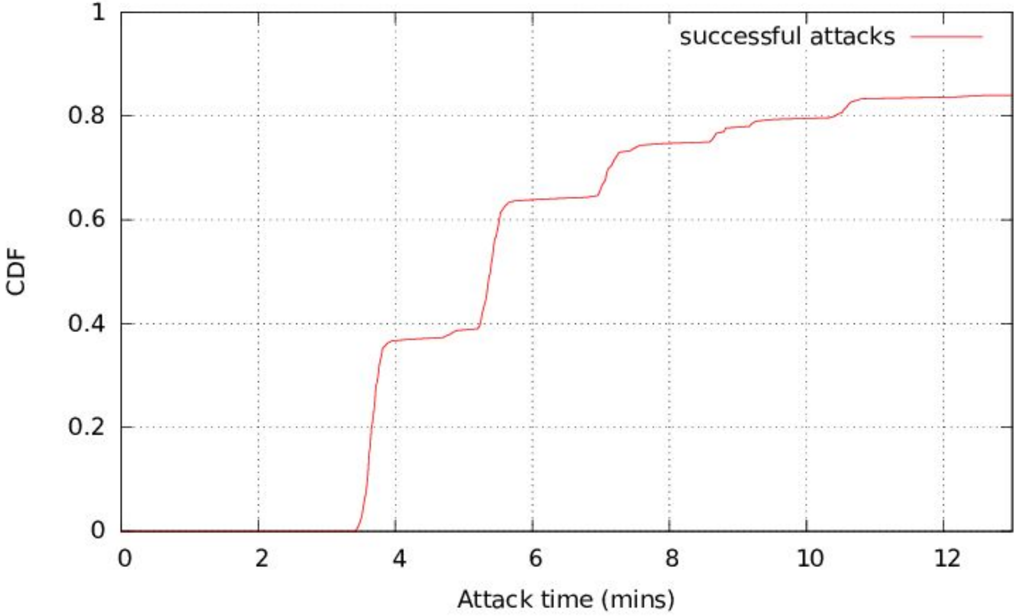
Factorizing Corrupted RSA Public Keys

$$n = p \times q \rightarrow \text{PK (public key)}$$

$$\text{PK} \xrightarrow{\text{FFS}} \text{PK'}$$

$$\text{PK'} \rightarrow n' = p' \times q' \times z' \times \ldots$$

Attack’s Success Rate



Flip Feng Shui on Mobile Devices (ARM)

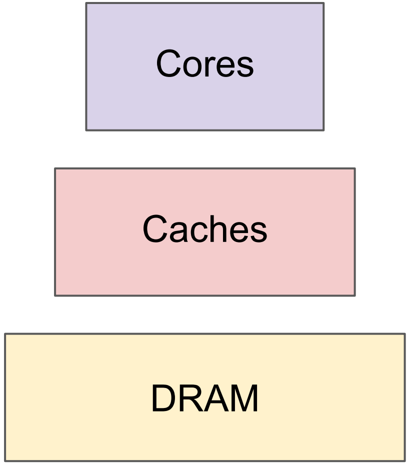
1) Templating

   Not possible to hammer → ION (DMA) memory

2) Massaging → buddy allocation

   No dedup

3) Exploitation



Results - Drammer on 27 phones

Device	#flips	1st exploitable flip after
LG Nexus 5[1]	1058	116s
LG Nexus 5[4]	0	-
LG Nexus 5[5]	747,013	1s
LG Nexus 4	1,328	7s
OnePlus One	3,981	942s
Motorola Moto G (2013)	429	441s
LG G4 (ARMv8 – 64-bit)	117,496	5s

22 seconds to root on 18 out of 27 tested phones.

Impact

- Similarly (GLitch S&P’18)
- Response:
    - Goog
    - Micro dup off

- Major media attention
- Two best paper and two pwnie awards

Defending These New Classes of Attacks

DRAM-based corruptions (Rowhammer)

Hardware-based information leakage

Proposed Defenses

Disabling features:

- Deduplication (massaging)
- ION contiguous heap (templating)

Expensive and not secure

Drammer (dedup), GuardION (ION)

Proposed Defenses

Disabling features:

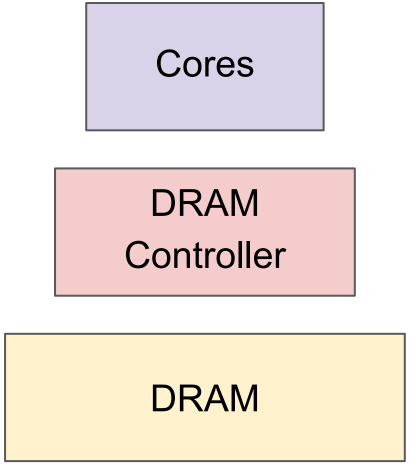
- Deduplication
- ION heaps

Hardware defenses:

- ECC (templating)
- PARA/TRR (templating)

Error-correction Codes (SECDED)

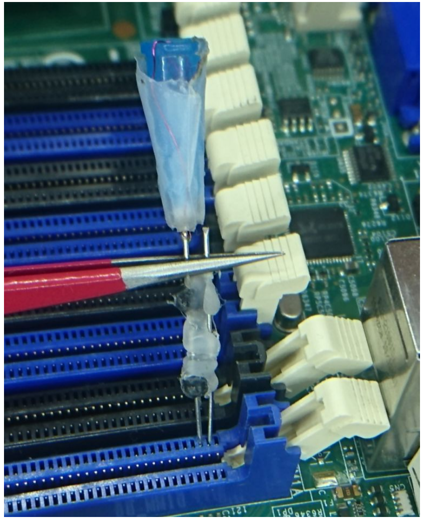
- Original paper demonstrated SECDED not to be enough
- … but exploitation turned out to be difficult
  - ECC implementation is closed (guarantees unknown)
  - 1 bit flips not visible, 2 bit flips crash the system



ECC DRAM as a practical secure defense.

Recovering ECC Functions

- Observing signals are not easy at 1Ghz+
  - Need custom interposer
  - Expensive logic analyzer
- Fault injection with syringe needles!
- Short-circuit data lines with Vss
  - High-to-low voltage flips
- With some math error reports allows for ECC recovery



Cojocar et al., “Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks,” S&P’19

Results

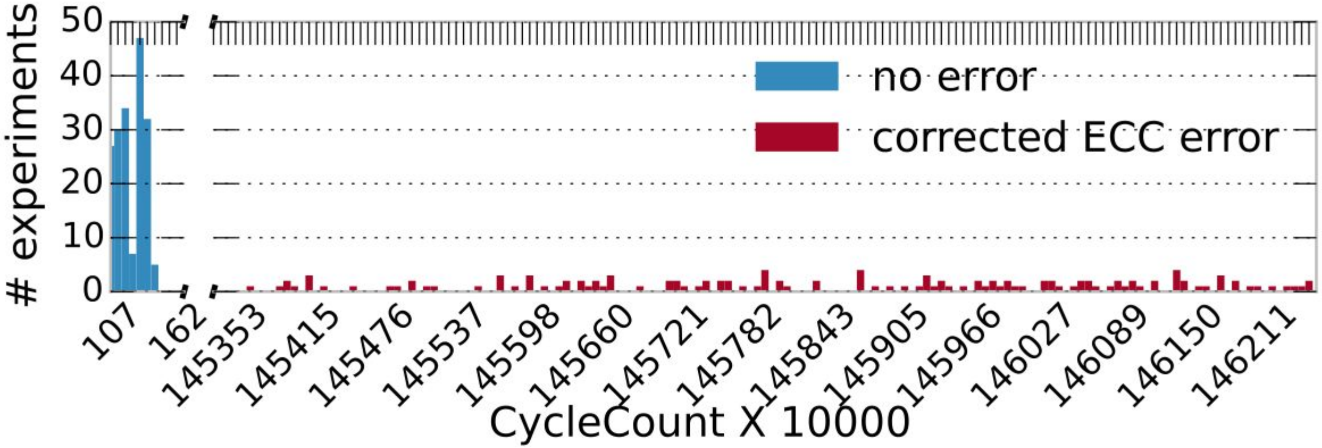
TABLE V: Error patterns that can circumvent ECC.

ID	Pattern	Config.	# flips	Flips location
AMD-1	[$\mathcal{P}_1$]	Ideal	3-BF-16	3 symbols, 1 in control bits
AMD-1	[$\mathcal{P}_2$]	Ideal	4-BF-16	Min. 2 symbols
Intel-1	[$\mathcal{P}_3$]	Ideal	4-BF-8	Min. 2 symbols
Intel-1	[$\mathcal{P}_4$]	Default	2-BF-8	Min. 2 symbols

TABLE VI: Percentages of rows with corruptions in an ECC DIMM.

