Security: Cache and DRAM Attacks

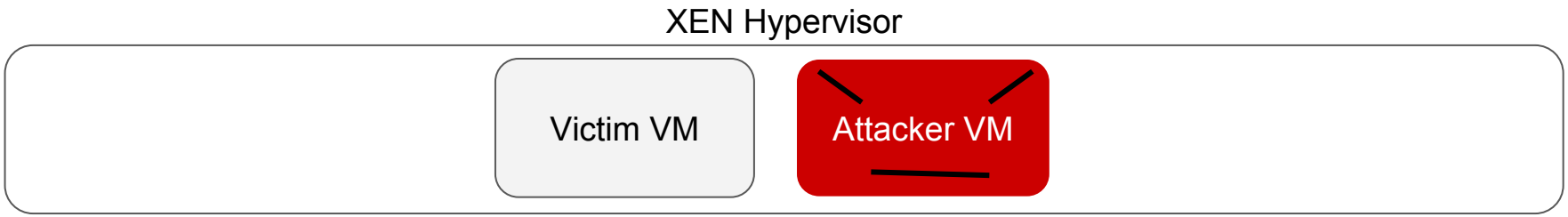
A short synthesis of 3 Usenix papers

Cyril ANAK STELL Josué ALVAREZ Aminata DIOP Cécile DUTHOIT Linn MJELSTAD Clovis OUEDRAOGO

One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation

Yuan Xiao, Xiaokuan Zhang, Yinqian Zhang, and Radu Teodorescu
The Ohio State University

1/3: Bit flip, cloud flop



Exploit Principle

Goal : Gain read-write access from **Attacker VM** to any memory location on **Victim VM**

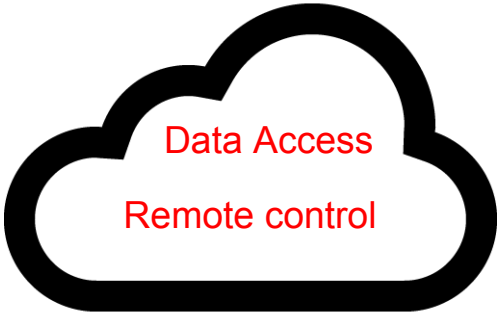
Exploit Limitations

Not reliable with ECC DDR4 DRAM

Works on XEN hypervisor **without** hardware-assisted virtualization

/!\ IoT Danger zone /!\

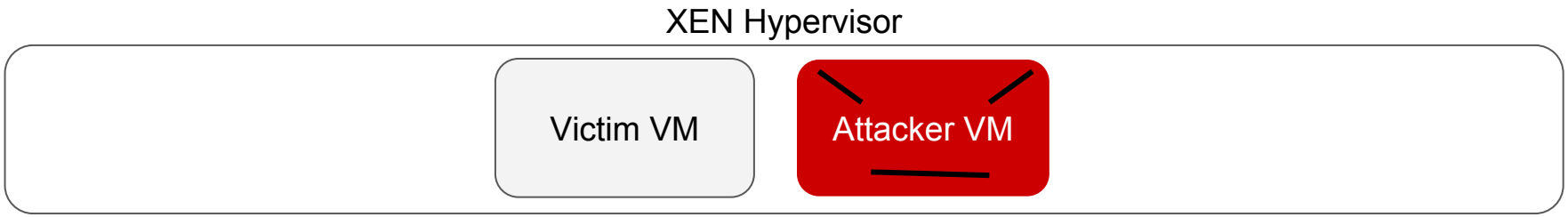
IoT Platforms hosted on cloud



Leak of personal or enterprise sensor data hosted on cloud

Perform malicious remote actions on connected objects

1/3: Bit flip, cloud flop



Exploit Roadmap

Goal : Gain read-write access from **Attacker VM** to any memory location on **Victim VM**

