CAIN:
Silently Breaking ASLR in the Cloud

Antonio Barresi, Kaveh Razavi, Mathias Payer, Thomas R. Gross

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

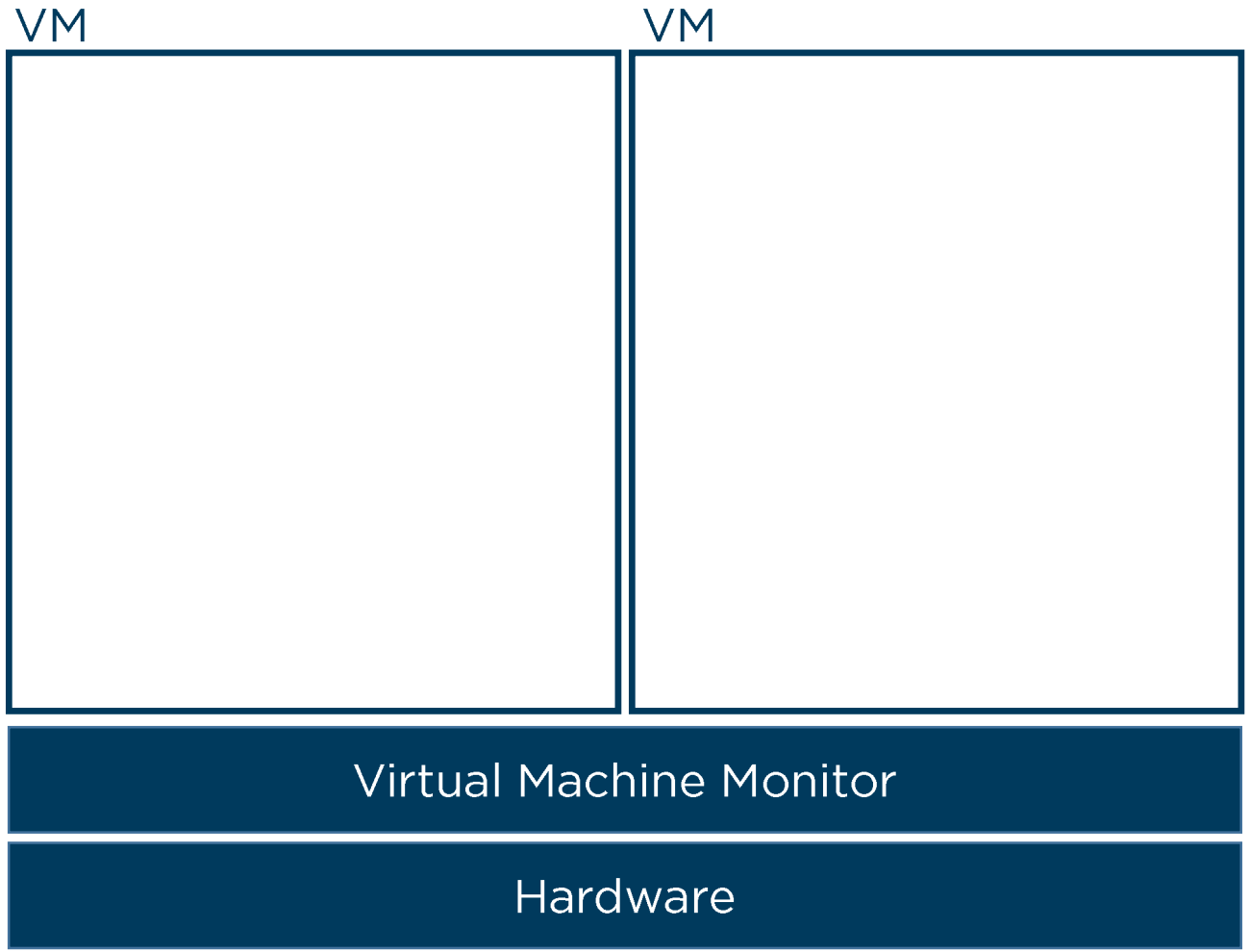
VU VRIJE UNIVERSITEIT AMSTERDAM

PURDUE UNIVERSITY

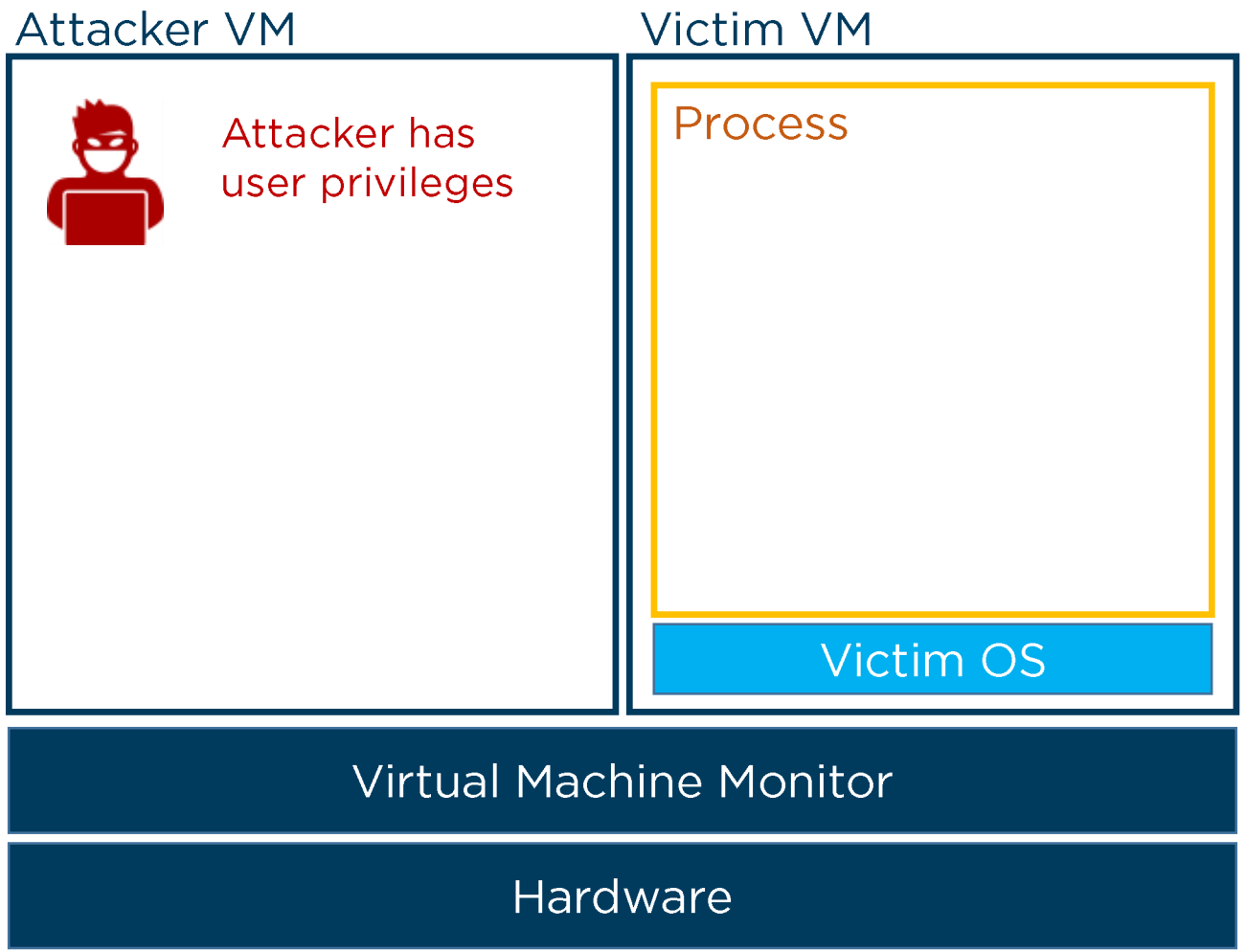
WOOT'15 - 9th USENIX Workshop on Offensive Technologies