[$\mathcal{P}_1$]	[$\mathcal{P}_2$]	[$\mathcal{P}_3$]	[$\mathcal{P}_4$]
0.12%	0.12%	0.06%	0.60%

Avoiding Crashes



Detect single flips and merge them for silent corruptions.

Distinguished Practical Paper Award at S&P'19

Proposed Defenses

Disabling features:

- Deduplication
- ION heaps

Hardware defenses:

- ECC (templating)
- PARA/TRR (templating)

Not deployed everywhere and some implementations are insecure (current work)

Proposed Defenses

Disabling features:

- Deduplication
- ION heaps

Hardware defenses:

- ECC (templating)
- PARA/TRR (templating)

Proper protection in software with existing hardware interfaces?

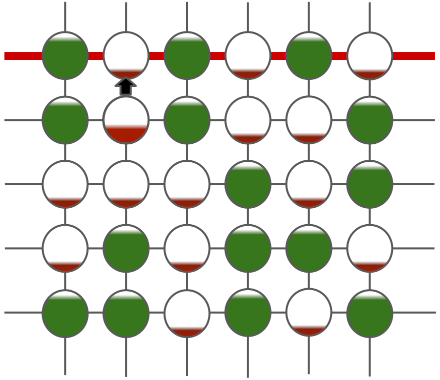
Defending These New Classes of Attacks

DRAM-based corruptions (Rowhammer)

Hardware-based information leakage

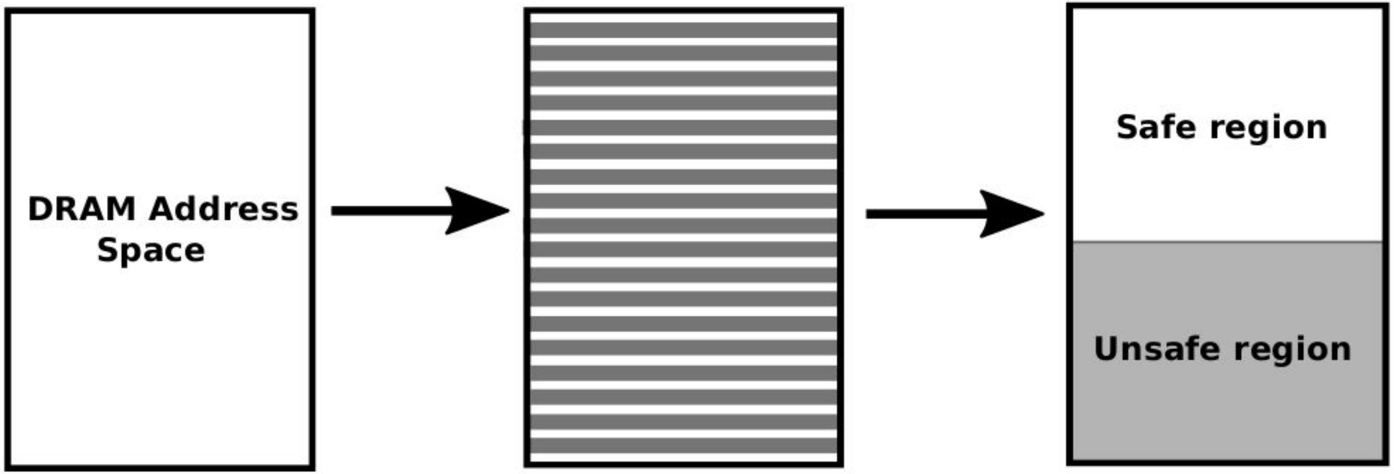
Rowhammer’s Fault Model

Bit flips affect adjacent rows



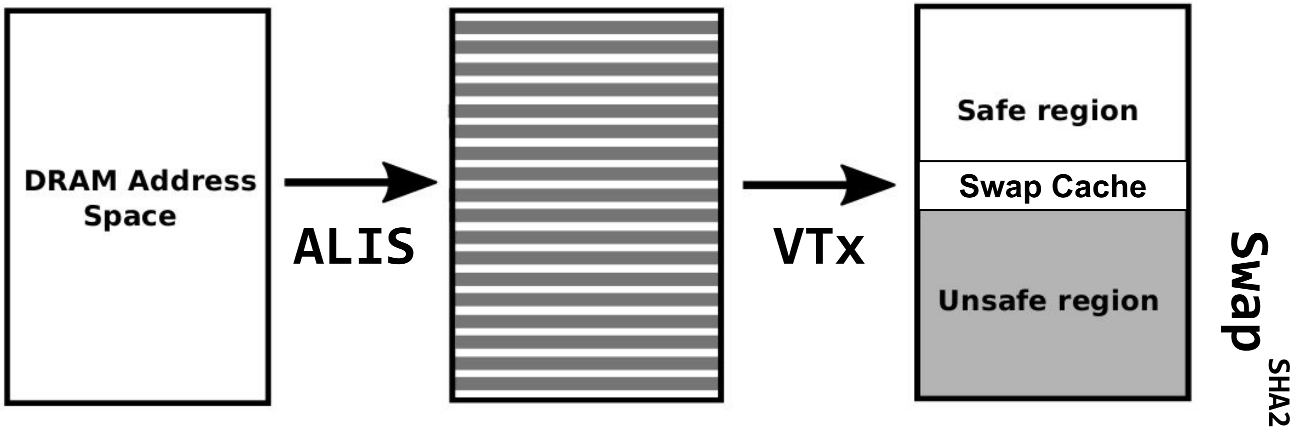
Isolate every memory row from another...

ZebRAM: Even/Odd Rows Isolated from Each Other



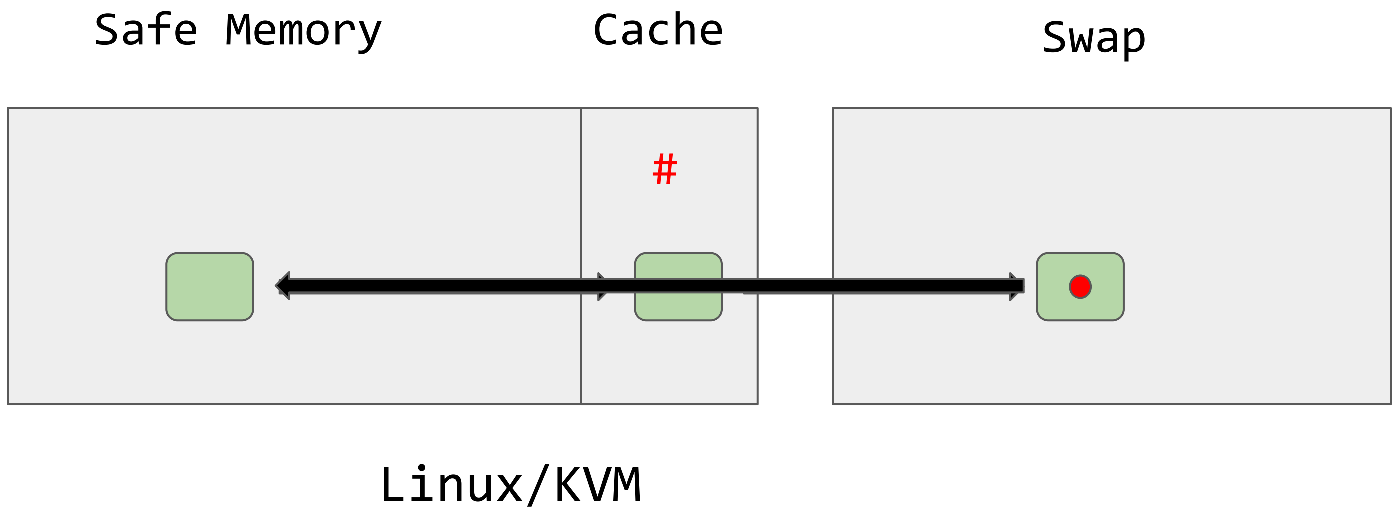
Konoth et al., “ZebRAM: Comprehensive and Compatible Software Protection Against Rowhammer Attacks,” OSDI’18

ZebRAM’s Design



1) How to allocate odd/even rows?
2) How to map odd/even rows to safe/unsafe regions?
3) How to utilize unsafe region?
4) How to protect the safe/unsafe regions?