Step 1. Map Physical Address to specific DRAM rows.	Step 2. Check for vulnerable bits	Step 3. Flip bits in hypervisor page tables	Win
Exploit *timing channel*	DRAM vulnerability with Row Hammering	Hypervisor memory mapping vulnerability	

Win

Example usages :

- Bypass SSH authentication
- Private key extraction from HTTPS Servers

1/3: Bit flip, cloud flop

Timing hidden-channel exploit

Goal : flipping specific bits at determined physical addresses

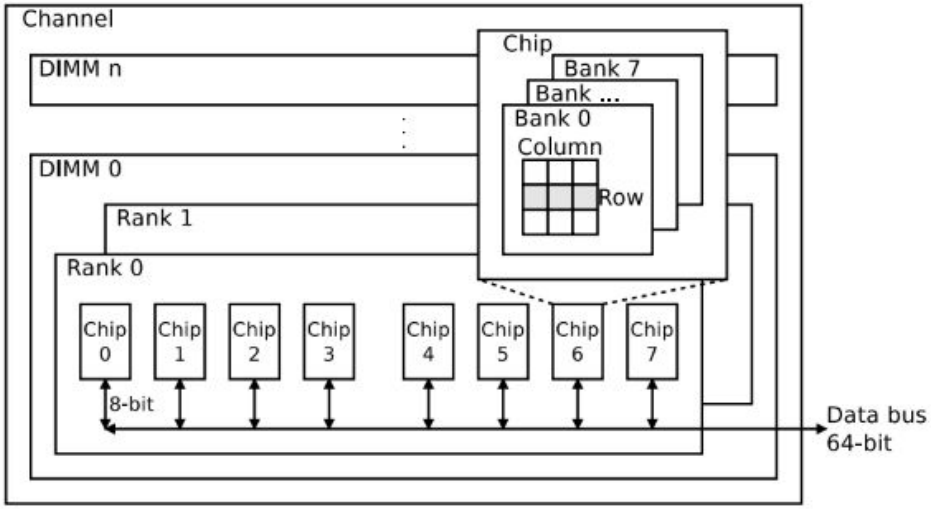
Map Physical Address to specific DRAM rows.

Repeat access to 2 memory addresses

Short latency : different rows

Long latency : same row

=> bank, row, column bits for each address.