Washington D.C., August 11 2015

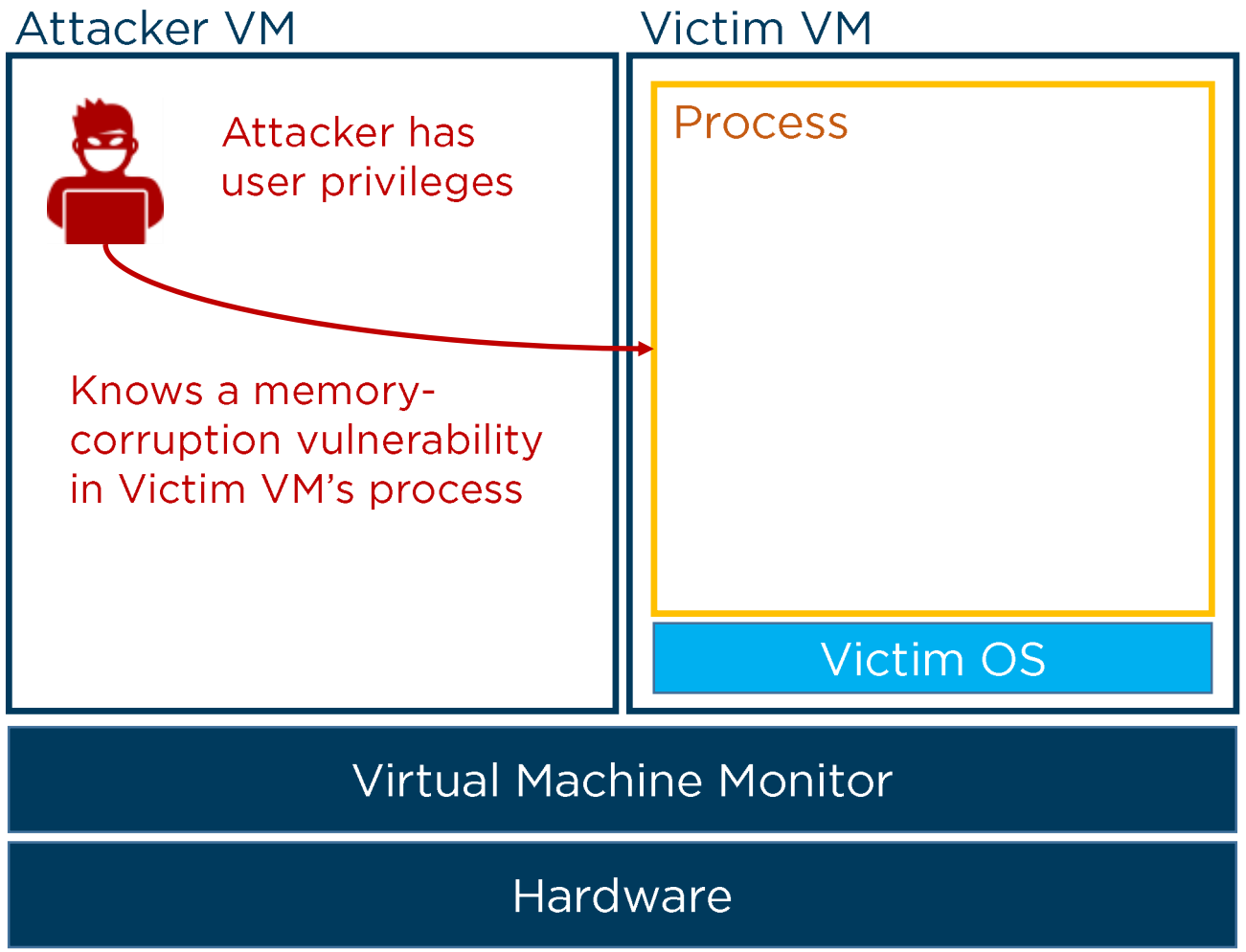
Threat scenario



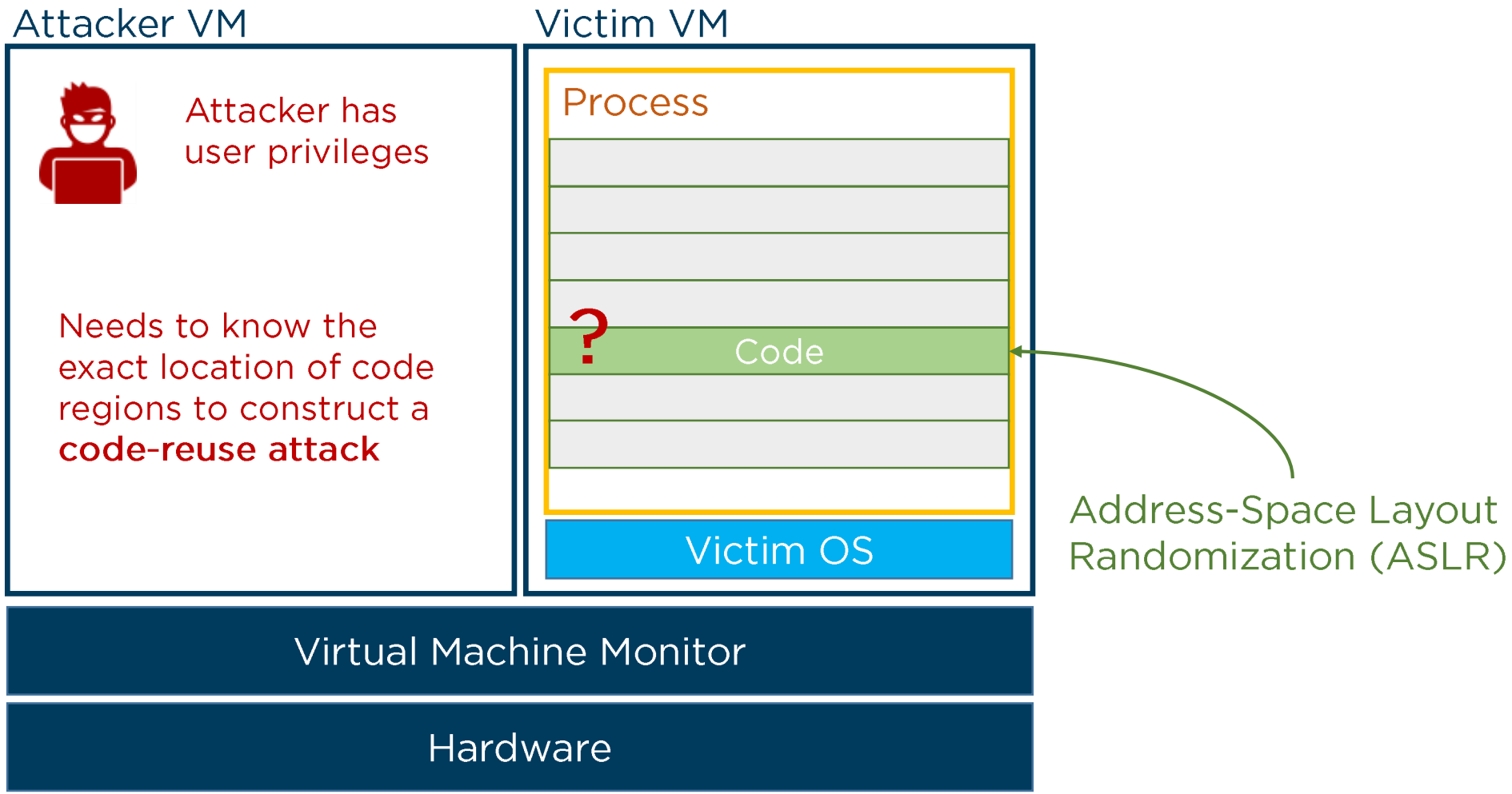
Threat scenario



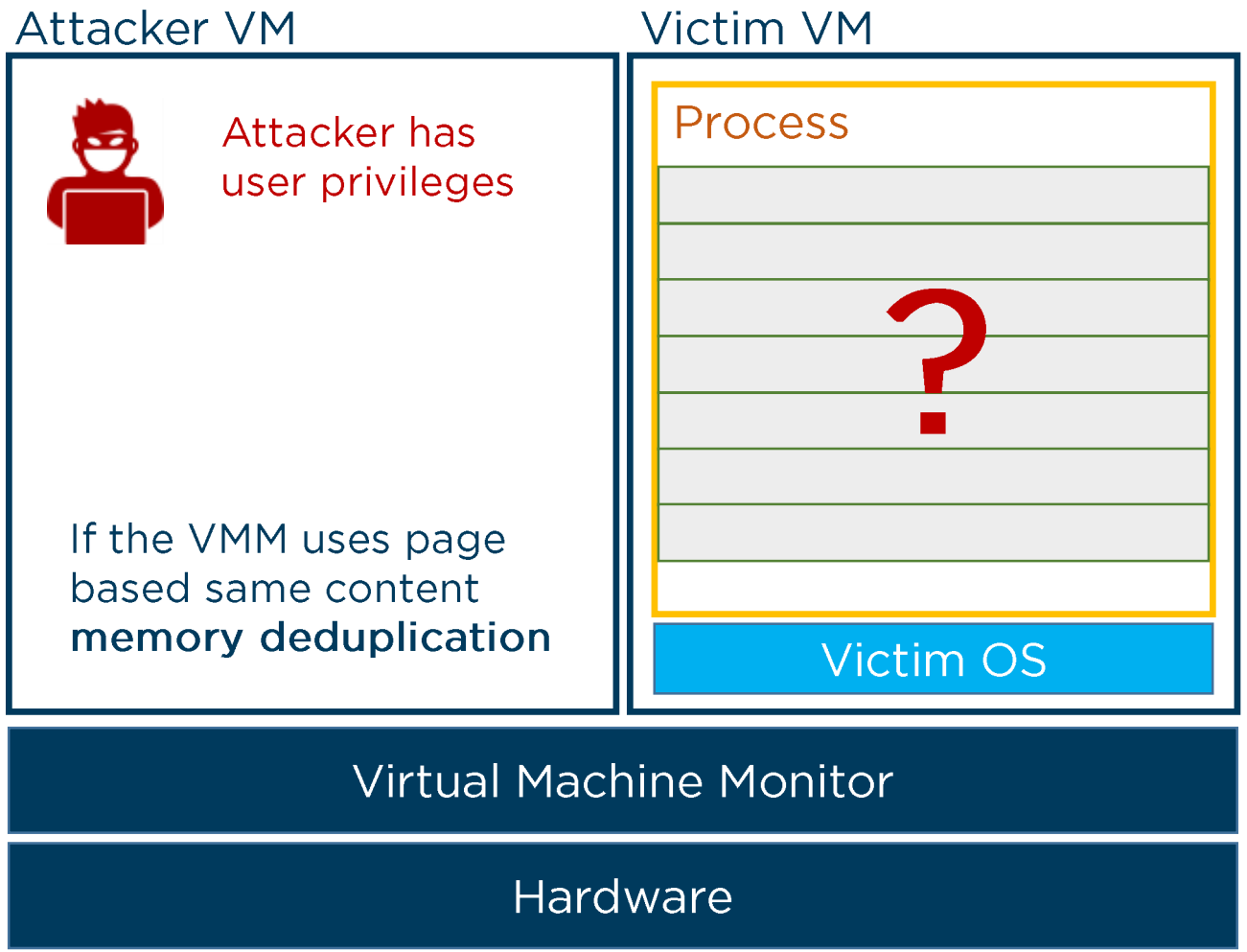
Threat scenario



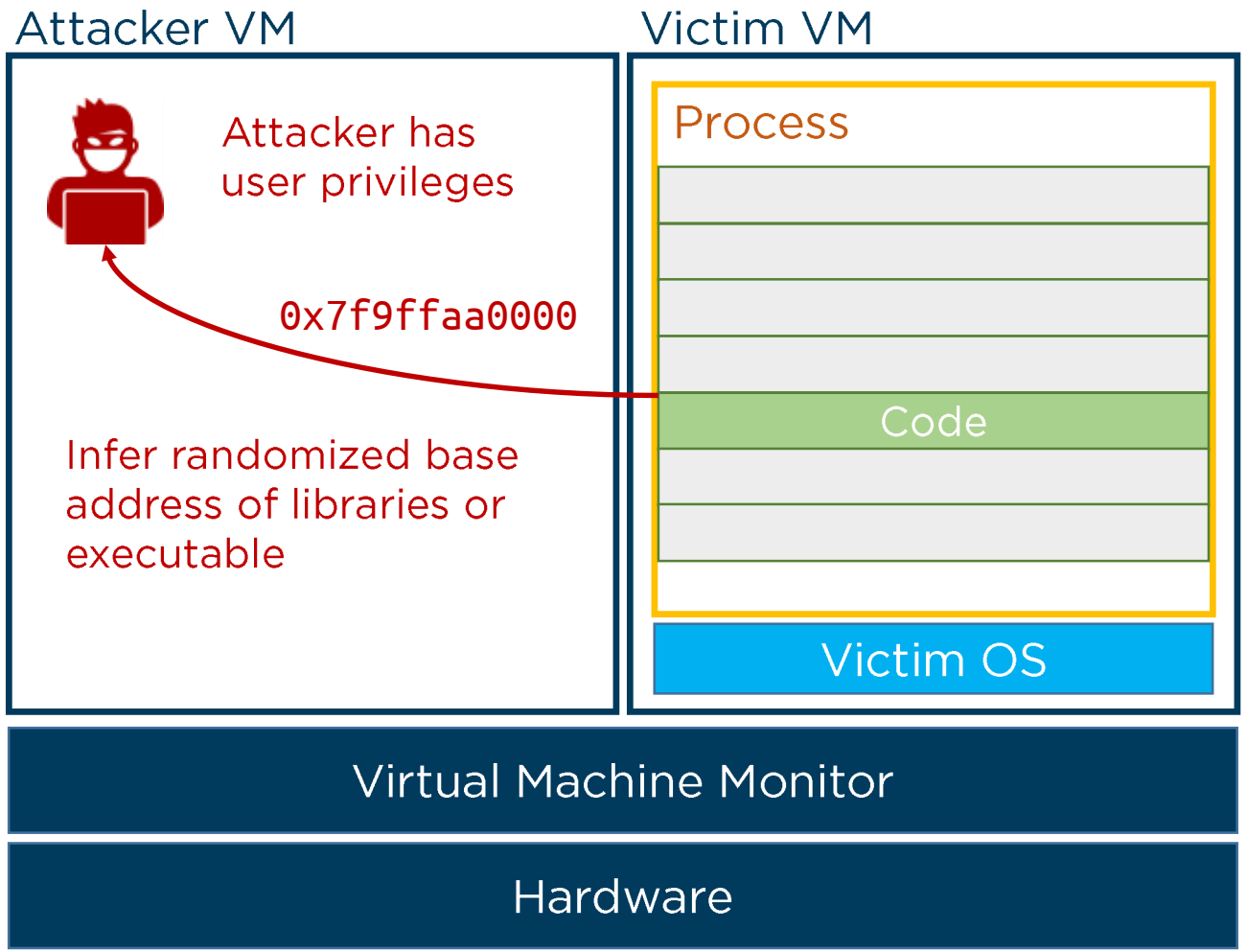
Threat scenario



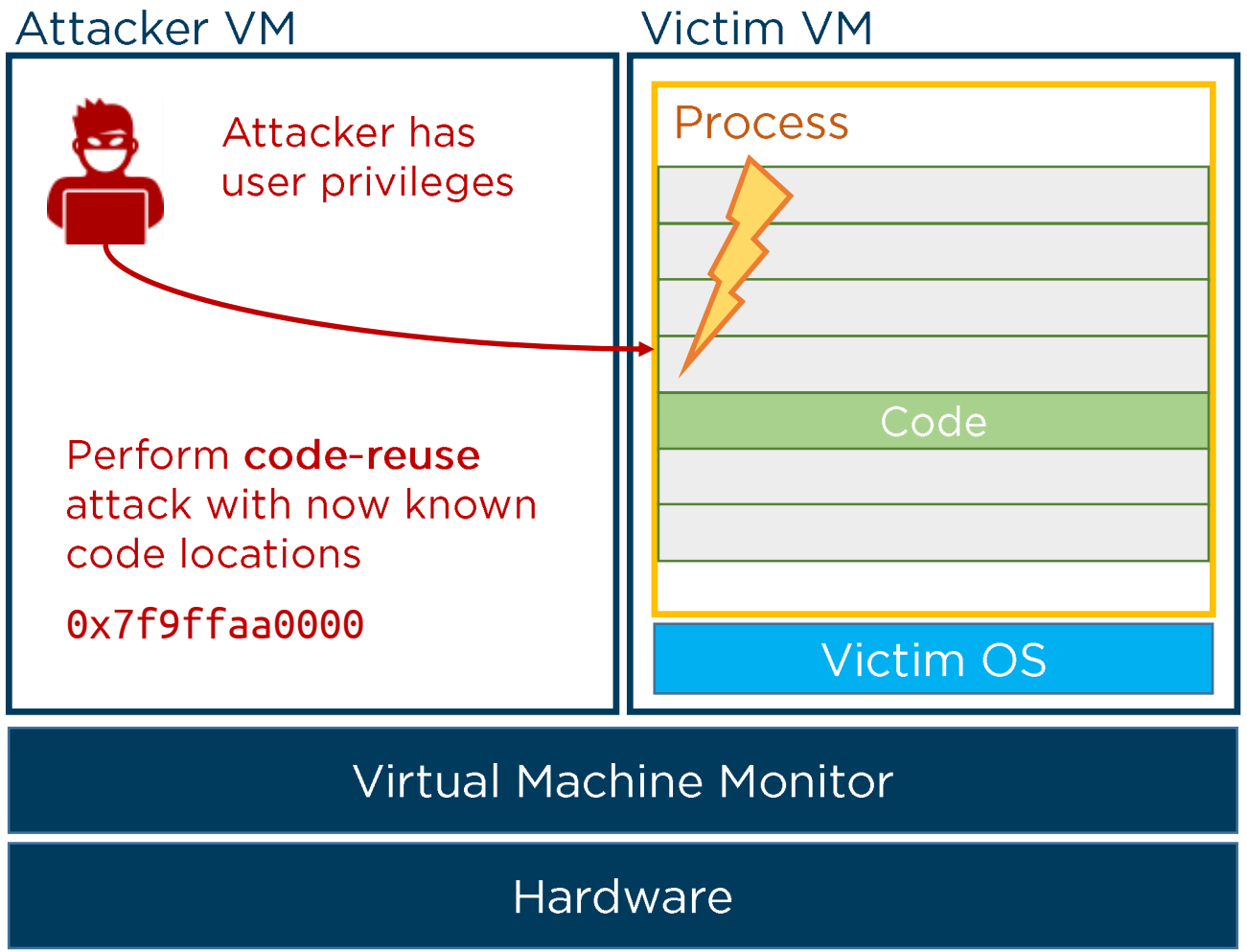
Threat scenario



Threat scenario



Threat scenario

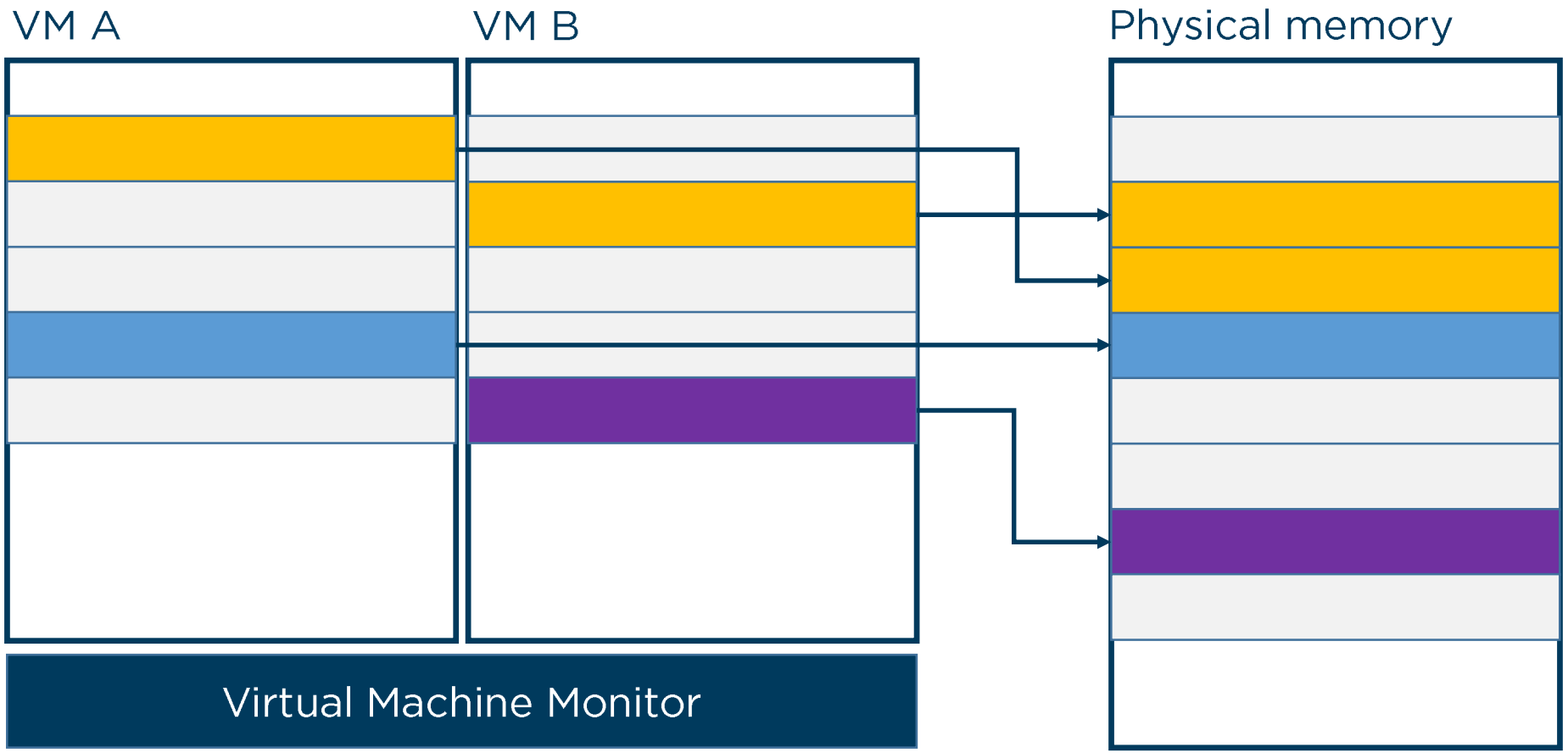


CAIN: Cross-VM ASL INtrospection