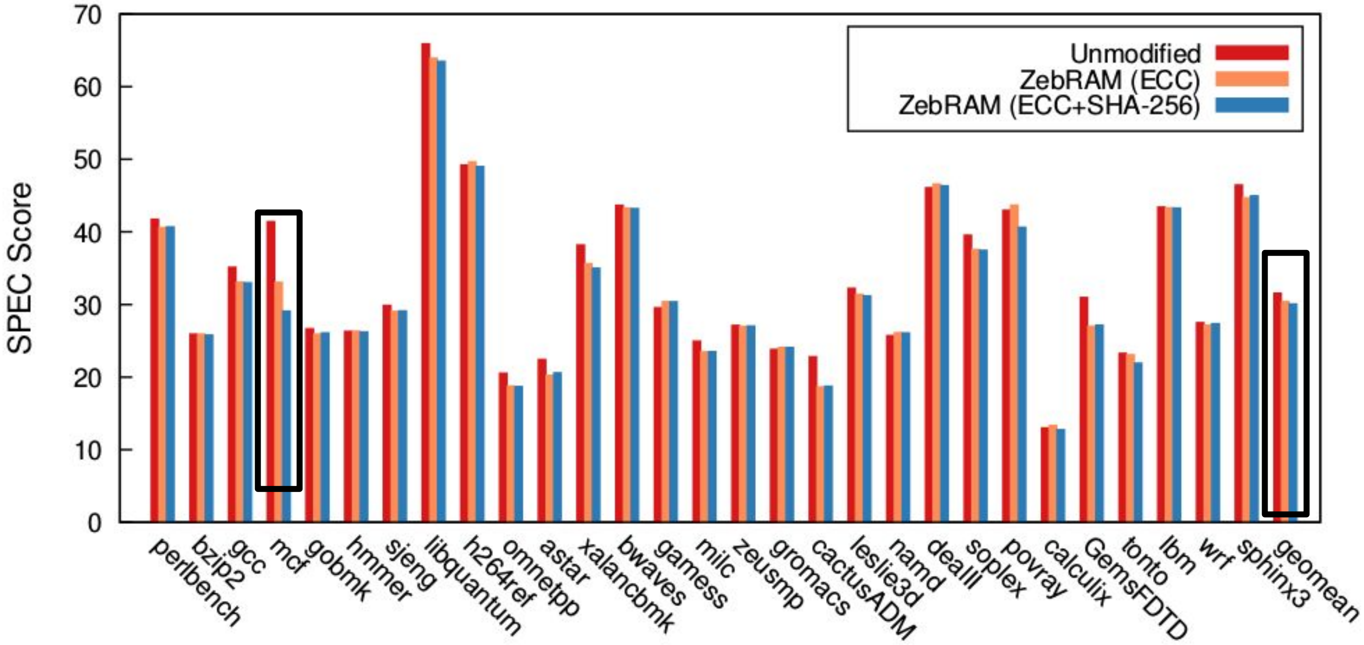
Life of a Page in ZebRAM



Kernel: 1454 LoC

User: 5118 LoC

Evaluation: SPEC 2006



Evaluation: Security

Run no.	Total Number of Flips	Detected by ZebRAM
1	4,702	4,702
2	5,132	5,132
3	2,790	2,790

First comprehensive and compatible Rowhammer protection.

Defending These New Classes of Attacks

Novel Attacks → Defense Assessment → Novel Defenses

DRAM-based corruptions (Rowhammer)

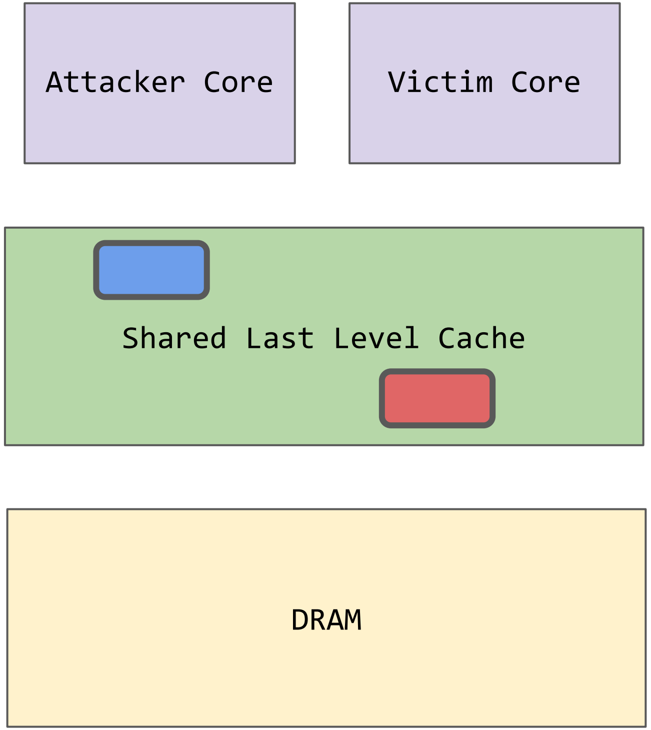
Hardware-based information leakage

Defending These New Classes of Attacks

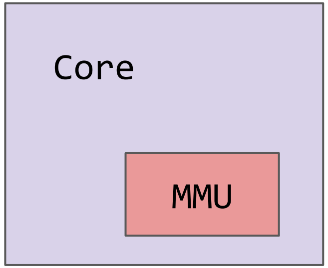
DRAM-based corruptions (Rowhammer)

Hardware-based information leakage

Traditional Cache Attacks



Attacking CPU-internal Components

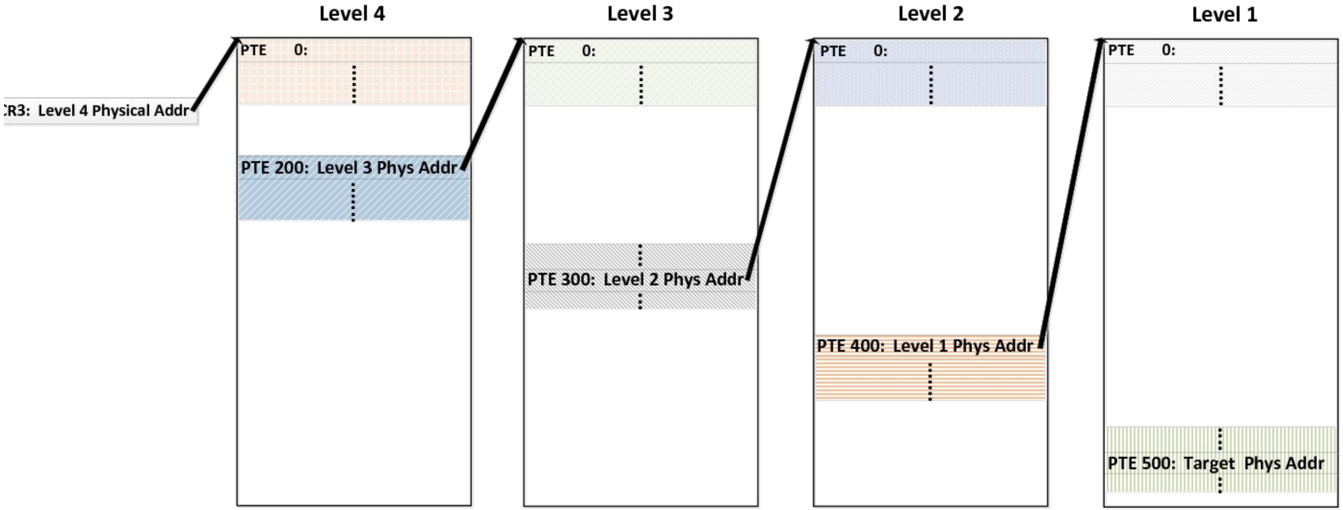


AnC

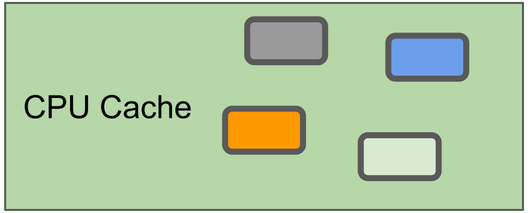
ASLR leak

2017

AnC: MMU Leaves a Trace in the CPU Caches

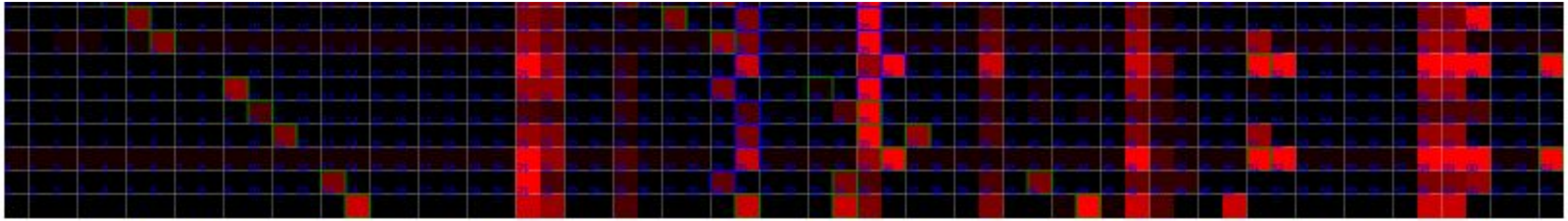


Secret: randomized virtual address



Gras/Razavi et al., "ASLR on the Line: Practical Cache Attacks on the MMU," NDSS'17