1/3: Bit flip, cloud flop

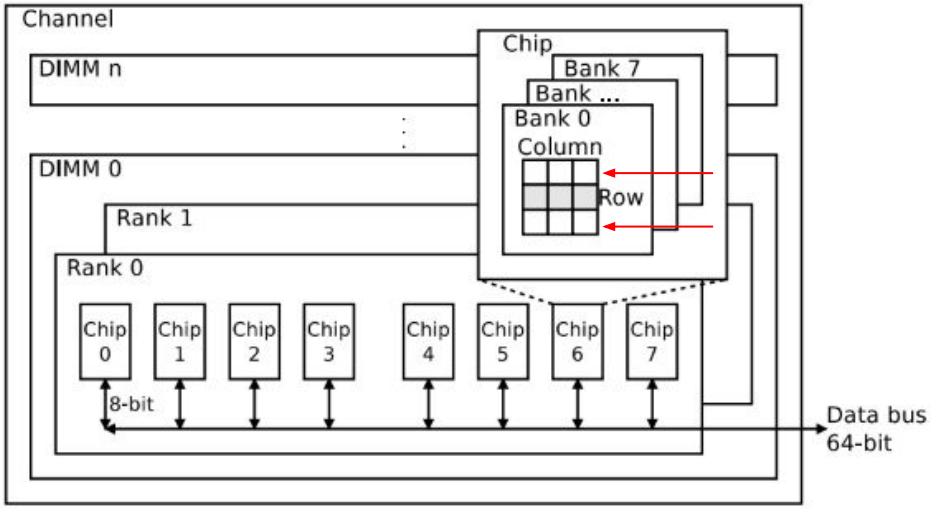
DRAM Hardware exploit

Goal : check which changes we can actually perform

Check for vulnerable bits

Perform writes on neighboring rows

Determine which bit are likely to flip



1/3: Bit flip, cloud flop

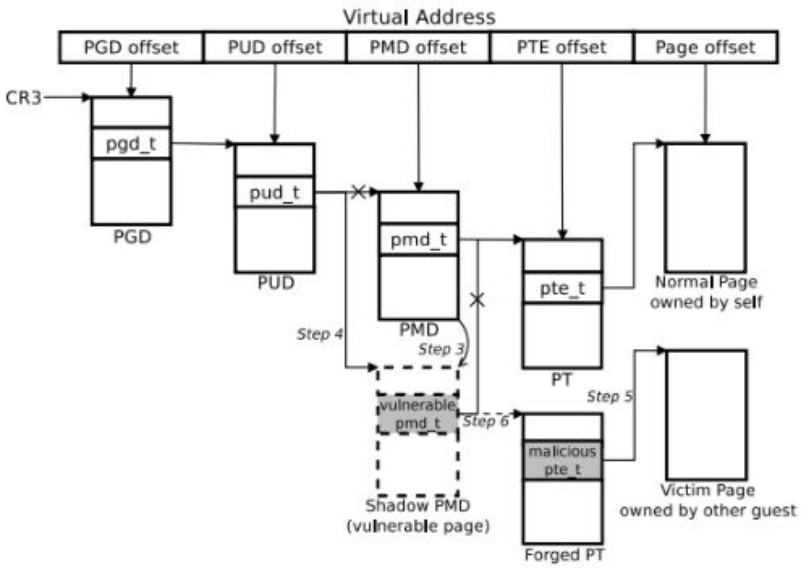
Hypervisor address space management exploit

Goal : access to VictimVM pages

Flib bits in hypervisor page tables

1. Virtual address of the attacker will be mapped to another physical address

2. Read / Write anything on any physical address !



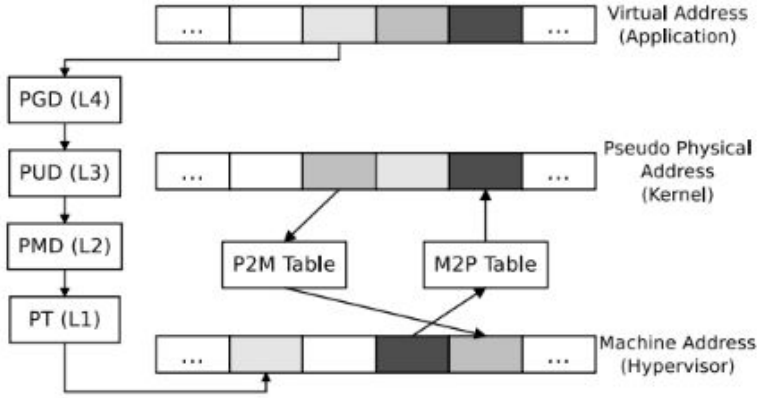


Figure 6: Memory management of Xen paravirtualized VMs.

ARMageddon: Cache Attacks on Mobile Device

Moritz Lipp, Daniel Gruss, Raphael Spreitzer, Clémentine Maurice, and Stefan Mangard