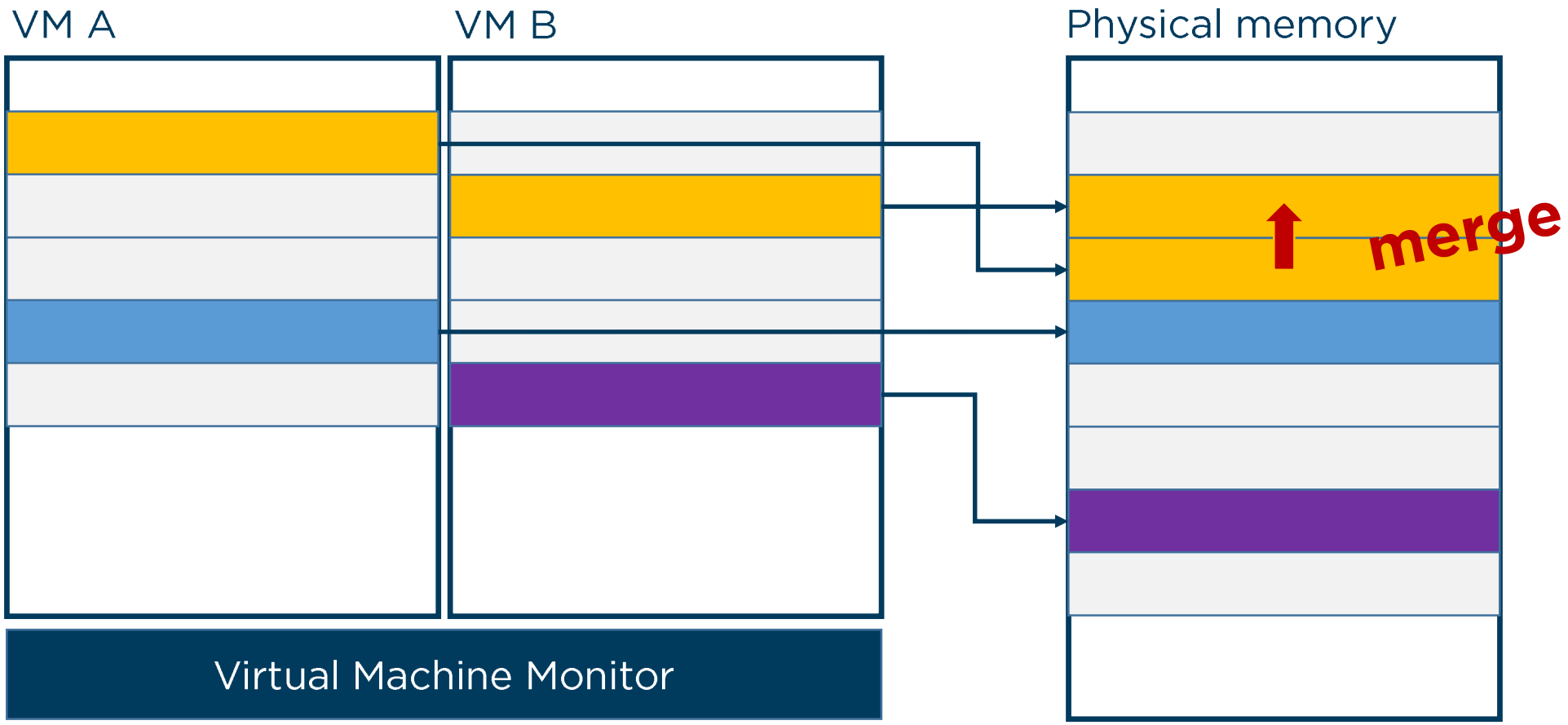
- New attack vector against memory deduplication

- Leaks **randomized base addresses** (RBAs) of
  - libraries and
  - executables
  - mapped in processes running on neighboring VMs

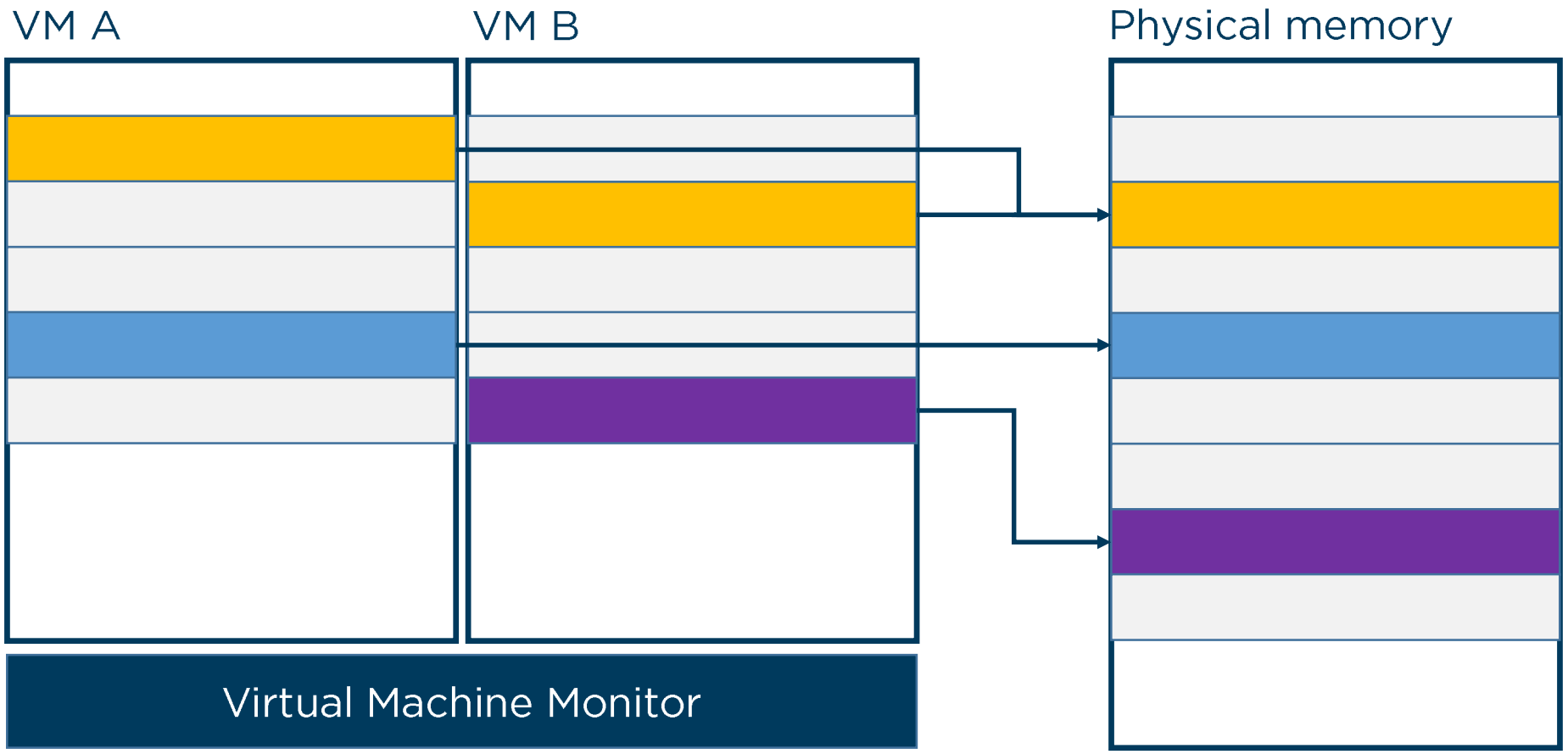
Memory deduplication (in VMM)



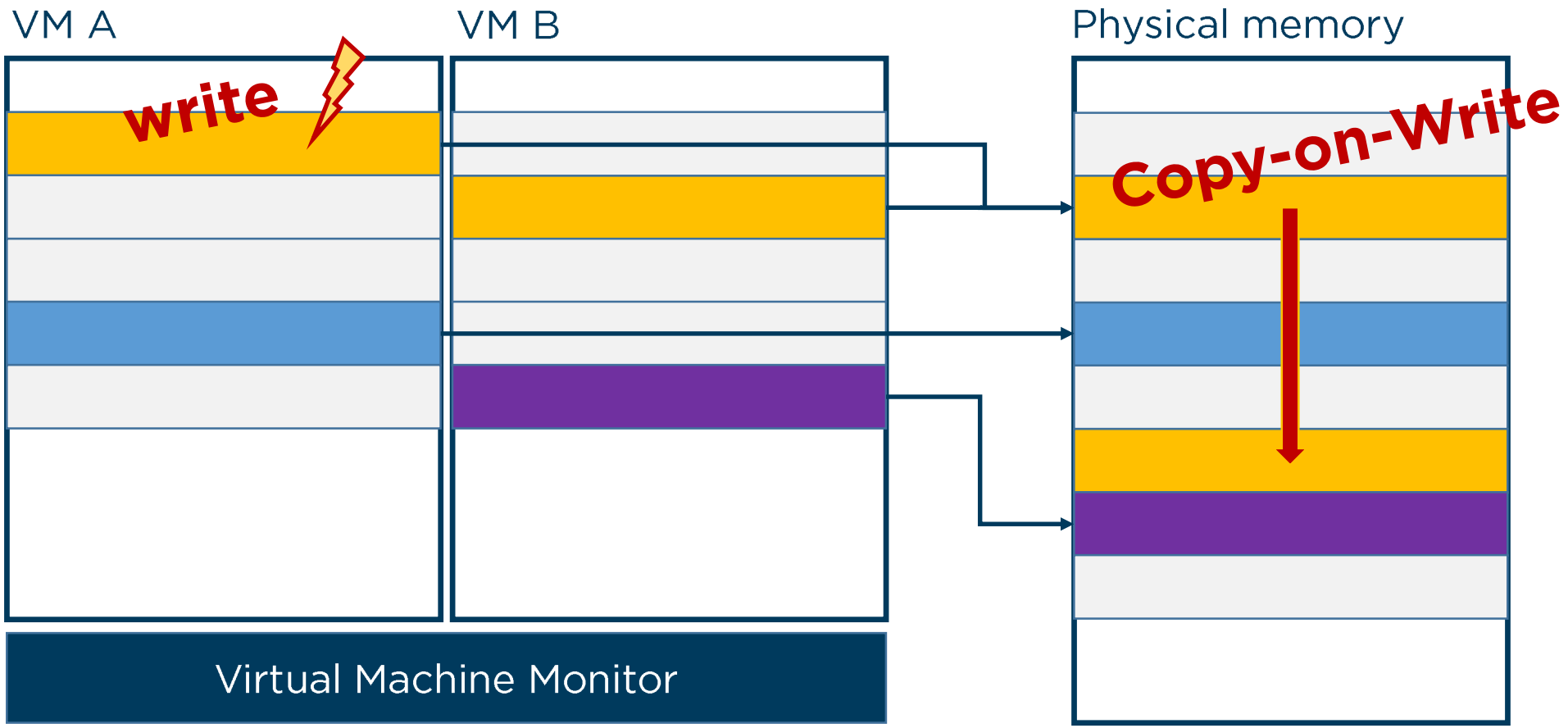
Memory deduplication (in VMM)



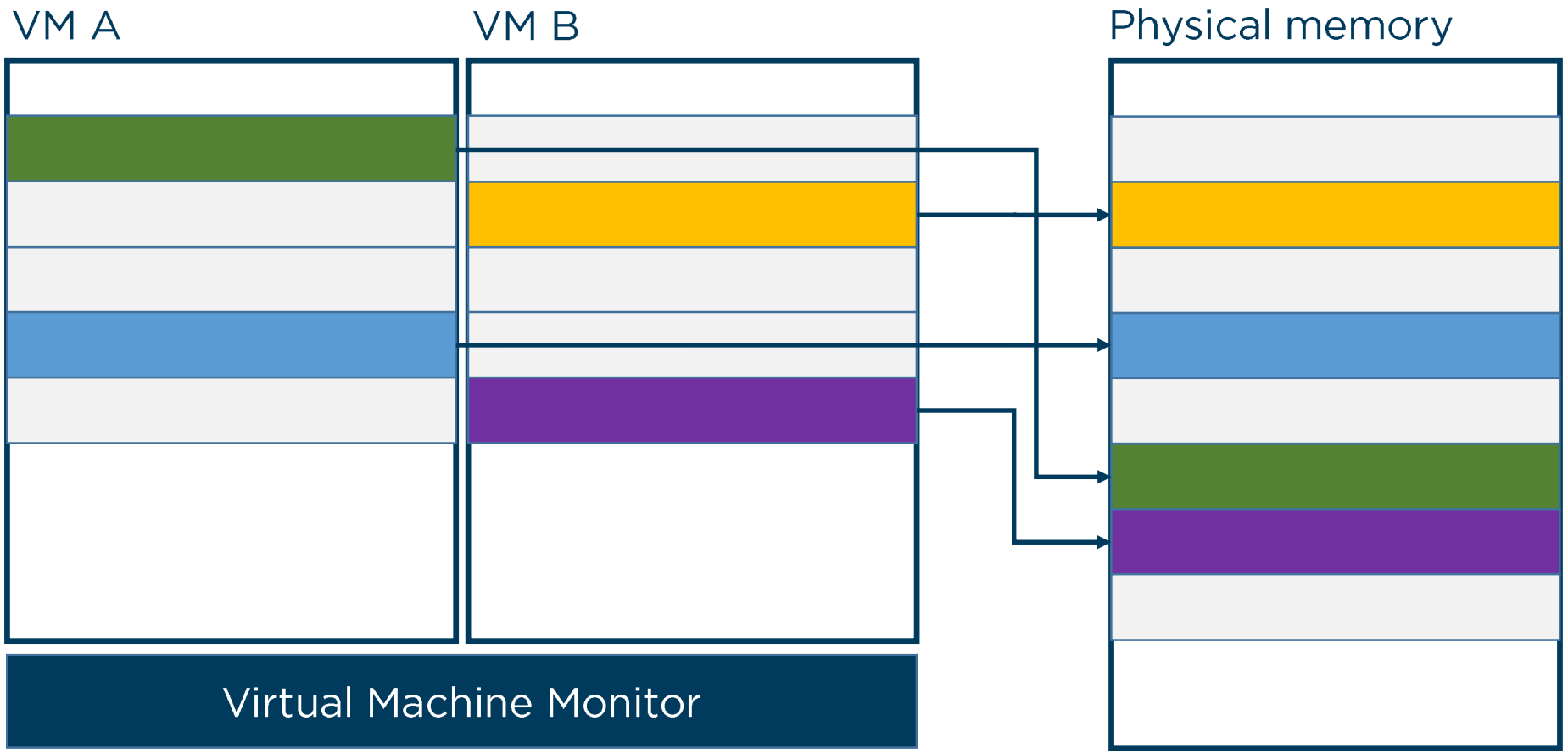
Memory deduplication (in VMM)