AnC from JavaScript



```
24.457 got level 4 - start slot 148, address 0x94000
24.993 got level 3 - start slot 295, address 0x24e94000
24.993 estimated remaining entropy 6 slot solutions: -1,-1,295,148
68.737 got level 4 - start slot 0, address 0x0
69.502 got level 3 - start slot 359, address 0x2ce00000
70.259 got level 2 - start slot 411, address 0x66ece00000
88.041 got level 1 - start slot 238, address 0x7766ece00000
88.041 estimated remaining entropy 0 slot solutions: 238,411,359,0
data: 0x7766ece00000, code slots: -1,-1,295,148, code: 0x7966e4e94000
```

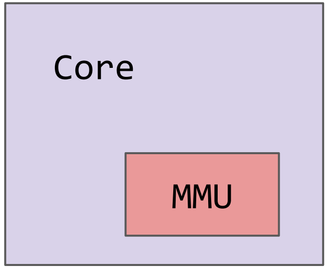
Affected Architectures

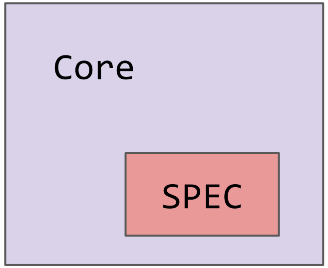
CPU	Year
Intel Core i7-7500U (Kaby Lake) @ 2.70GHz	2016
Intel Core m3-6Y30 (Skylake) @ 0.90GHz	2015
Intel Xeon E3-1240 v5 (Skylake) @ 3.50GHz	2015
Intel Core i7-6700K (Skylake) @ 4.00GHz	2015
Intel Celeron N2840 (Silvermont) @ 2.16GHz	2014
Intel Core i7-4500U (Haswell) @ 1.80GHz	2013
Intel Core i7-3632QM (Ivy Bridge) @ 2.20GHz	2012
Intel Core i7-2620QM (Sandy Bridge) @ 2.00GHz	2011
Intel Core i5 M480 (Westmere) @ 2.67GHz	2010
Intel Core i7 920 (Nehalem) @ 2.67GHz	2008
AMD Ryzen 7 1700 8-Core (Zen) @ 3.3GHz	2017
AMD Ryzen 5 1600X 6-Core (Zen) @ 3.6GHz	2017
AMD FX-8350 8-Core (Piledriver) @ 4.0GHz	2012
AMD FX-8320 8-Core (Piledriver) @ 3.5GHz	2012
AMD FX-8120 8-Core (Bulldozer) @ 3.4GHz	2011
AMD Athlon II 640 X4 (K10) @ 3.0GHz	2010
AMD E-350 (Bobcat) @ 1.6GHz	2010
AMD Phenom 9550 4-Core (K10) @ 2.2GHz	2008
Rockchip RK3399 (ARM Cortex A72) @ 2.0GHz	2017
Rockchip RK3399 (ARM Cortex A53) @ 1.4GHz	2017
Allwinner A64 (ARM Cortex A53) @ 1.2GHz	2016
Samsung Exynos 5800 (ARM Cortex A15) @ 2.1GHz	2014
Nvidia Tegra K1 CD580M-A1 (ARM Cortex A15) @ 2.3GHz	2014
Nvidia Tegra K1 CD570M-A1 (ARM Cortex A15; LPAE) @ 2.1GHz	2014
Samsung Exynos 5800 (ARM Cortex A7) @ 1.3GHz	2014
Samsung Exynos 5250 (ARM Cortex A15) @ 1.7GHz	2012

Impact

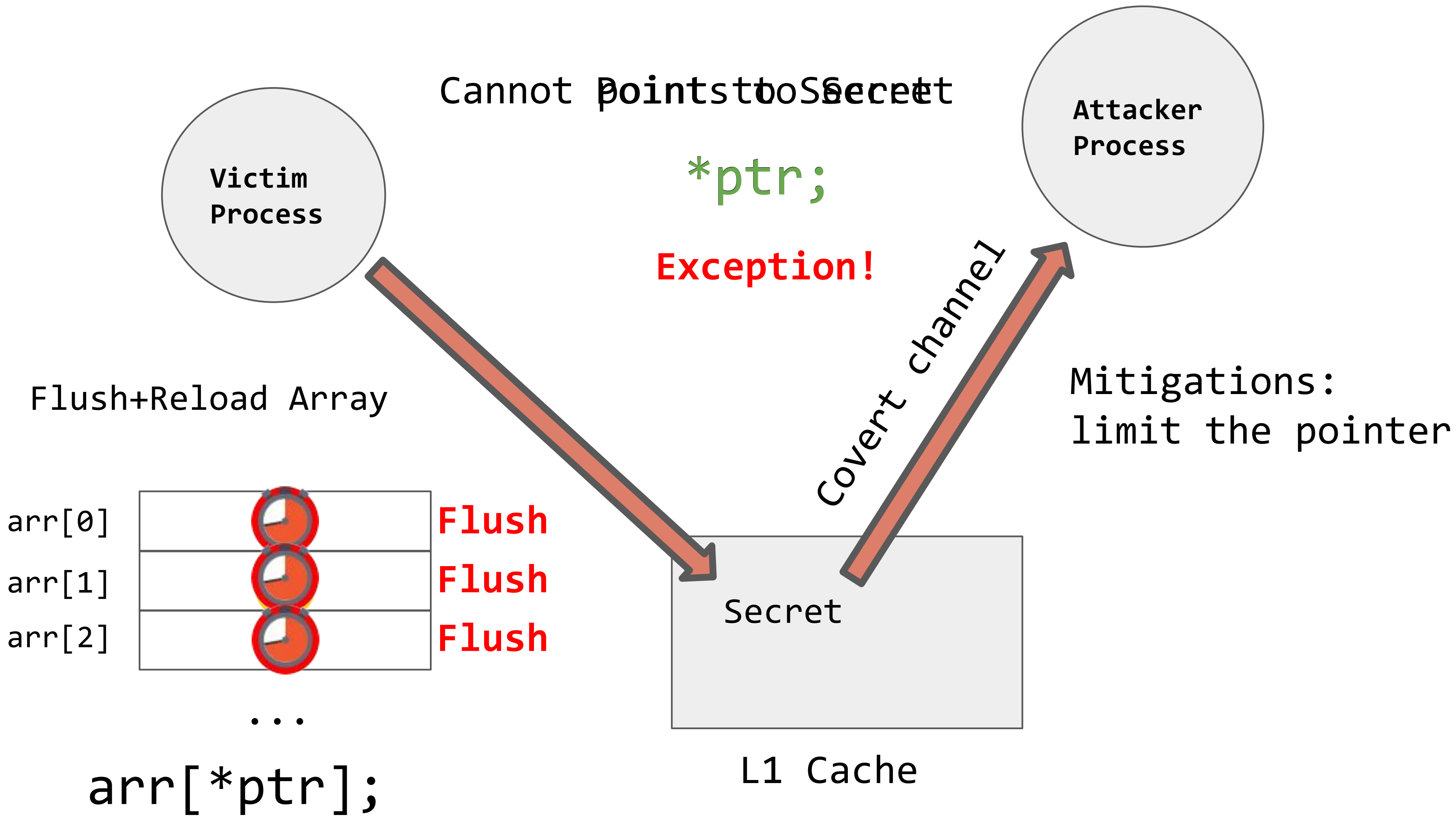
- Response: spot mitigations
    - Apple updated WebKit allocation policies
    - Jitter in the timers
- Best Dutch cyber security research award
- Pwnie for most innovative research

Attacking CPU-internal Components





AnC	Spectre
ASLR leak	Arbitrary leak
2017	2018


Cannot point to Secret
Points to Secret
Victim Process
Attacker Process
*ptr;
Exception!
Covert channel
Flush+Reload Array
Mitigations:
limit the pointer
arr[0]
arr[1]
arr[2]
Flush
Flush
Flush
...
Secret
L1 Cache
arr[*ptr];

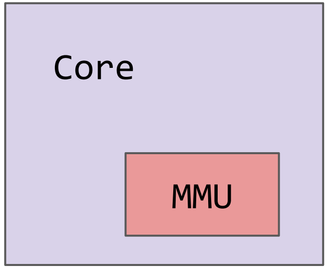
Are these spot mitigations enough?