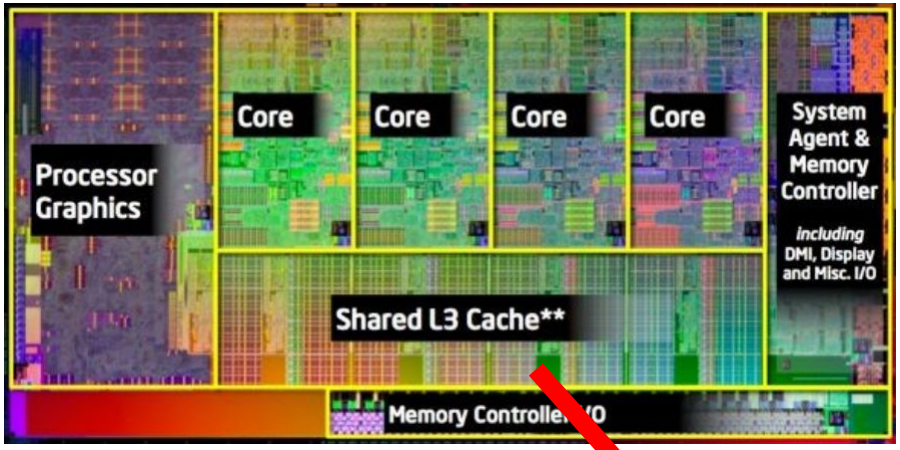
Graz University of Technology

2/3: Mobile devices - *how to know if a program has been used ?*

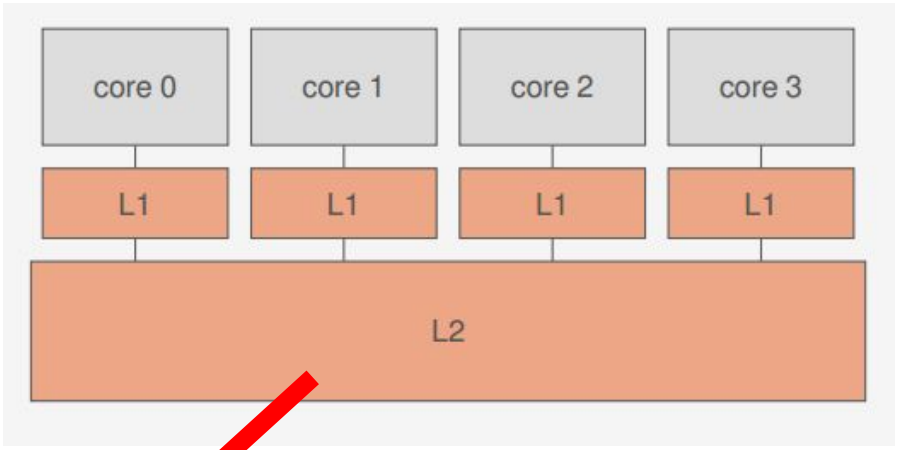
Modern smartphones use one or more multi-core ARM CPUs

→ **Cache attacks** use the **weaknesses of the hardware design** of CPUs

Caches on Intel CPUs



Caches on ARM Cortex-A CPUs



Cache levels shared globally across all users and privilege levels

2/3: Mobile devices - *how to know if a program has been used ?*

Prerequisite:

Know the victim application

Choose a common lib/file/etc.

Steps:

1. Design an app that uses a same lib/file/etc. as the victim program
2. Install it on the victim device
3. Proceed to a cache attack

	Flush + reload JS	Prime + probe JS	Flush + flush JS	Evict + time JS
Victim app has been used if...	FAST Flush cache and reload to check if cache memory has been reloaded by the victim	MISS OCCUPIED SETS Occupy specific cache sets before victim program is scheduled, and check which are still occupied afterwards.	SLOW 2e times Flush and reflush to check if cache memory was reloaded by victim meanwhile	FAST 2e times Evict a specific cache set and compare execution time before and after

Why is it so dangerous ?

Not so difficult...

- No need of any permission
- Can be executed in unprivileged userspace
- No need of a rooted device
- Any Android version (it doesn't exploit specific vulnerabilities of Android versions)

No solution to protect our phones for now (and more globally any device)...