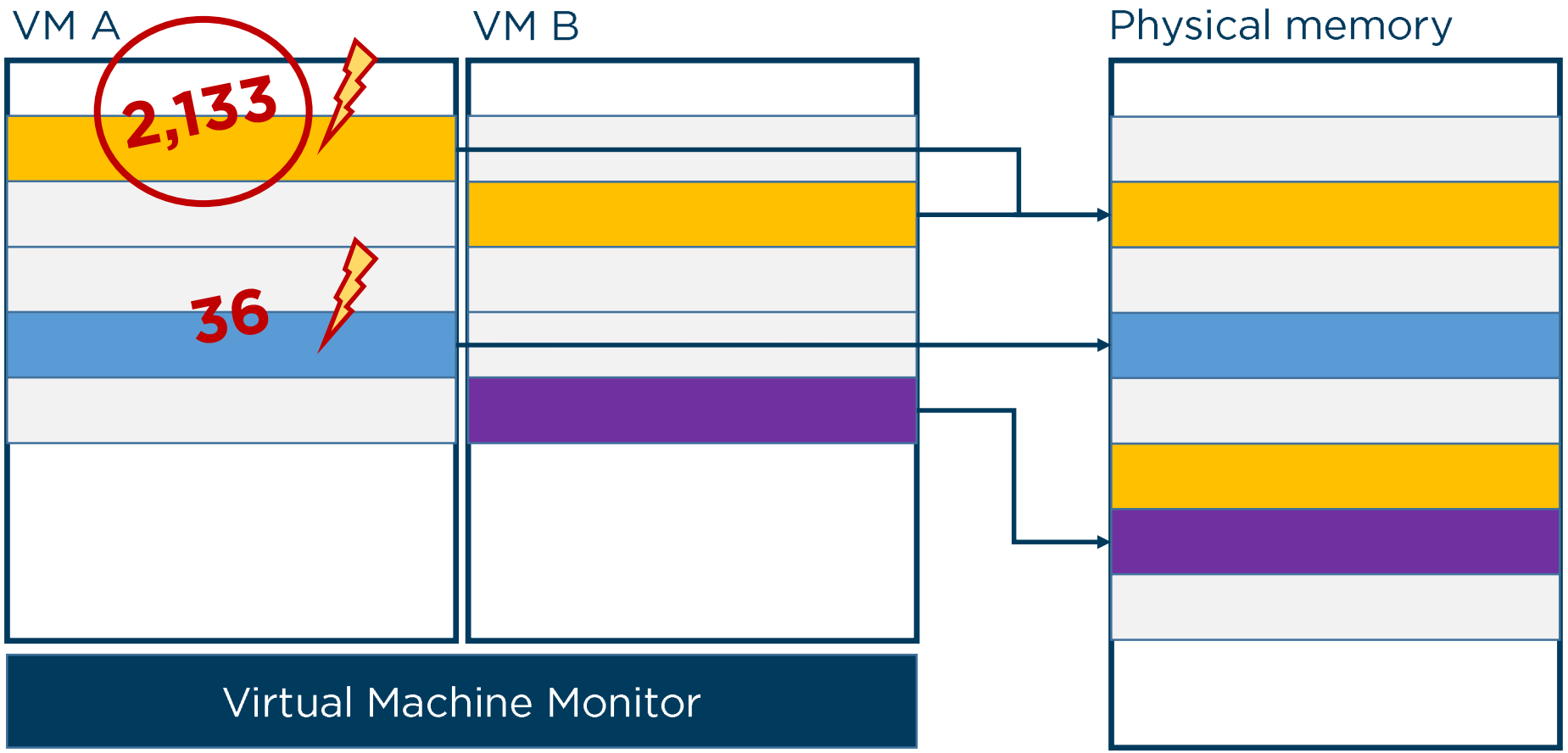
Memory deduplication (in VMM)



Memory deduplication (in VMM)



Memory deduplication side-channel



Memory deduplication side-channel

- Attacker can craft a page (guess)
- Wait for a certain amount of time
- Write to page and measure time
- **Write time will reveal if page was shared**

Memory deduplication side-channel

- Attacker can craft a page (guess)
- Wait for a certain amount of time
- Write to page and measure time
- **Write time will reveal if page was shared**

Page content?

Memory deduplication side-channel

- Attacker can craft a page (guess)
- Wait for a certain amount of time
- Write to page and measure time
- **Write time will reveal if page was shared**

Page content?

How long?

Memory deduplication side-channel

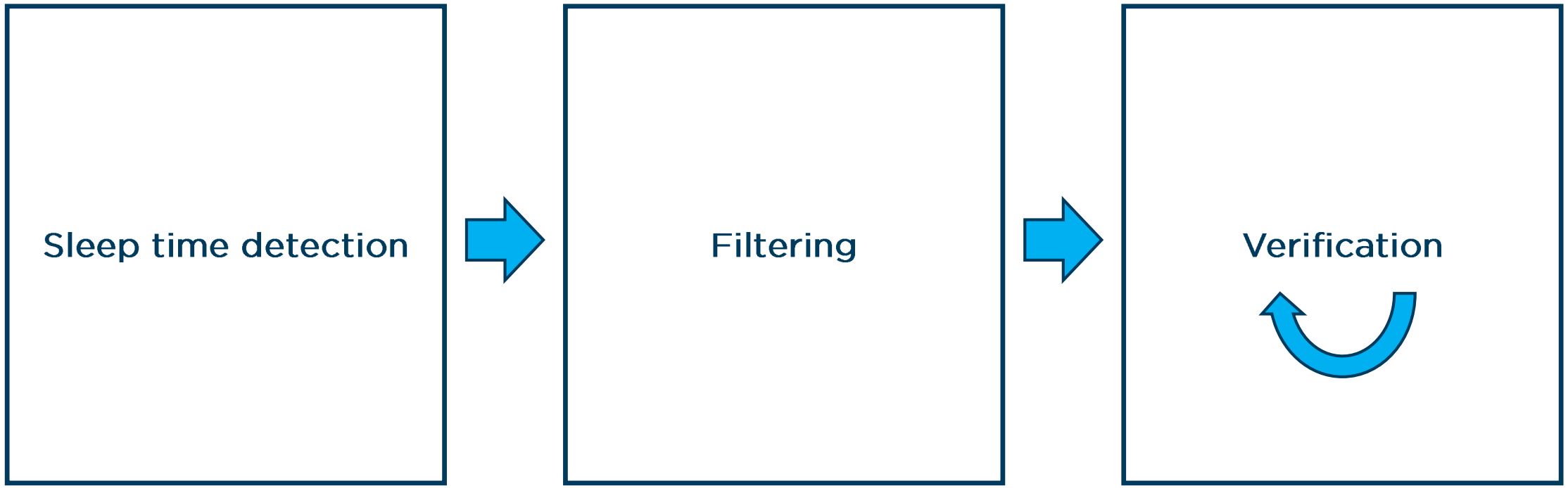
- Attacker can craft a page (guess)
- Wait for a certain amount of time
- Write to page and measure time
- **Write time will reveal if page was shared**

Page content?

How long?

Noise?
Threshold?

CAIN: Cross-VM ASL INtrospection



Suitable pages to break ASLR

Page content?

Page aligned

Mostly static

Read-only
in victim VM

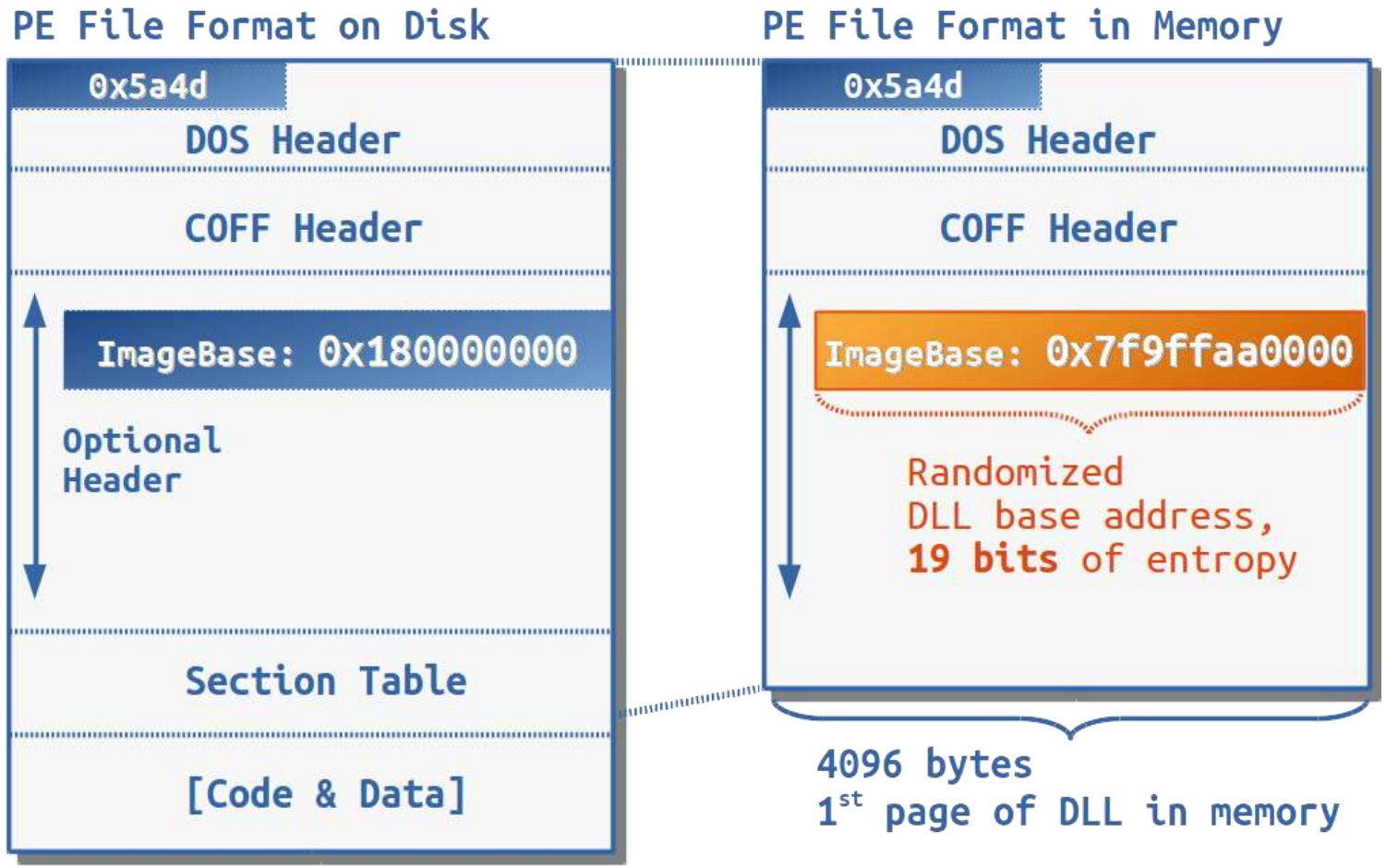
Known to exist

Suitable page
to break ASLR

Contains base address

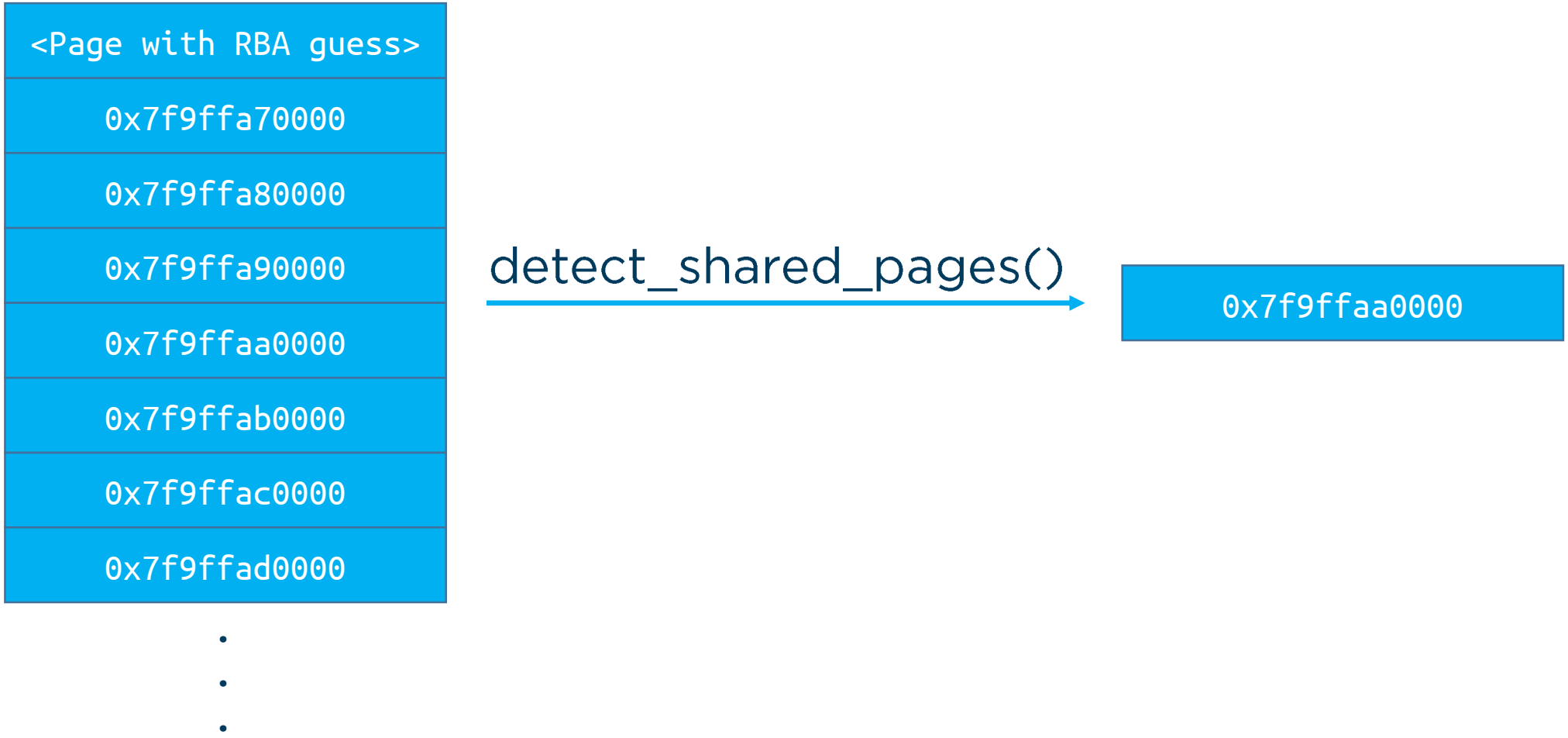
Suitable page under Windows

Page content?

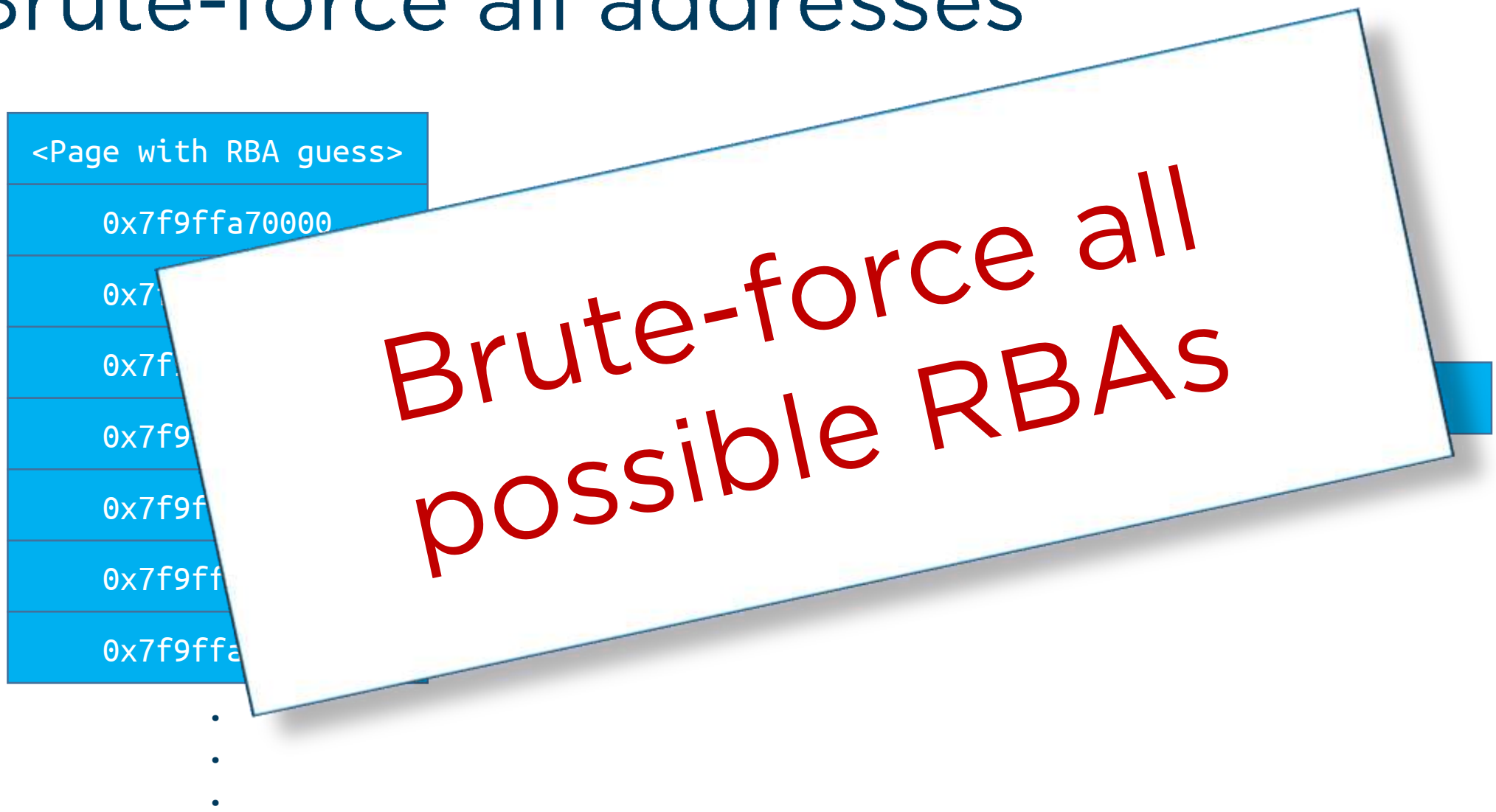


Brute-force all addresses

Page content?