Defending These New Classes of Attacks

DRAM-based corruptions (Rowhammer)

Hardware-based information leakage

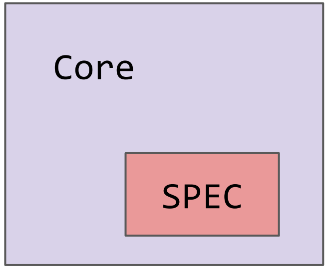
Attacking CPU-internal Components



AnC

ASLR leak

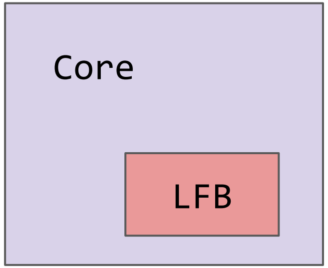
2017



Spectre

Arbitrary leak

2018



RIDL

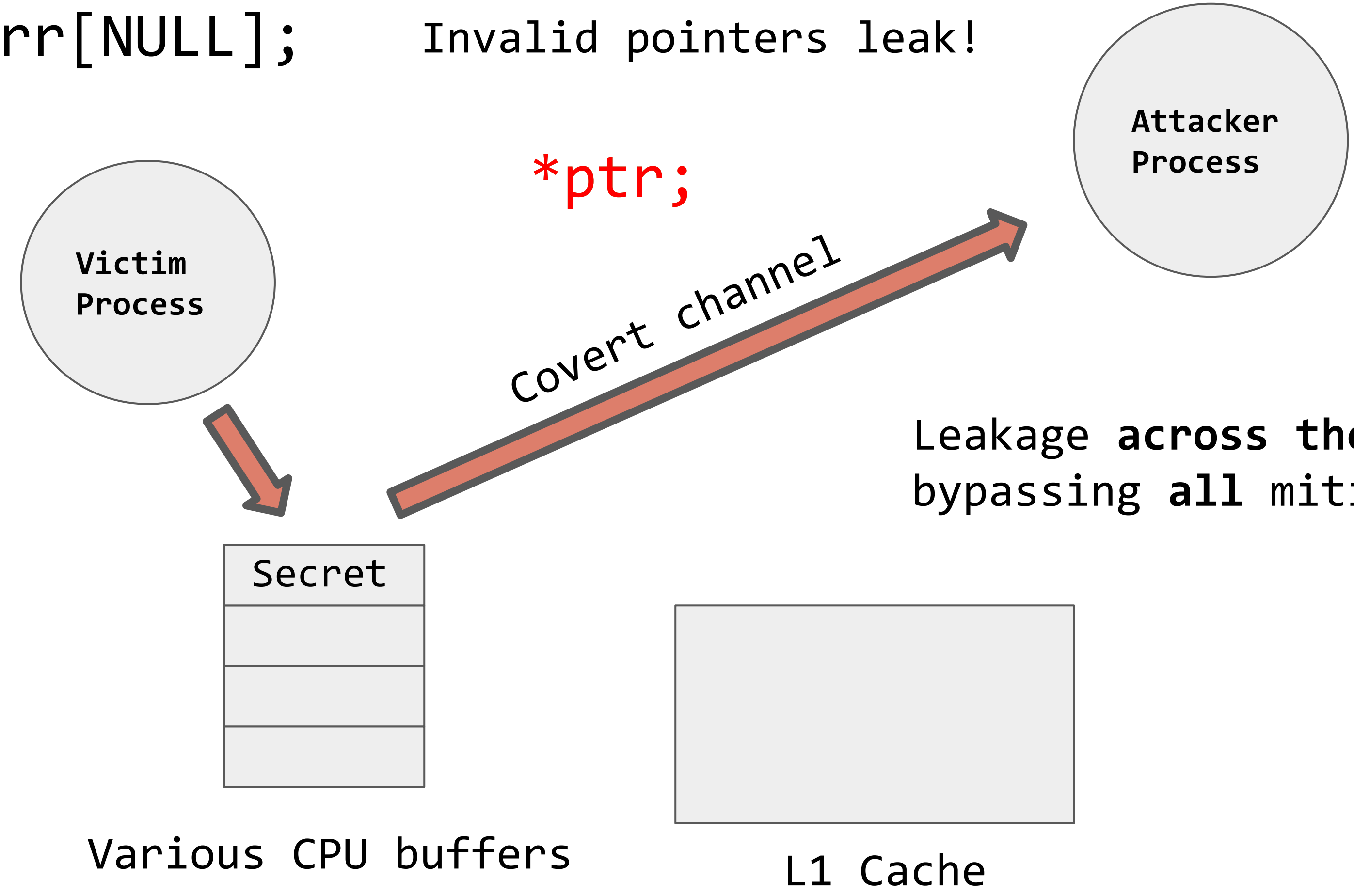
Arbitrary leak

2019

Van Schaik et al., “RIDL: Rogue Inflight Data Load,” S&P’19

arr[NULL];

Invalid pointers leak!



Leakage **across the board**,
bypassing **all** mitigations.

Which CPUs Are Vulnerable?

- ✓ Intel Core i9-9900K (Coffee Lake R) - 2018
- ✓ Intel Xeon Silver 4110 (Skylake SP) - 2017
- ✓ Intel Core i7-8700K (Coffee Lake) - 2017
- ✓ Intel Core i7-7800X (Skylake X) - 2017
- ✓ Intel Core i7-7700K (Kaby Lake) - 2017
- ✓ Intel Core i7-6700K (Skylake) - 2015
- ✓ Intel Core i7-5775C (Broadwel) - 2015
- ✓ Intel Core i7-4790 (Haswell) - 2014
- ✓ Intel Core i7-3770K (Ivy Bridge) - 2012
- ✓ Intel Core i7-2600 (Sandy Bridge) - 2011
- ✓ Intel Core i3-550 (Westmere) - 2010
- ✓ Intel Core i7-920 (Nehalem) - 2008
- ✗ AMD Ryzen 5 2500U (Raven Ridge) - 2018
- ✗ AMD Ryzen 7 2600X (Pinnacle Ridge) - 2018
- ✗ AMD Ryzen 7 1600X (Summit Ridge) - 2017

1 Year of CVD with Intel

$100,000 bounty award

Other Defenses: Partitioning

- Partitioning is imperfect
    - TLBLeed (SEC'18), XLATE (SEC'18)

- New OS primitives allow for secure partitioning (VUsion, SOSP'17)

Conclusion

Hardware is the new software except it is harder to fix

Spot mitigations are costly and ineffective

Principled mitigations in software/hardware

Ben Gras, Victor van der Veen, Erik Bosman, Pietro Frigo, Andrei Tatar, Radhesh Konoth, Stephan van Schaik, Alyssa Milburn, Sebastian Ostersund, Dennis Andriesse, Elias Athanasopoulos, Daniel Gruss, Clementine Maurice, Yanick Fratantonio, Martina Lindorfer, Giovanni Viga, Bart Preneel, Cristiano Giuffrida, Herbert Bos

I am hiring PhD students!

To do exciting hardware security research

Email: kaveh@cs.vu.nl

Twitter: @kavehrazavi

What’s Next?

Throwhammer: Rowhammer Attacks over the Network and Defenses

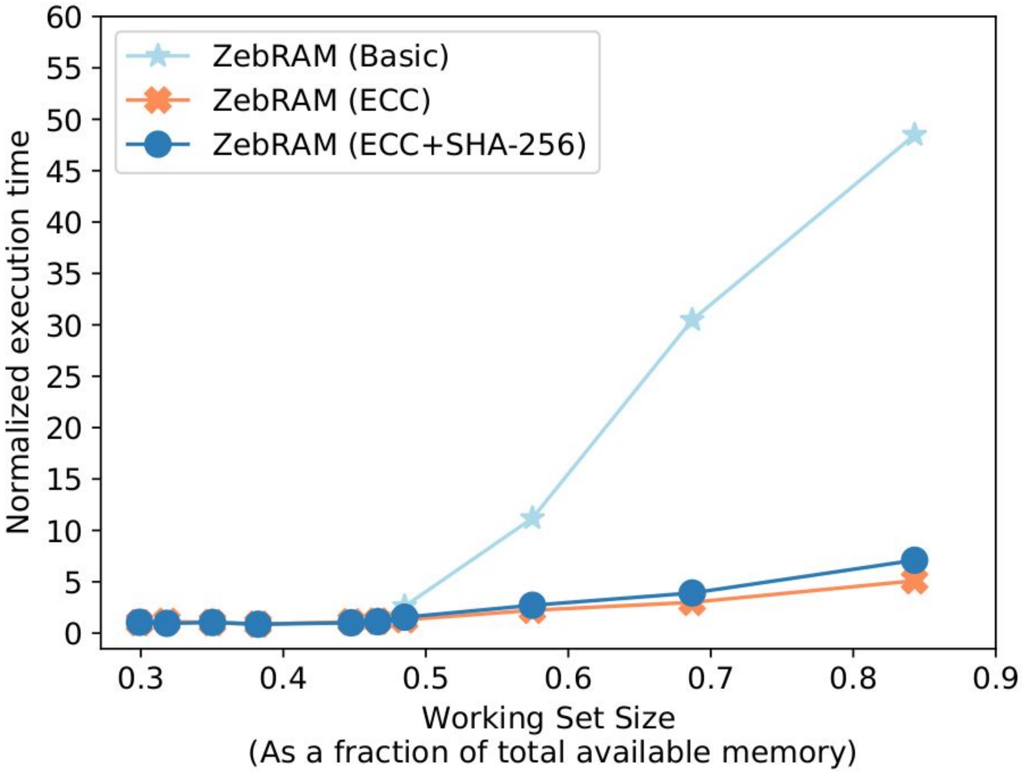
NetCAT: Practical Cache Attacks from the Network