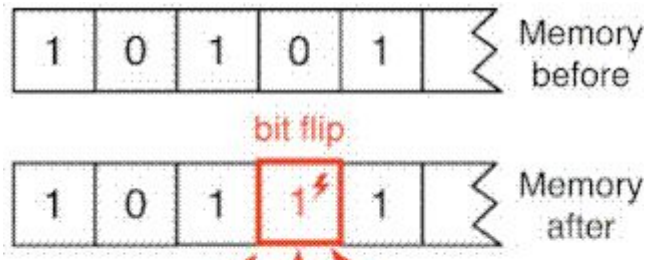
Flip Feng Shui: Hammering a Needle in the Software Stack

Kaveh Razavi, Ben Gras, and Erik Bosman
Vrije Universiteit Amsterdam
Bart Preneel, Katholieke Universiteit Leuven; Cristiano Giuffrida and Herbert Bos
Vrije Universiteit Amsterdam

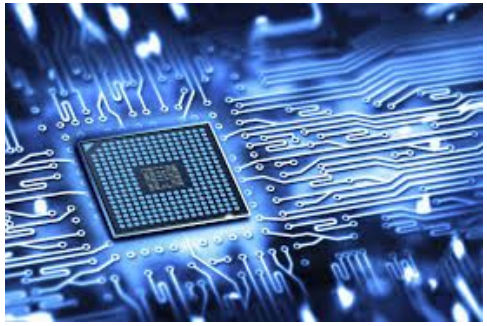
3/3: FFS, Hammering a needle in the software attack