Brute-force all addresses



Wait for how long?

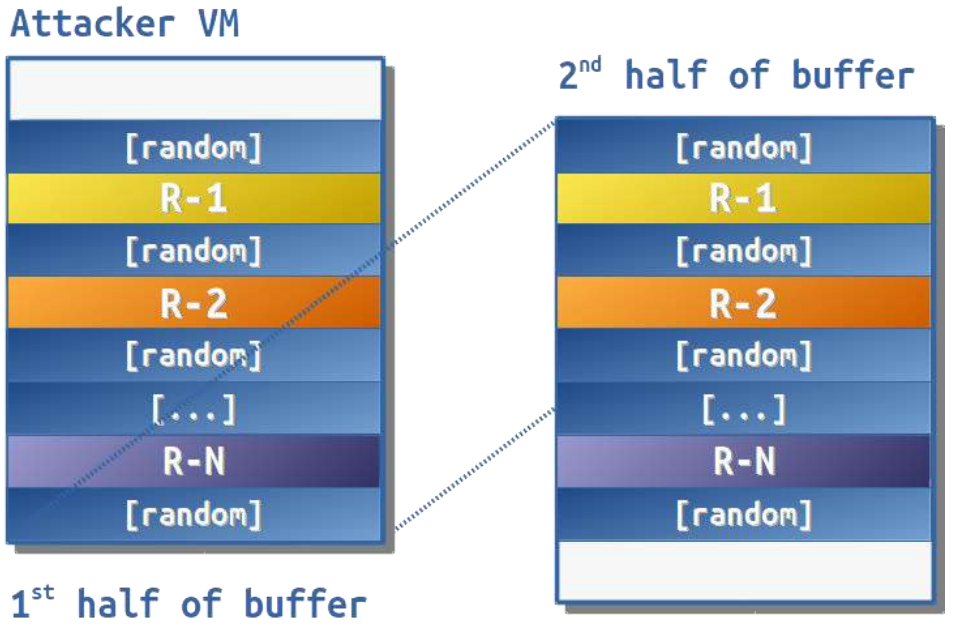
How long?

- Depends on the memory deduplication implementation
- Varies depending on amount of memory used
- Attacker trade-off
  - Waiting too little obstructs the attack
  - Waiting too long increases attack time

Sleep-time detection

How long?

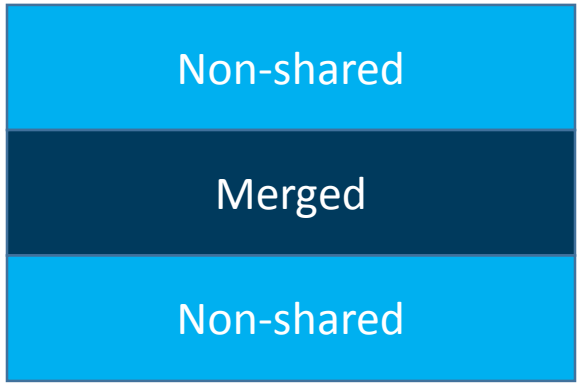
- Create random buffer
- Copy every second page of 1st half to the 2nd half
- Start with test time t_{start}
  - Detect merged pages
  - Iterate and increase test time until detection rate is near 100%



N = 10,000, t_{start} = 10min

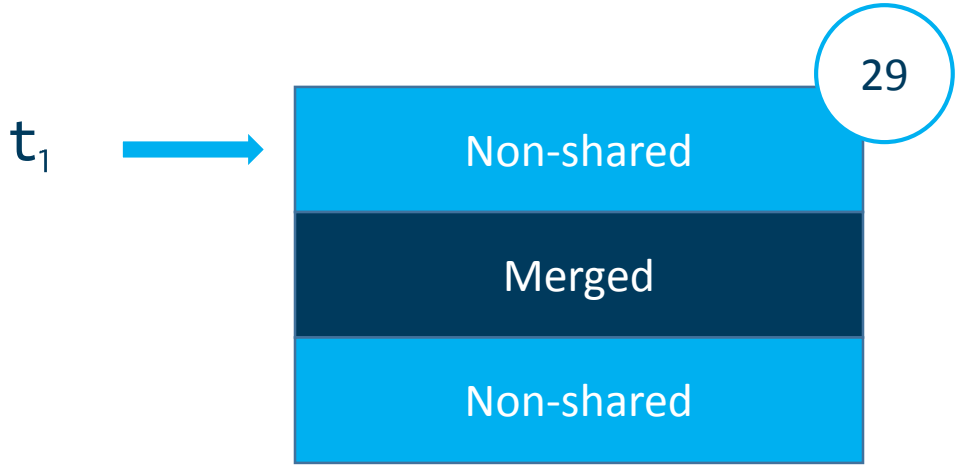
Detect merged pages

Noise?
Threshold?



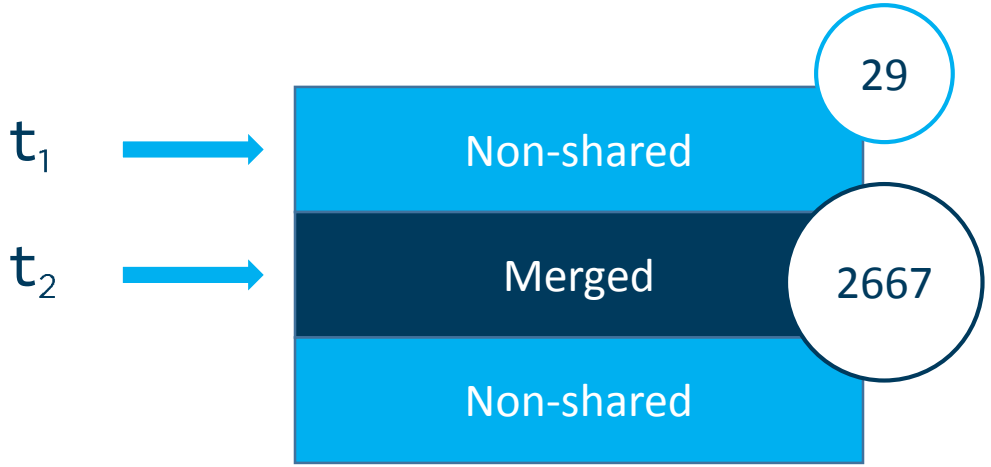
Detect merged pages

Noise?
Threshold?



Detect merged pages

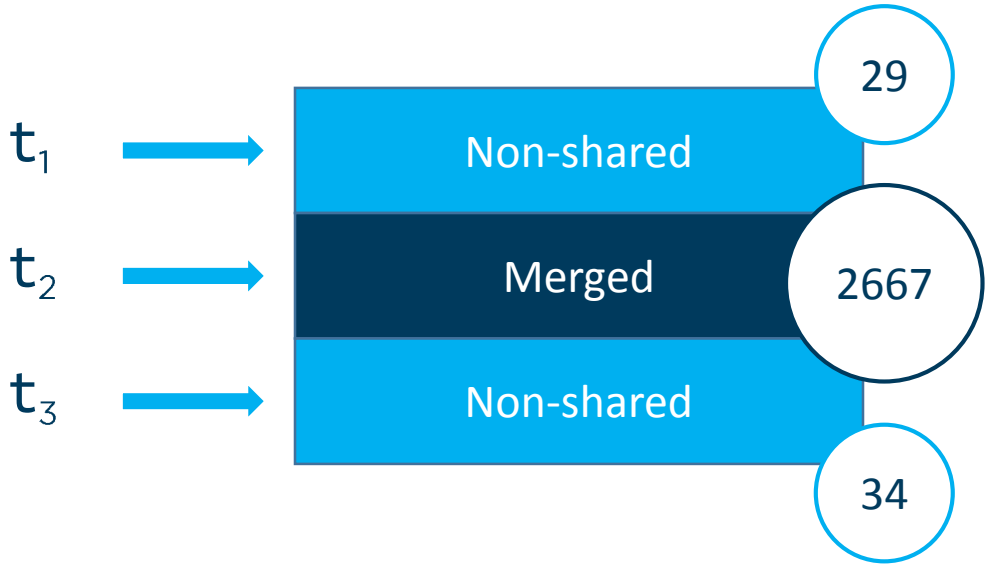
Noise?
Threshold?



Detect merged pages

Noise?
Threshold?

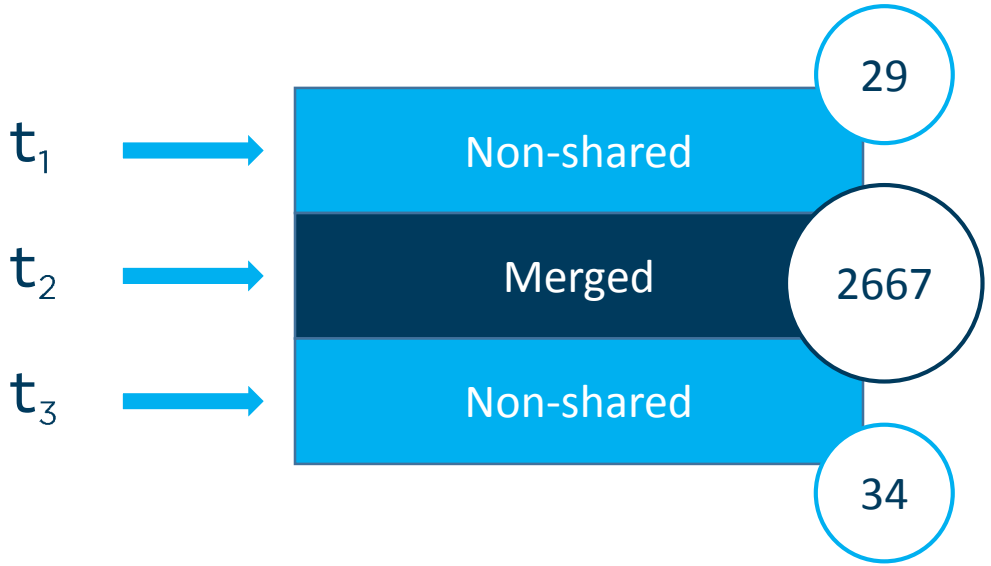
Measure write
time with `rdtsc`
(Read Time Stamp Counter)



Detect merged pages

Noise?
Threshold?

Measure write time with `rdtsc`
(Read Time Stamp Counter)



$t_2 > 2 * (t_1+t_3)/2$

$T_{1,3} < M = 1000$

$t_1 < t_3, (t_3-t_1) < t_3/3$

Detect merged pages

Noise?
Threshold?

Mea
tim
(Rea

These heuristics worked for different HW configurations

$t_2 > 2 * (t$

$T_{1,3} < M = 1000$

$t_1 < t_3, (t_3-t_1) < t_3/3$

Handling noise

Noise?
Threshold?

- Noise can affect write time
  - High write time for non-shared page
  - Miss shared page because of increased write time of adjacent pages

False positive

False negative

- Perform **several rounds of detection**
  - Noise will cancel out
  - Eliminate candidates

Handling noise

Noise?
Threshold?

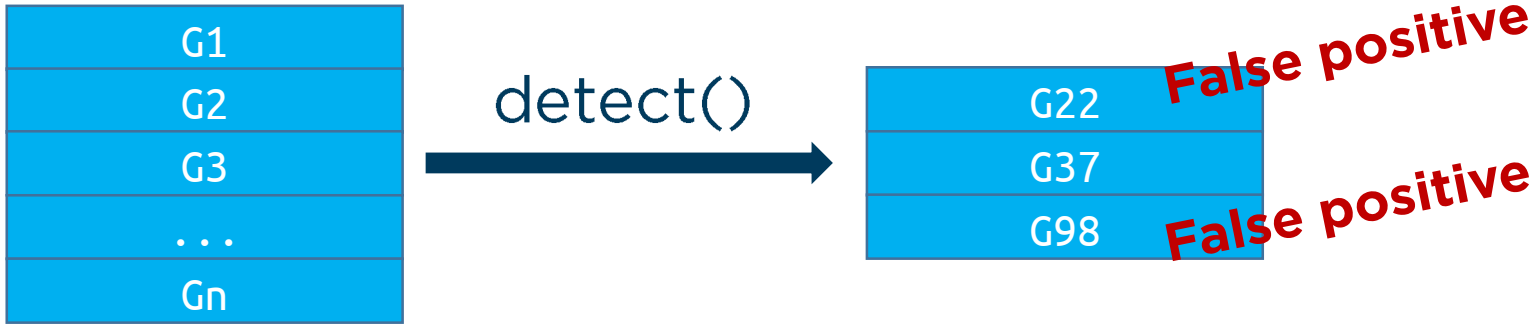
Round 1



Handling noise

Noise?
Threshold?

Round 1

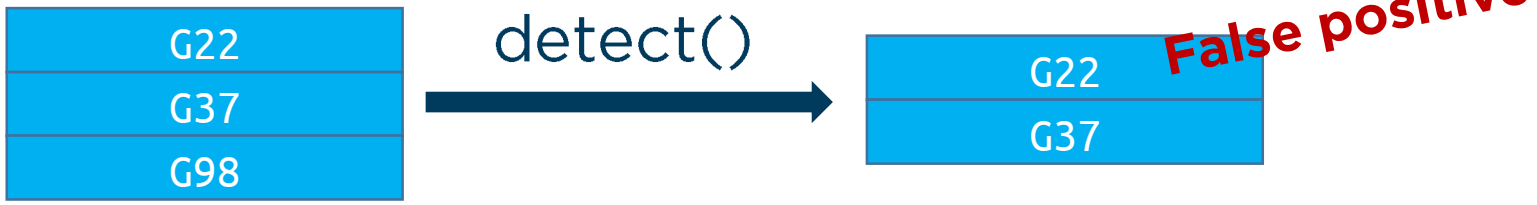


Handling noise

Noise?
Threshold?