Michael Kurth[*§], Ben Gras[*], Dennis Andriesse[*], Cristiano Giuffrida[*], Herbert Bos[*], and Kaveh Razavi[*]

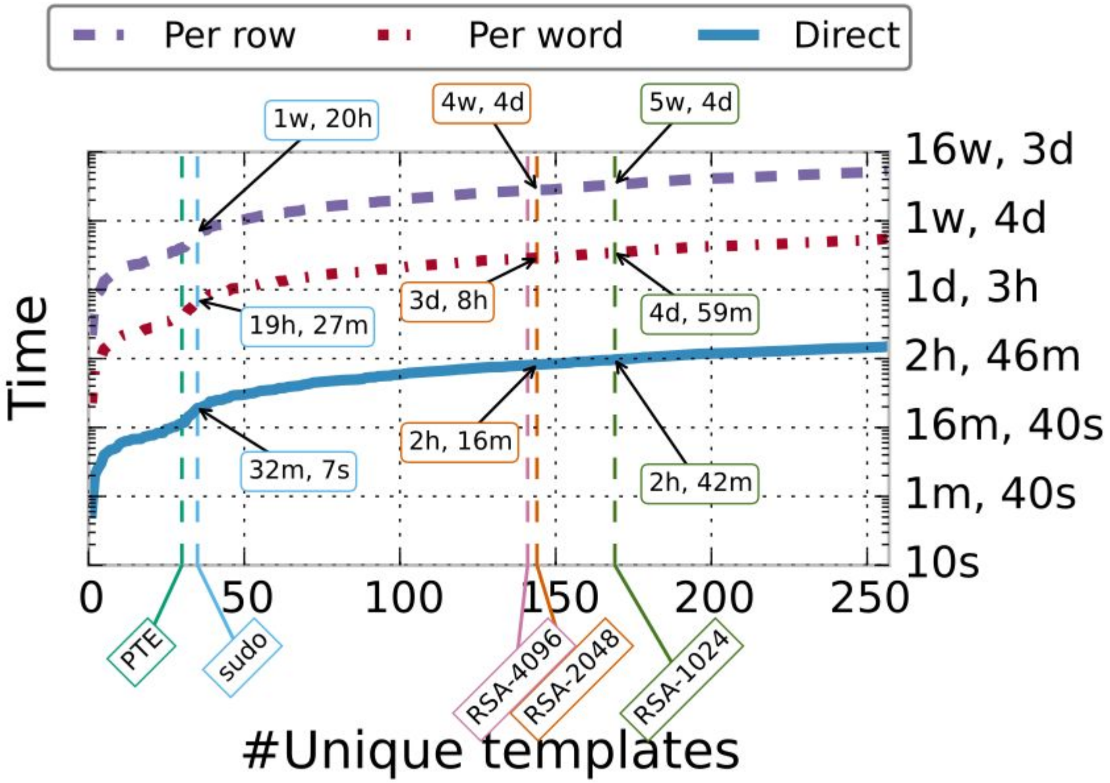
[*]Department of Computer Science
Vrije Universiteit Amsterdam, The Netherlands
m.kurth@vu.nl, beng@cs.vu.nl, da.andriesse@few.vu.nl
{kaveh, herbertb, giuffrida}@cs.vu.nl

[§]Department of Computer Science
ETH Zurich, Switzerland
kurthm@ethz.ch

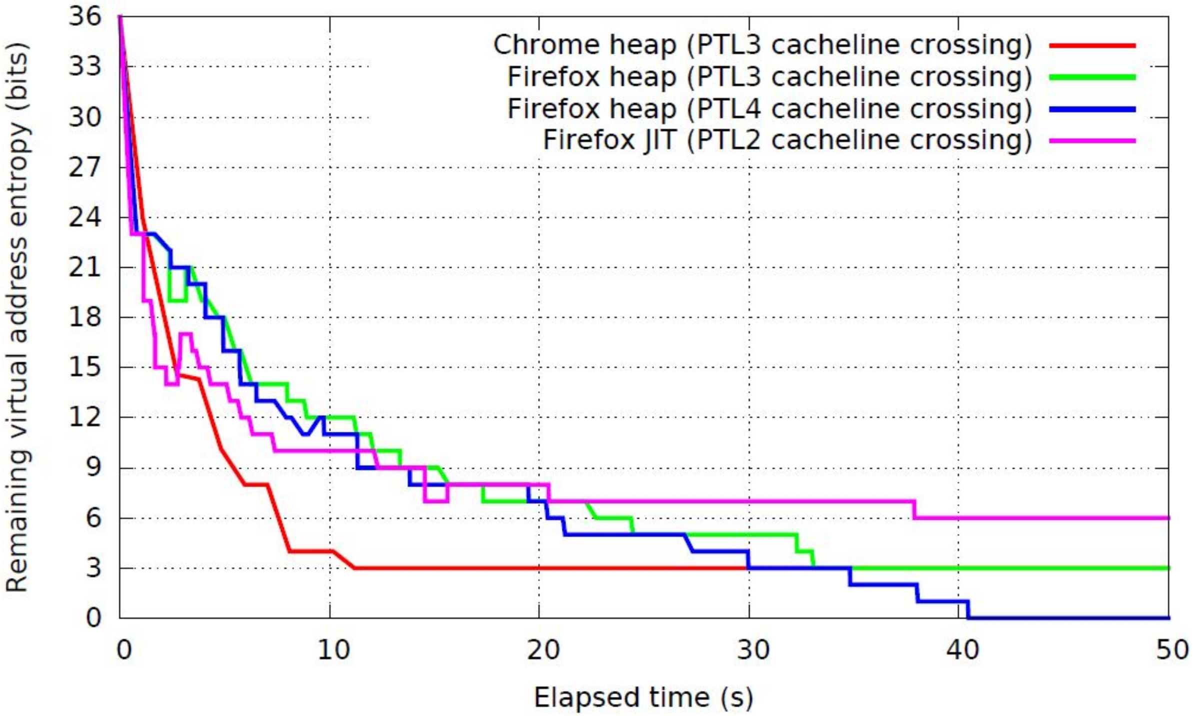
Evaluation: Redis Throughput at Saturation



ECC: Replicating Existing Attacks



Reducing ASLR Entropy



(p)TRR

- Original paper (PARA): on DRAM row activation refresh adjacent rows with a certain probability
  - Found to be effective
- (LP)DDR4 standard: count activations and refresh adjacent rows
- Many different implementations
  - Some look insecure, deployability? (current work)

Proposed Defenses

Disabling features:

- Deduplication
- ION heaps

Hardware defenses:

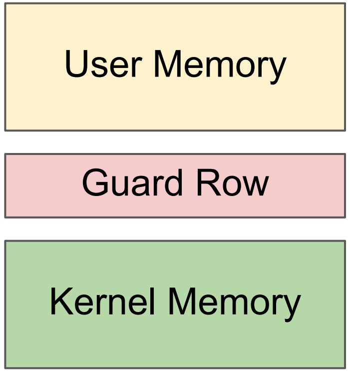
- ECC
- (p)TRR

Software defenses:

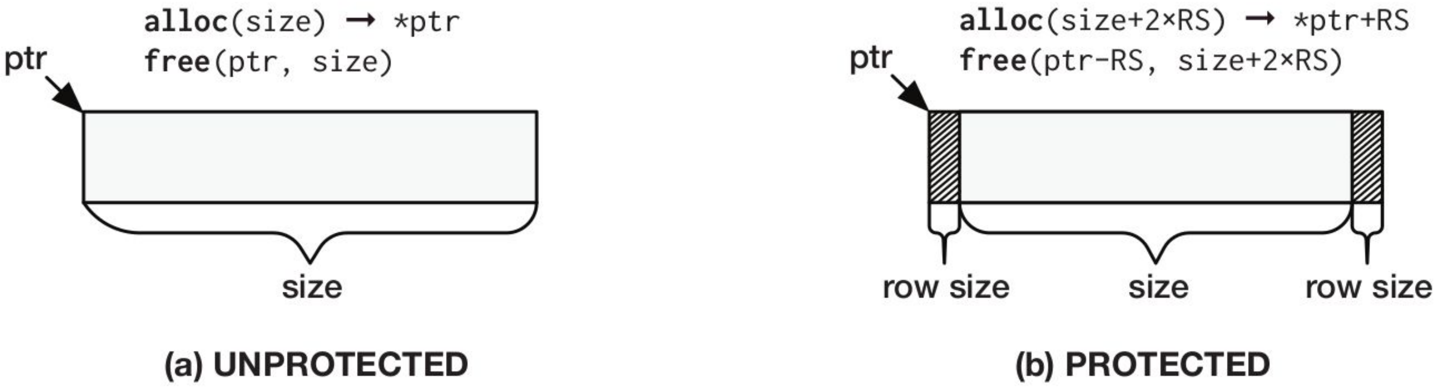
- ANVIL (templating)
- CATT (memory massaging)