➢ **What is it ?**

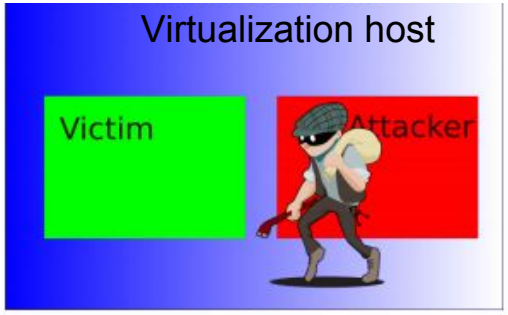
Bit flip



➢ **What does the attacker need to do FFS attack**



A vulnerable hardware

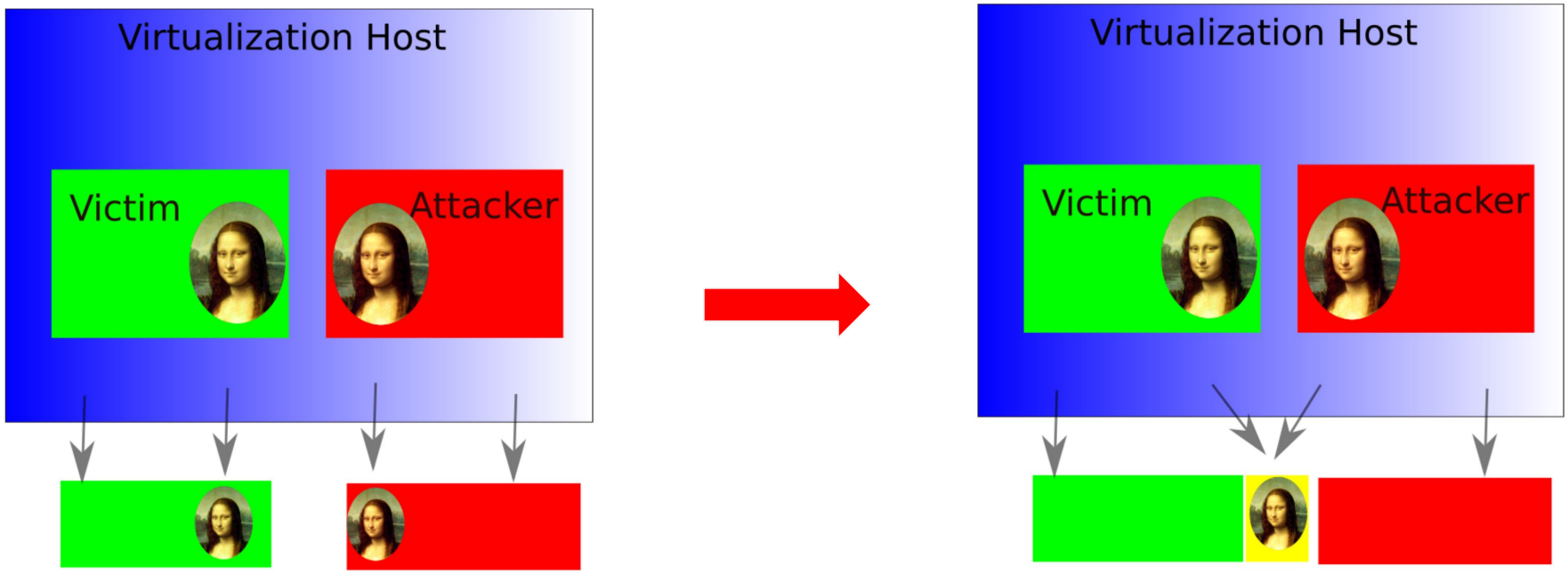


Co-hosted VMs

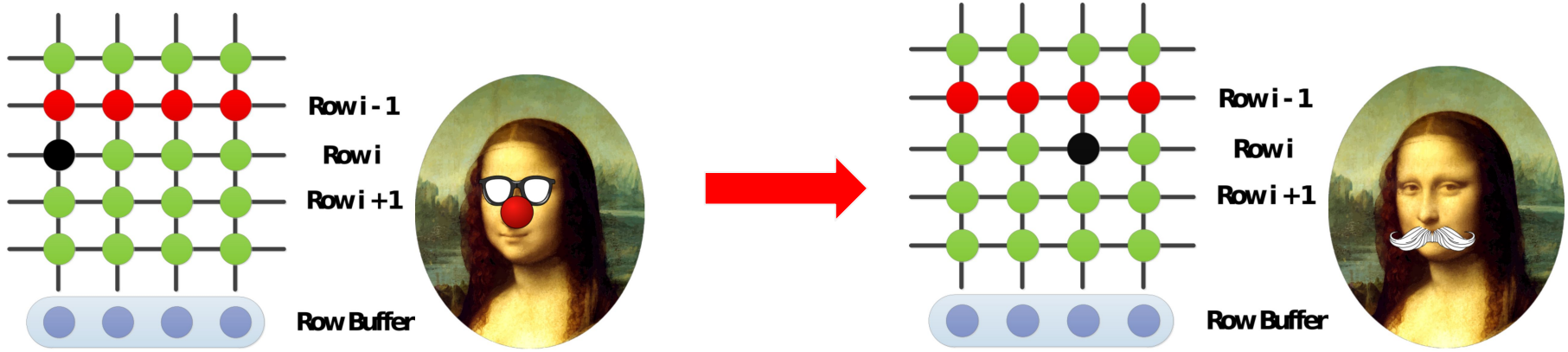
All targeted data in the software stack can be corrupted