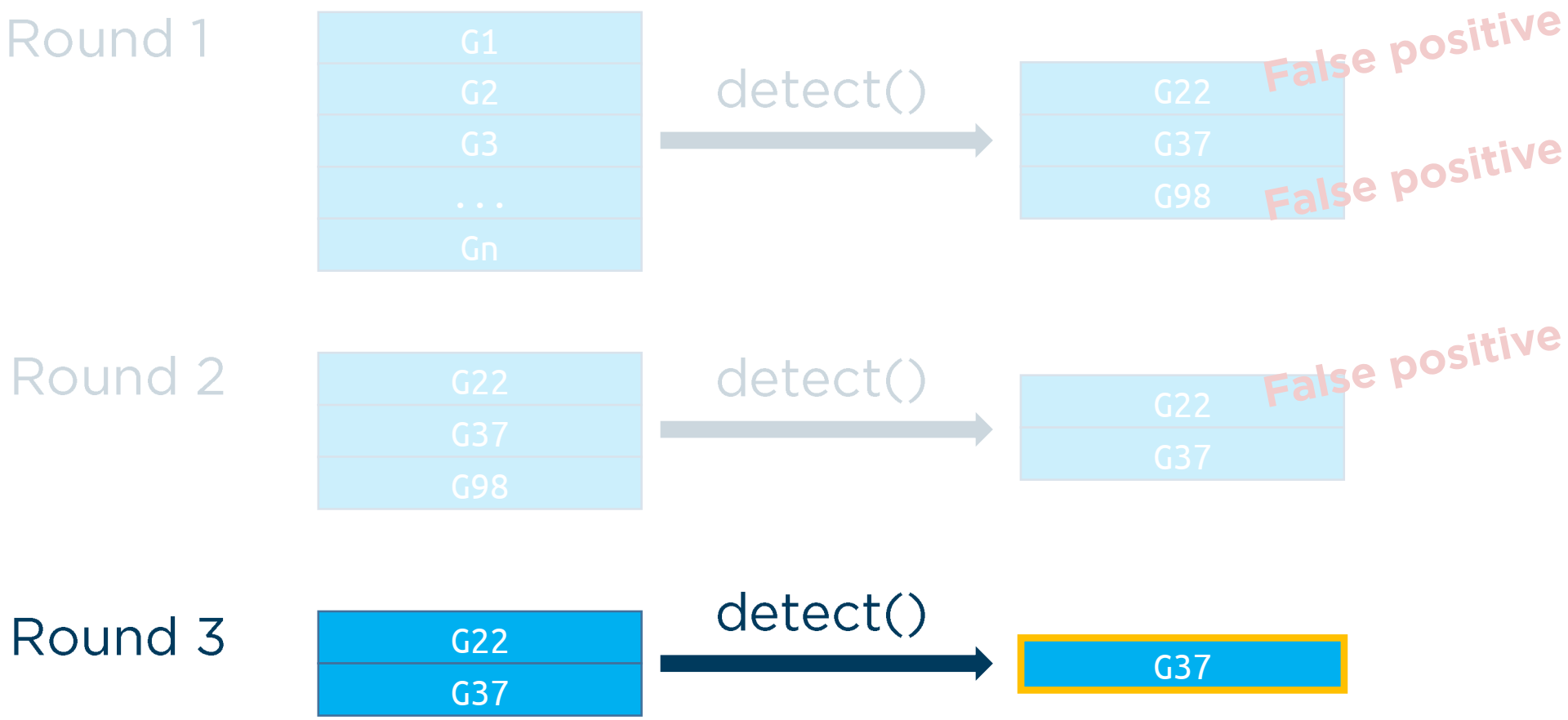
Round 1

Round 2



Handling noise

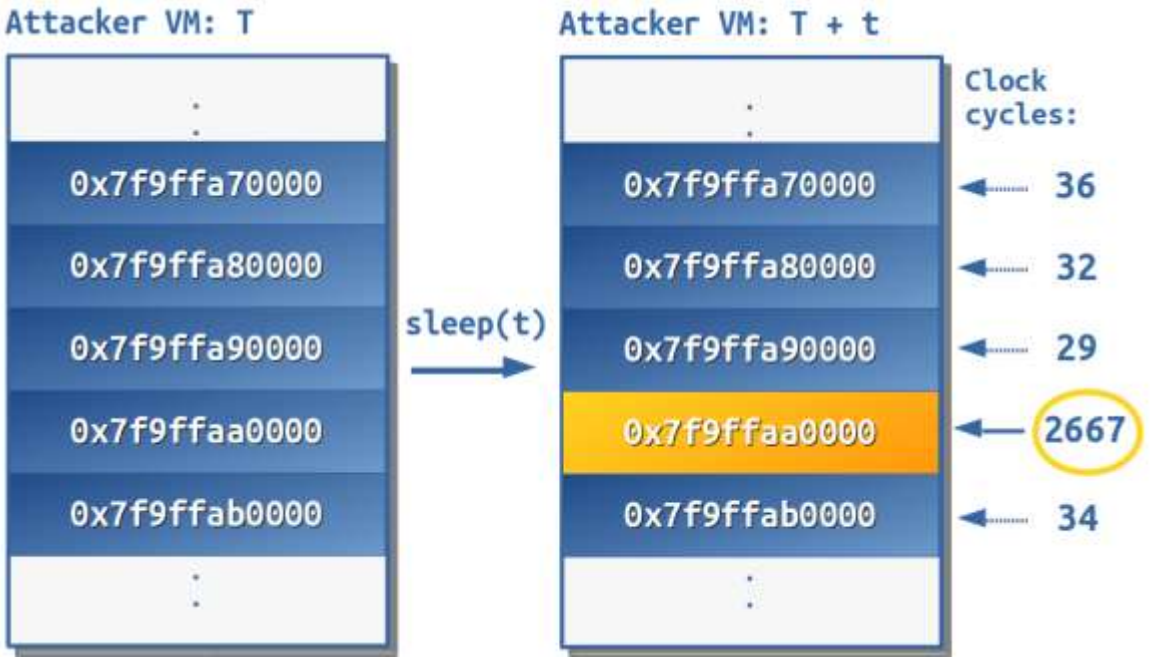
Noise?
Threshold?



Filtering

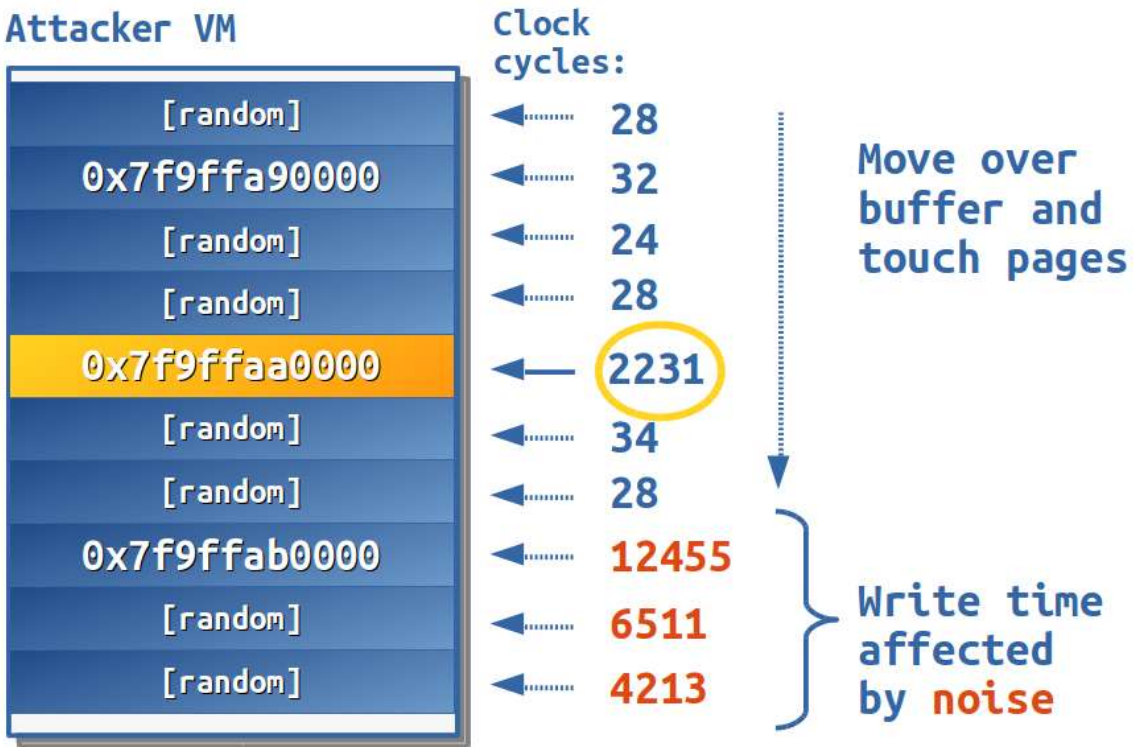
Noise?
Threshold?

- As little memory per candidate as possible
- Once at the beginning to eliminate as many candidates as possible



Verification

- Verify remaining potential candidates
- Use more memory per candidate
- Eliminate remaining false positives



CAIN attack phases



Evaluation

Evaluation setup

- Dual CPU Blade Server
  - 2 x AMD Opteron 6272 CPUs with 16 cores each
  - 32GB of RAM
  - VMM: KVM on Ubuntu Server 14.04.2 LTS x86_64
  - Linux Kernel 3.16.0

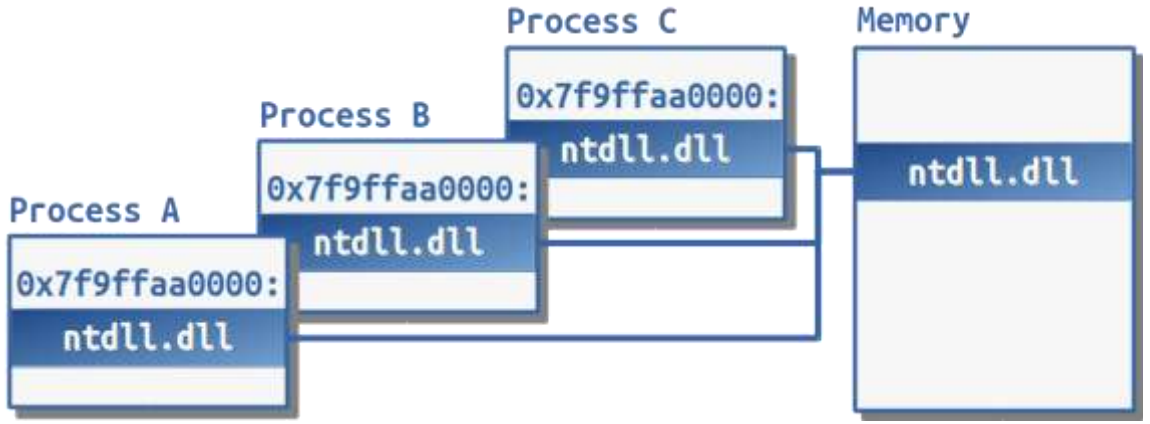
- 1 attacker VM with Ubuntu Linux 14.04
- 7 victim VMs with Windows Server 2012 (6.2.9200 Build 9200)
  - 4 vCPUs, 4 GB per VM

ASLR in Windows x64

- High Entropy ASLR

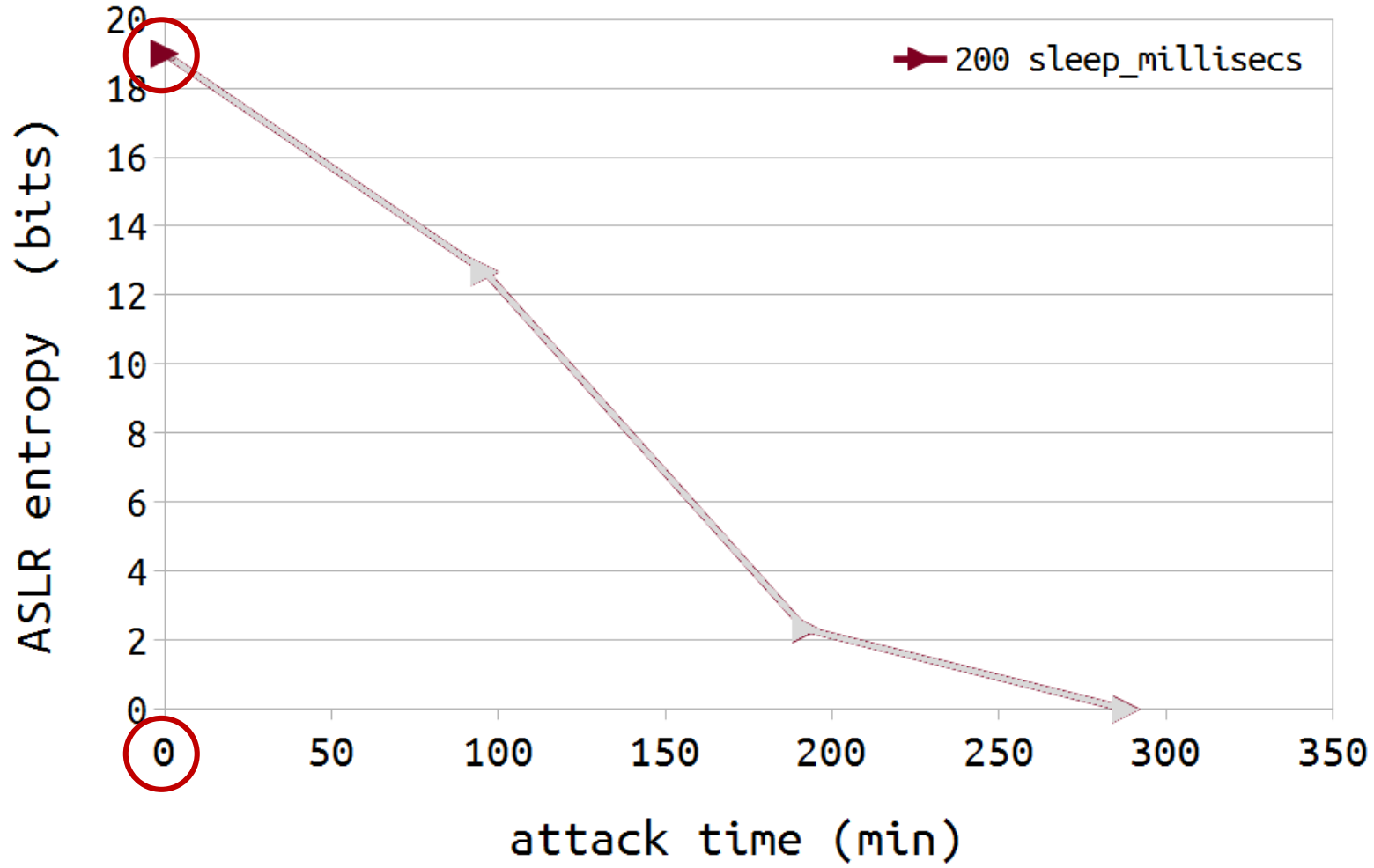
- 33 bits for stacks
- 24 bits for heaps