Software Defenses

- ANVIL (ASPLOS’16): software TRR
  - Requires hardware-specific Intel feature
- CATT (SEC’16): separate kernel-user memory
  - Only protects the kernel
  - Limits memory management
  - Page-cache attacks



Securing DMA Memory

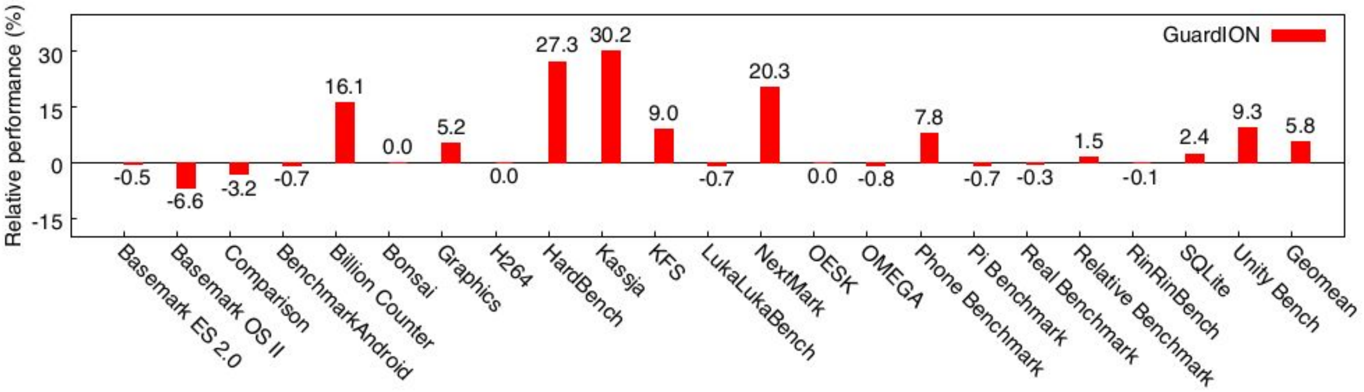


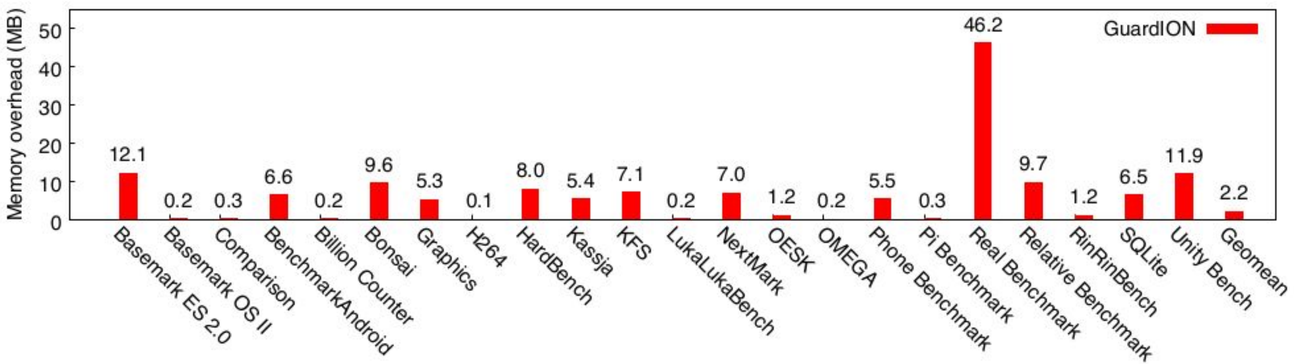
Android kernel patch: 844 LoC

Re-enabled ION contig. heap

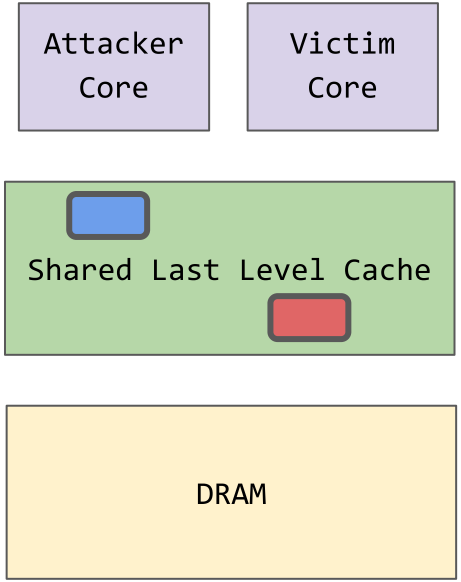
Van der Veen et al., “GuardION: Practical Mitigation of DMA-based Rowhammer Attacks on ARM,” DIMVA’18

Evaluation Results





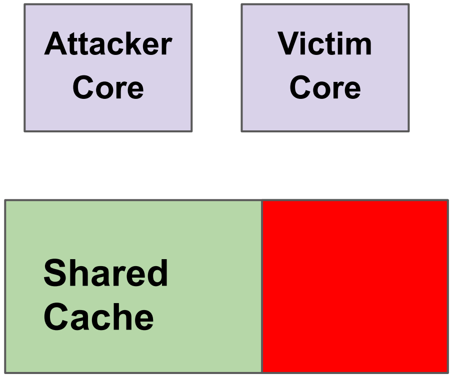
Traditional Cache Attacks



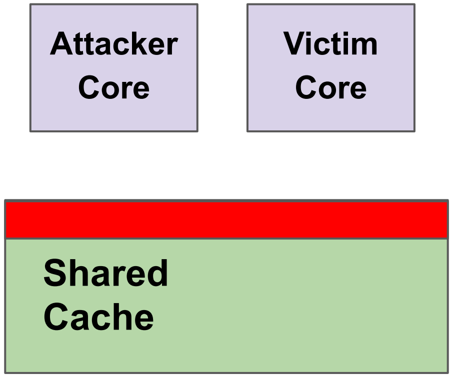

```
if (secret_value == 1)
{
      something();
}
else
{

      something_else();
}
```