3/3: FFS, Hammering a needle in the software attack

Step 1: Memory deduplication

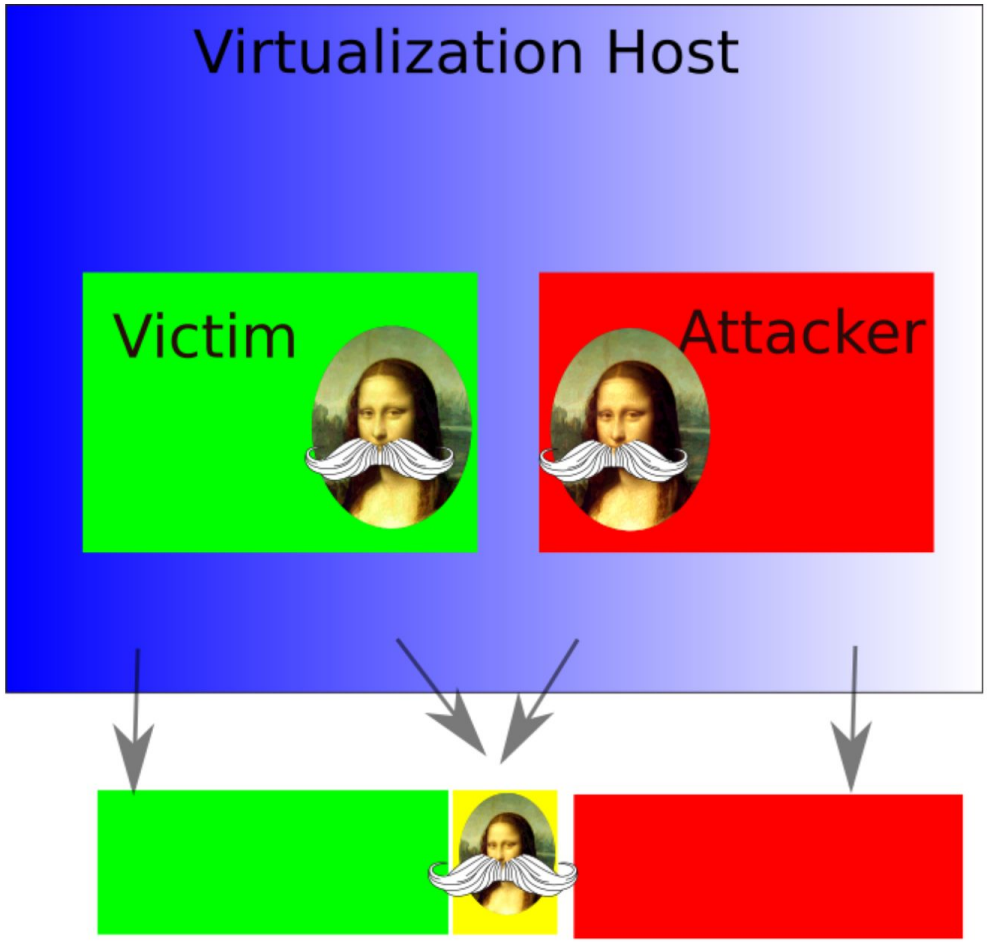


Step 2: Rowhammer attack



3/3: FFS, Hammering a needle in the software attack

Result



Memory deduplication + Rowhammer = Flip Feng Shui = VM compromised!

There is no need for a software bug!

3/3: FFS, Hammering a needle in the software attack

➢ **Exemple: The Ubuntu Update Attack**

- **8192** trials (both 1-to-0 and 0-to-1 flips) to factorize

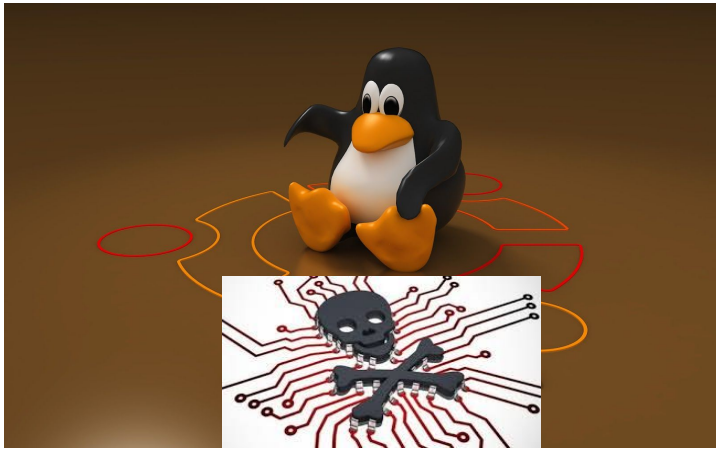
  344 templates of Ubuntu automatic RSA keys (page cache entry of trusted.gpg file.

- Find a bit flip in the URL of the Ubuntu update servers in the page cache entry for apt's sources.list file.

  29 templates result in a valid domain name
  Exemple : ubun**v**u.com

- Wrong RSA key injected + wrong DNS controlled = Wrong package injectable

Updating package



Thanks!

Any questions?