- 17 bits for executables
- 19 bits for DLLS



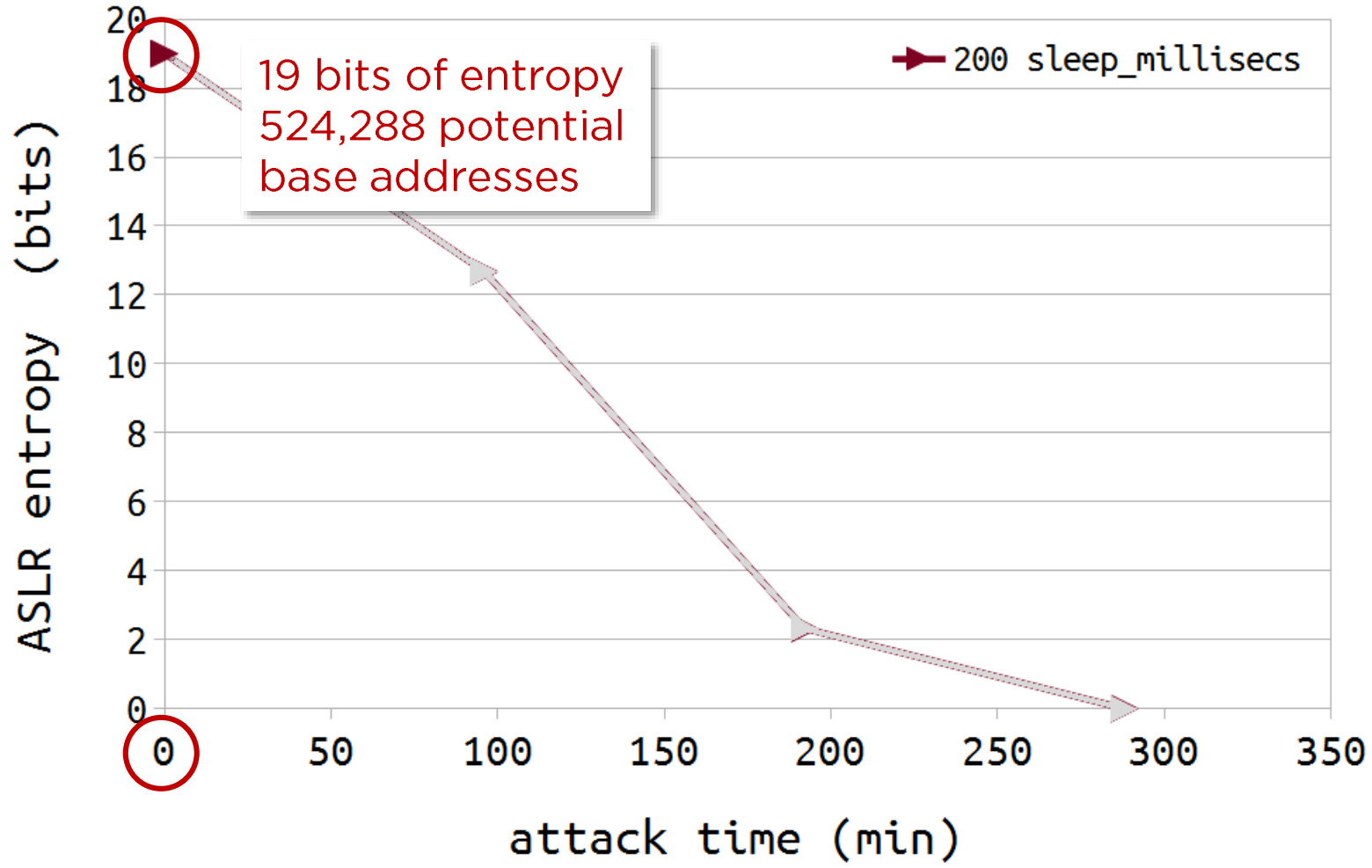
System-wide
at boot-time

Attacking a single Windows VM



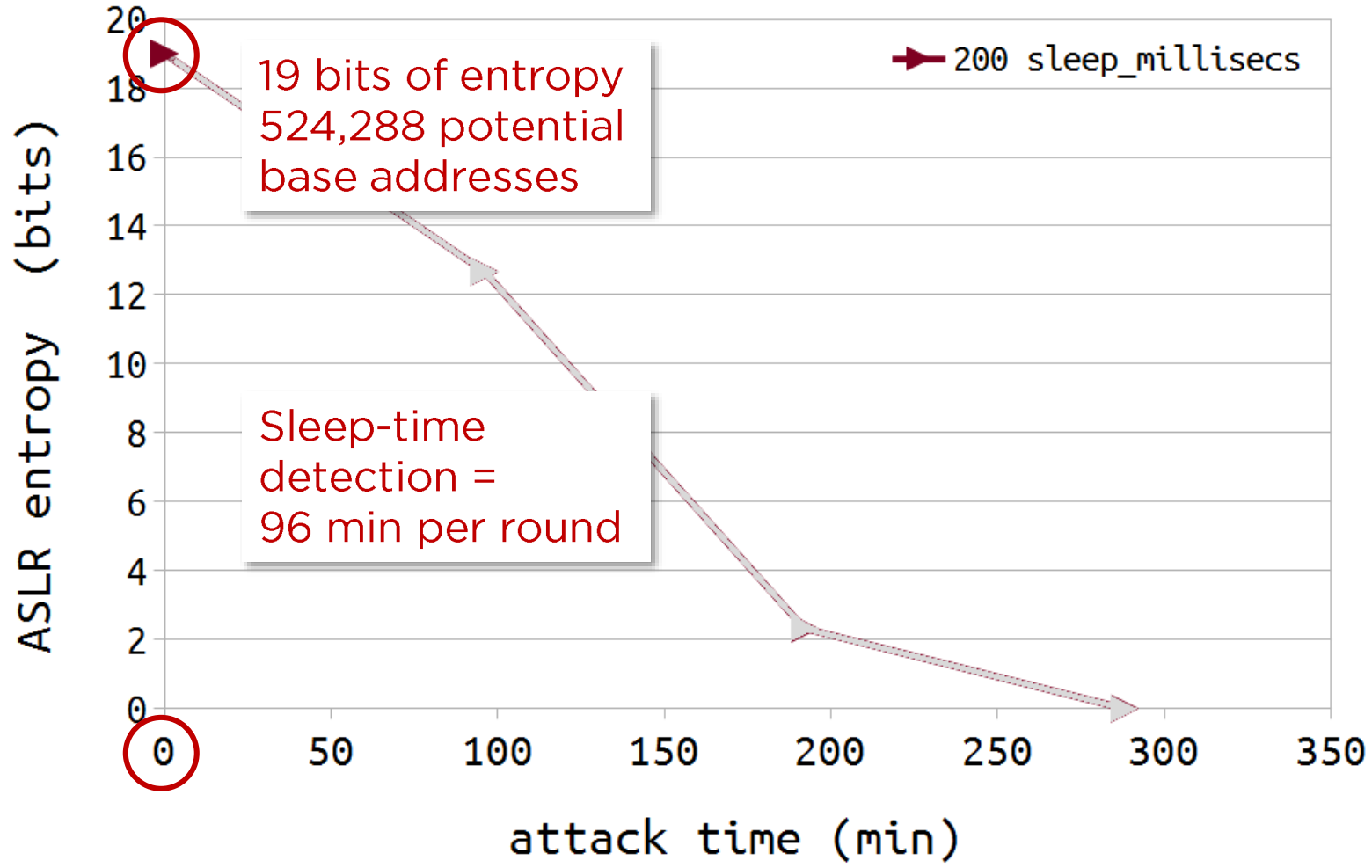
time t = 0

Attacking a single Windows VM



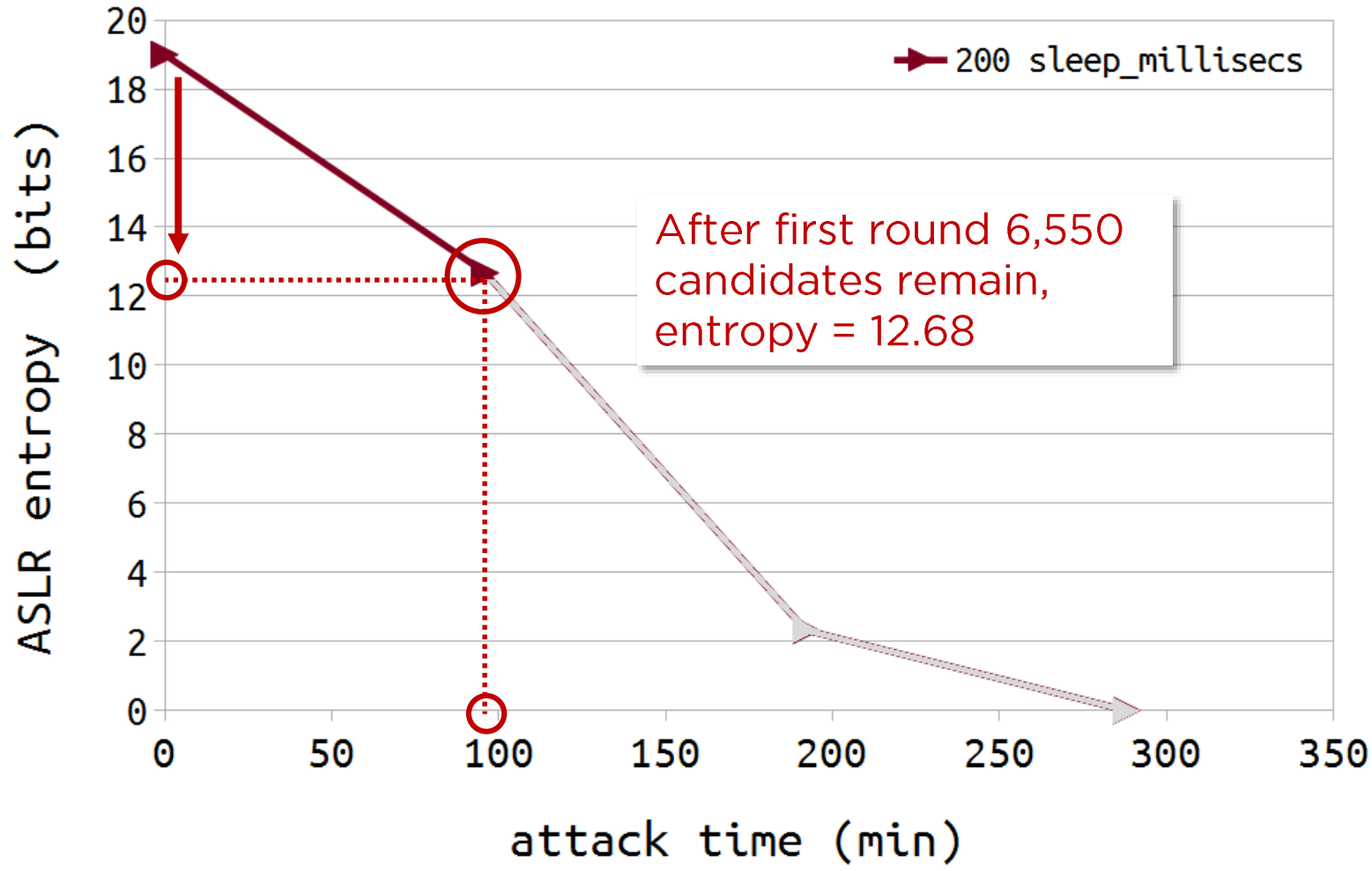
time t = 0

Attacking a single Windows VM



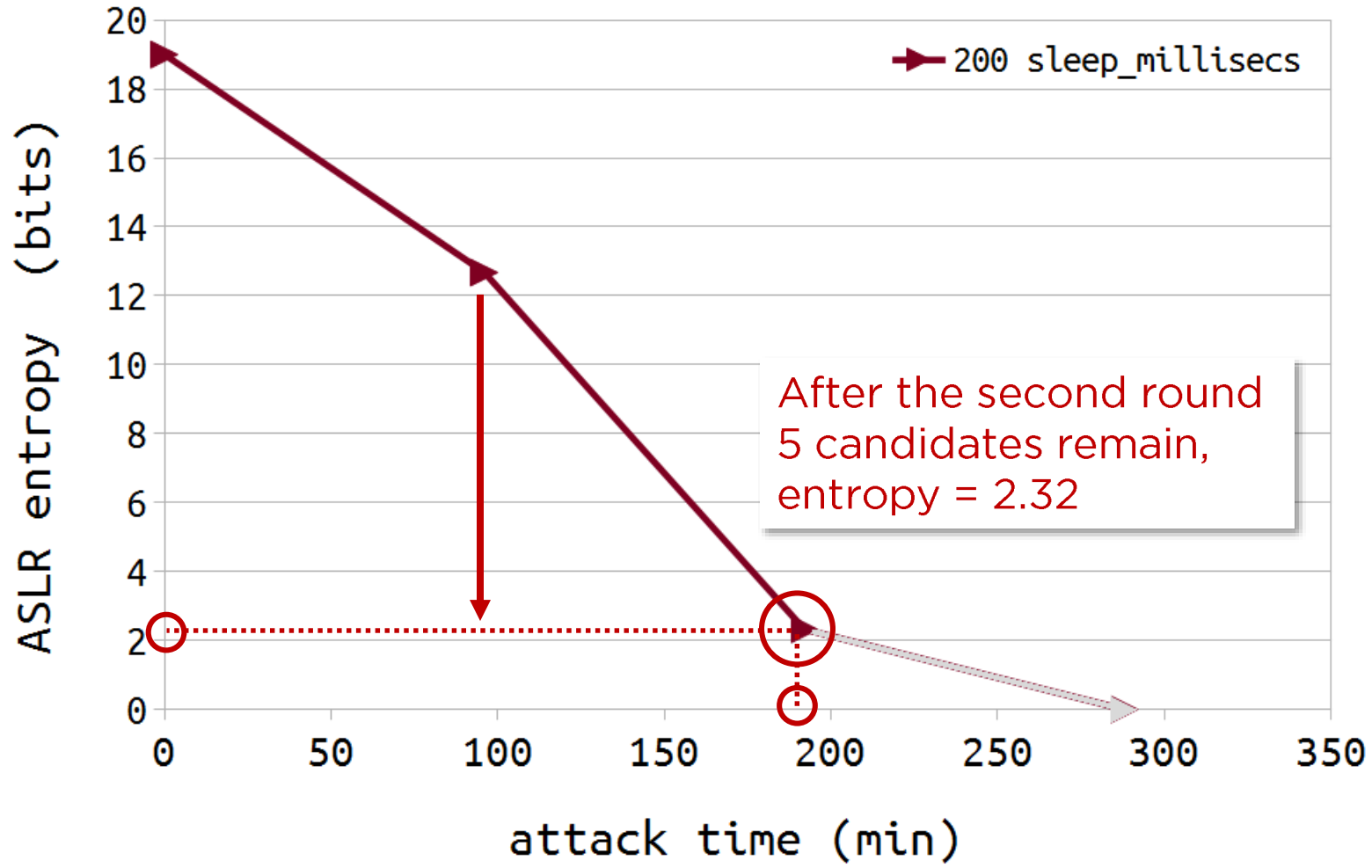
time t = 0

Attacking a single Windows VM



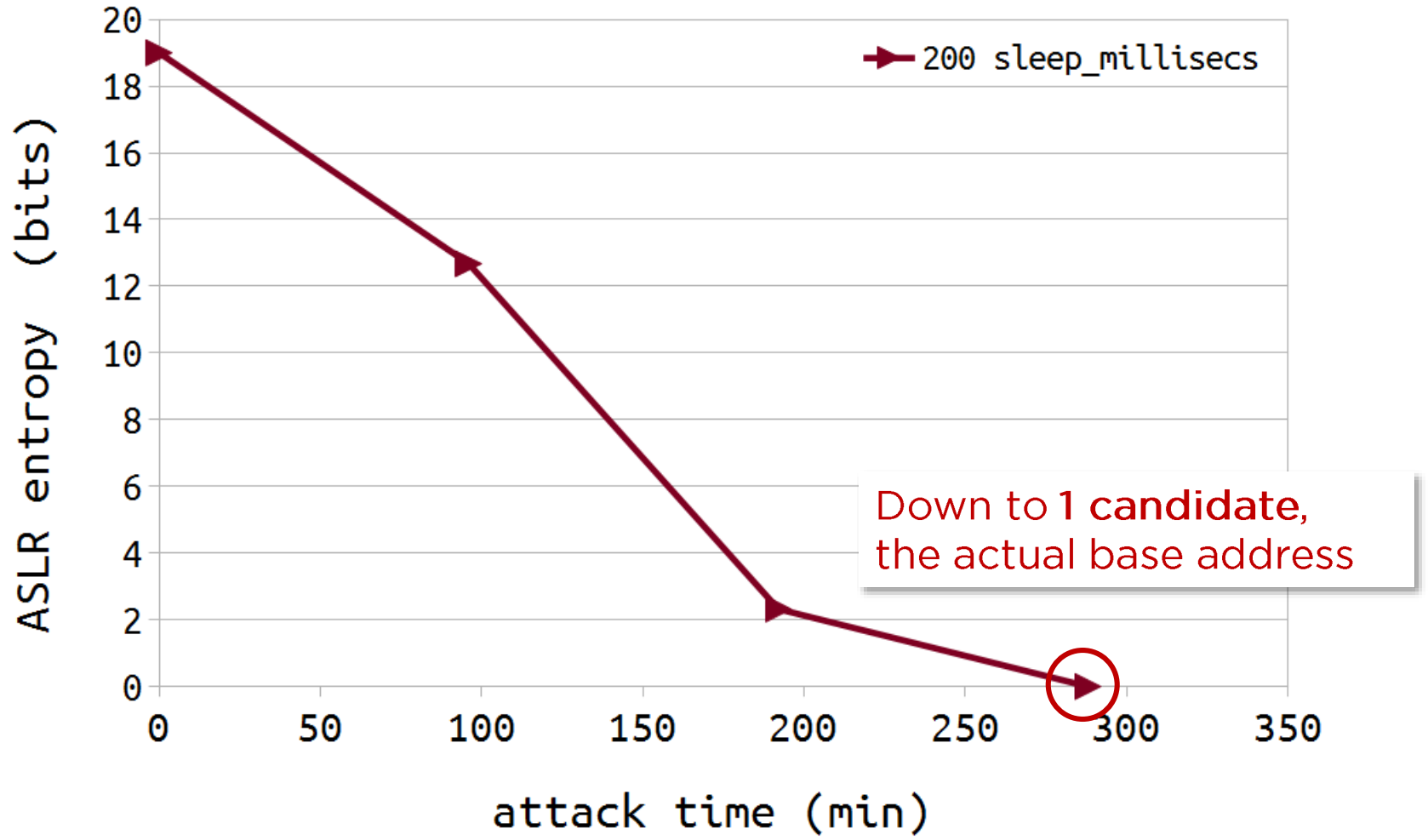
time t = 96

Attacking a single Windows VM



time t = 192

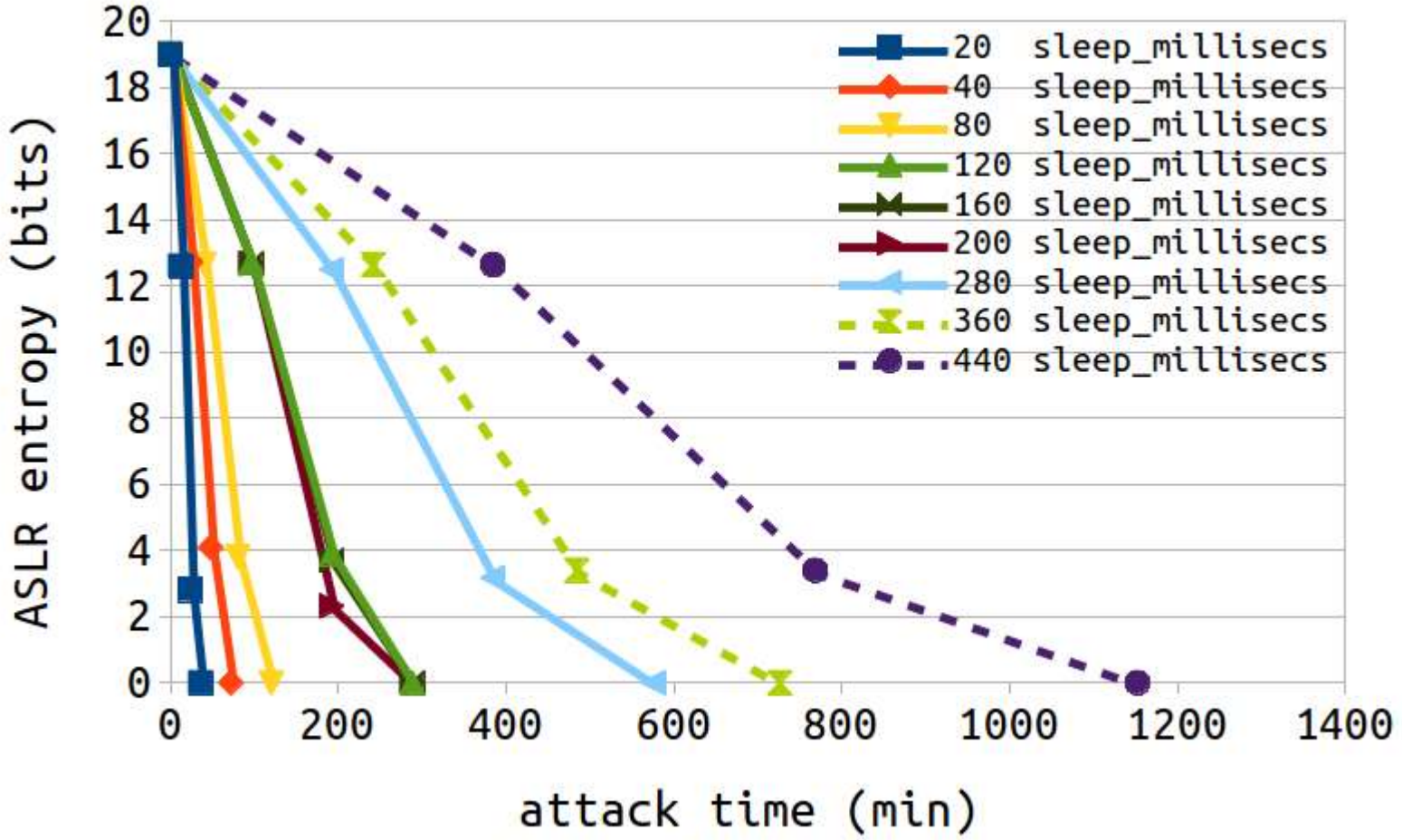
Attacking a single Windows VM



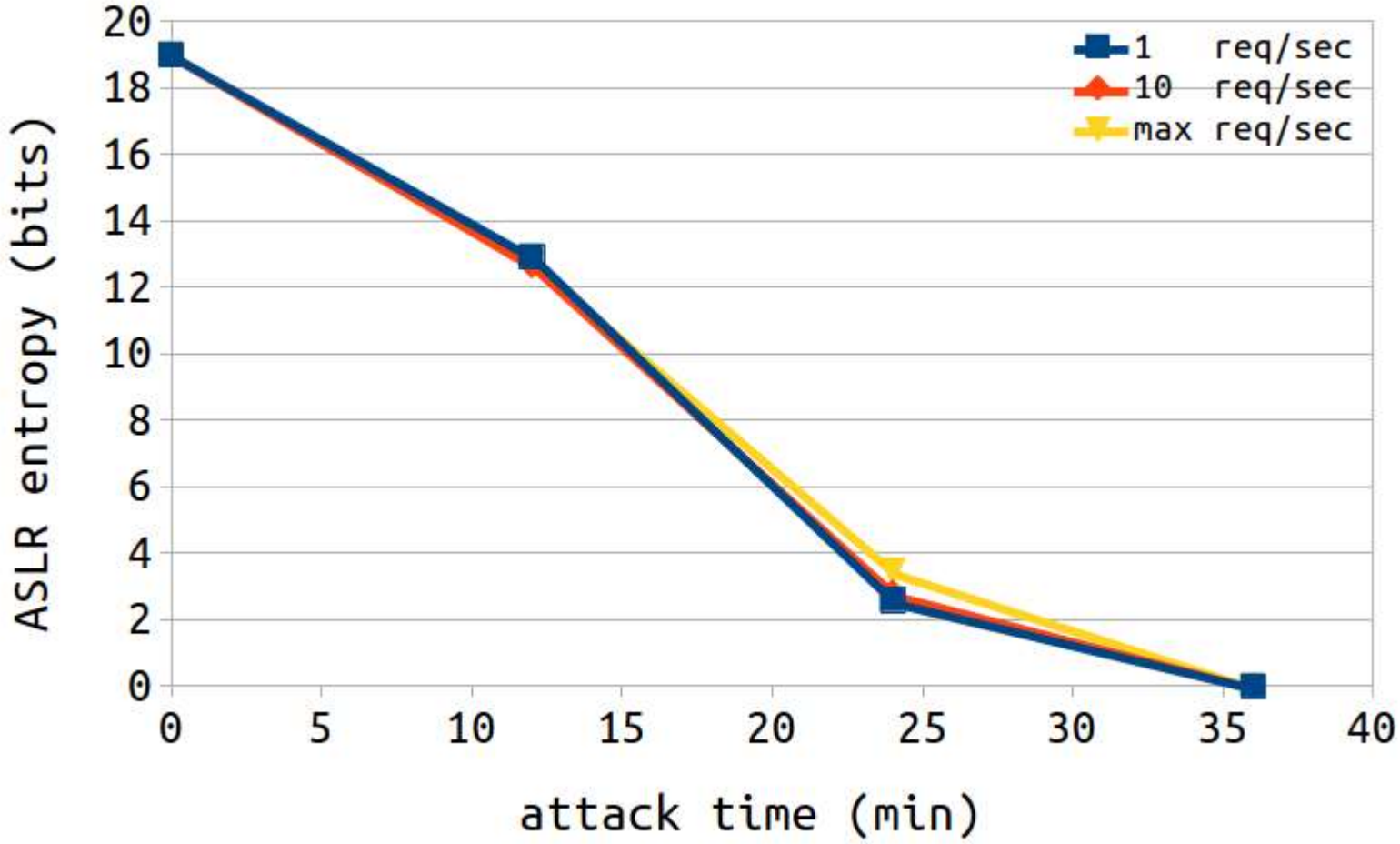
time t = 288

Attacking a single Windows VM

and various merge times



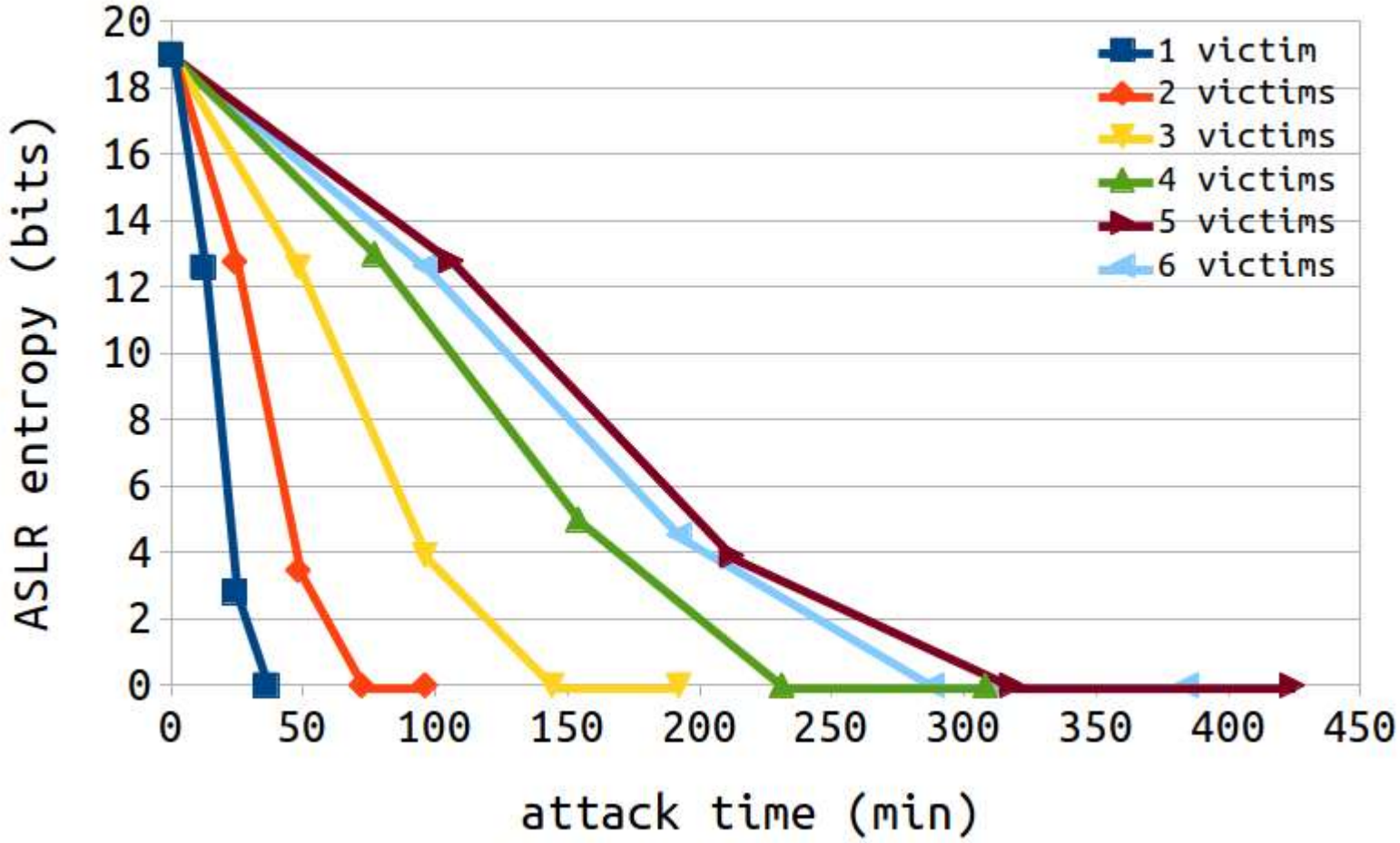
Attacking a Windows VM under load



Victim VM runs IIS webserver, load generated with a separate physical machine and AB (apache benchmark)

sleep_millisecs = 20

Attacking multiple Windows VMs



sleep_millisecs = 20

Post-CAIN exploitation

- De-randomized base address can now be used in code-reuse attack
- Against a single victim, use the inferred value
- Against multiple victims, try all base addresses
  - Or link base addresses to specific VMs or processes
  - E.g., through Import Address Table (Windows) or .got.plt (Linux), or any other suitable page

Mitigations

- VMM layer: **Deactivation of memory deduplication**
- System layer: **Attack detection**
- ASLR layer: **Increase ASLR entropy**
- Process layer: **More entropy in sensitive memory pages**

Conclusion

- CAIN effectively leaks RBAs through VMMs with same content page-based memory deduplication
  - Impact: ASLR no longer effective as defense!
  - Attacking Windows < 5 hrs
  - Higher entropy ASLR > days (e.g., Linux)

- Reliable and automated PoC exploit
- Real-world cloud providers affected
- Vendors notified on June 4, 2015 (**CVE-2015-2877**)

Thanks!

antonio.barresi@inf.ethz.ch