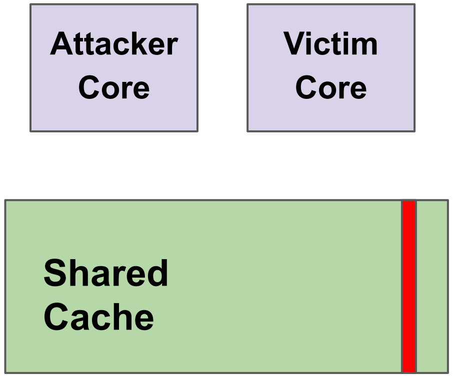
Proposed Defenses: Cache Partitioning



COLORIS
Page Coloring
PACT'14



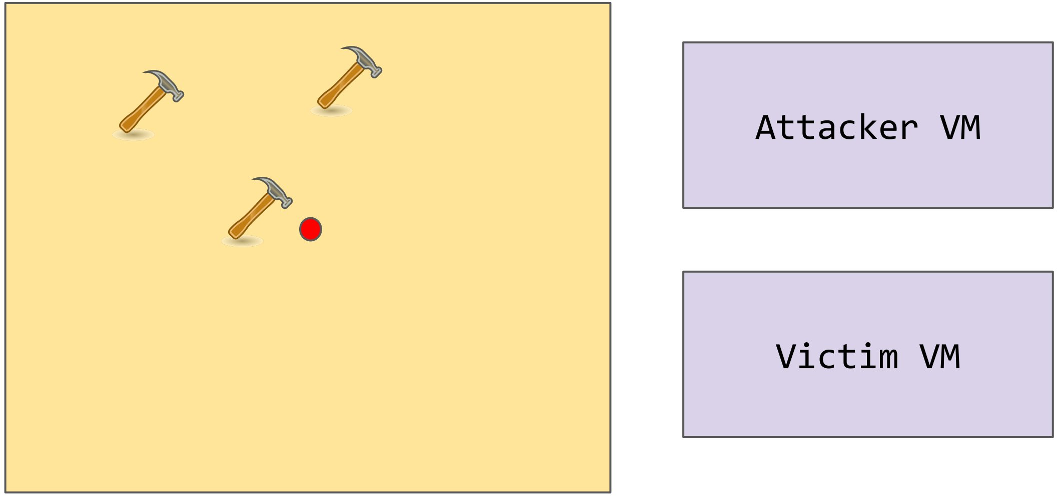
CATalyst
Intel CAT
HPCA'17



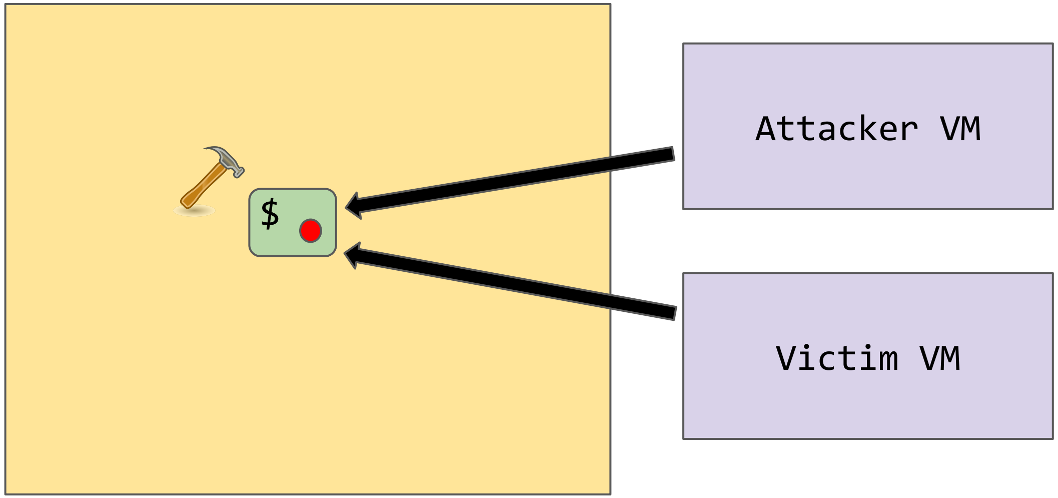
Cloak
Intel TSX
SEC'17

Backup: Other Components

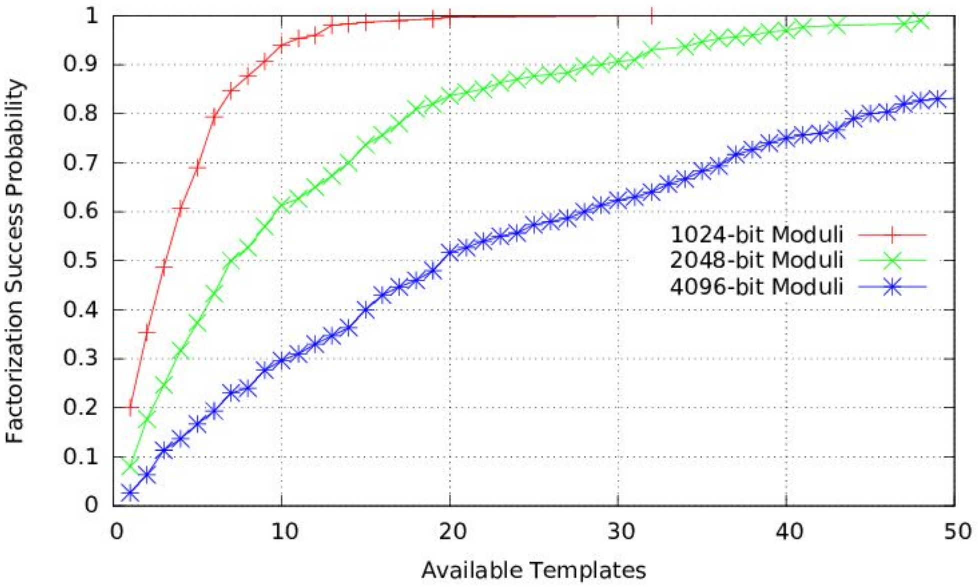
Templating



Memory Deduplication



Factorizing Corrupted RSA Public Keys



The Drammer Attack

1) Templating

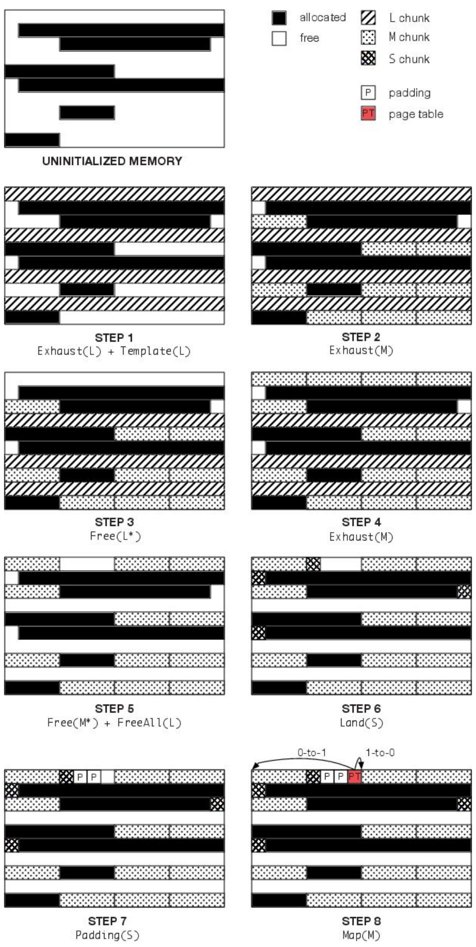
ION DMA memory allocation

2) Massaging

Predictable behavior of buddy allocator

3) Exploitation

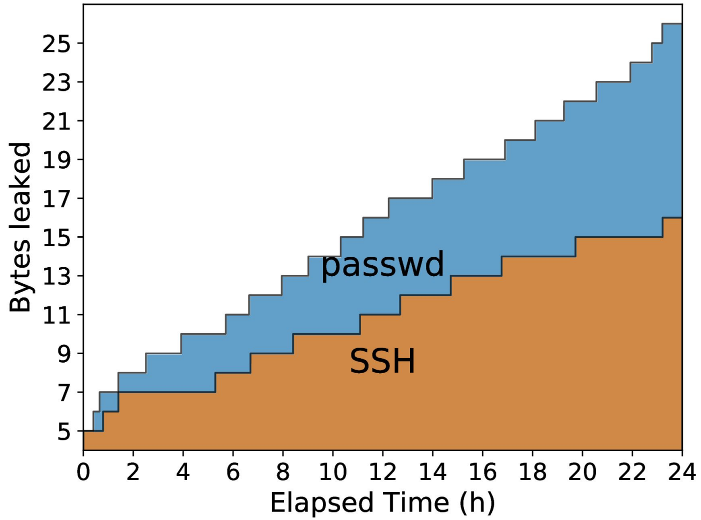
Corrupting page tables



Van der Veen et al., “Drammer: Deterministic Rowhammer Attacks on Mobile Platforms,” CCS’16

Leaking /etc/shadow with RIDL

Deep optimizations in the CPU pipeline

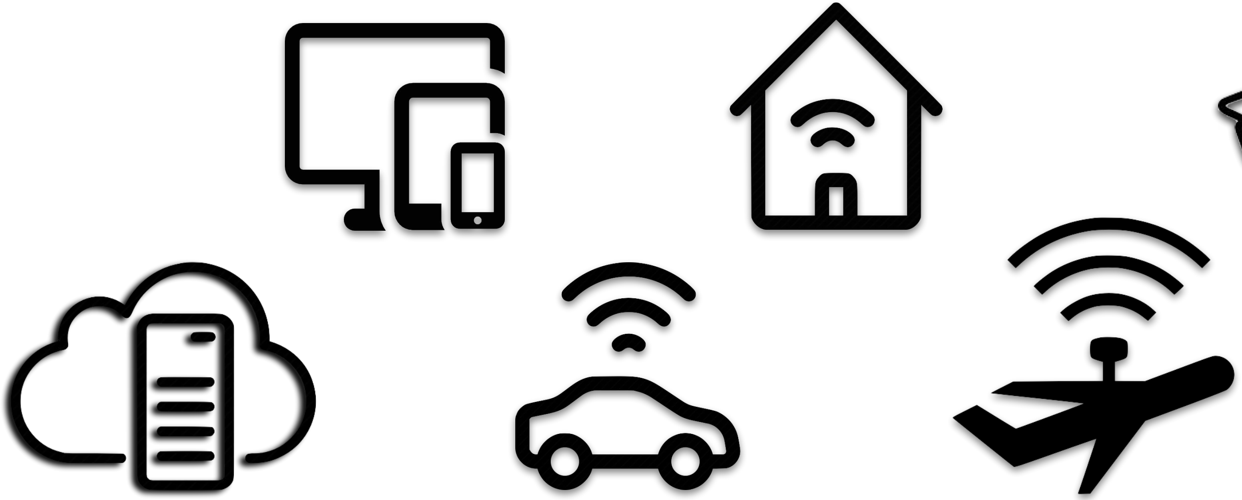


Industry-wide mitigation efforts underway.

Van Schaik et al., “RIDL: Rogue Inflight Data Load,” S&P’19

$600 Billion Lost to Cyber Crime in 2018

Lots of efforts on securing systems ($114 Billion in 2019)



Securing Software (2000-)

Software Update

seL4
Security. Performance. Proof.

Assuming secure software, what is still possible?
And what can we do about it?