Silently Breaking ASLR In The Cloud

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

black hat EUROPE 2015

Amsterdam, November 12, 2015

xorlab VU Vrije Universiteit Amsterdam Purdue University ETH zürich

Who we are

Antonio Barresi
antonio.barresi@xorlab.com
@AntonioHBarresi

Co-founder of

xorlab

Interested in software and systems security topics.

http://www.antoniobarresi.com

https://www.xorlab.com

Dr. Kaveh Razavi
kaveh@cs.vu.nl

Researcher at

VU VRIJE UNIVERSITEIT AMSTERDAM

Research on building reliable and secure computing systems.

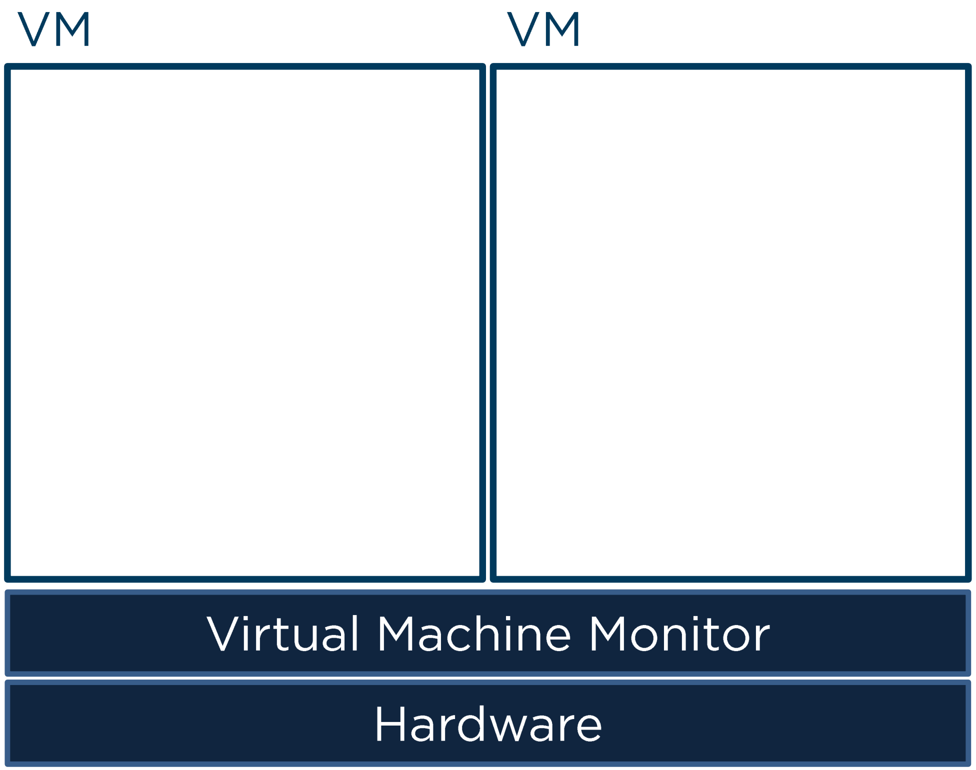
http://www.cs.vu.nl/~kaveh/

http://www.cs.vu.nl

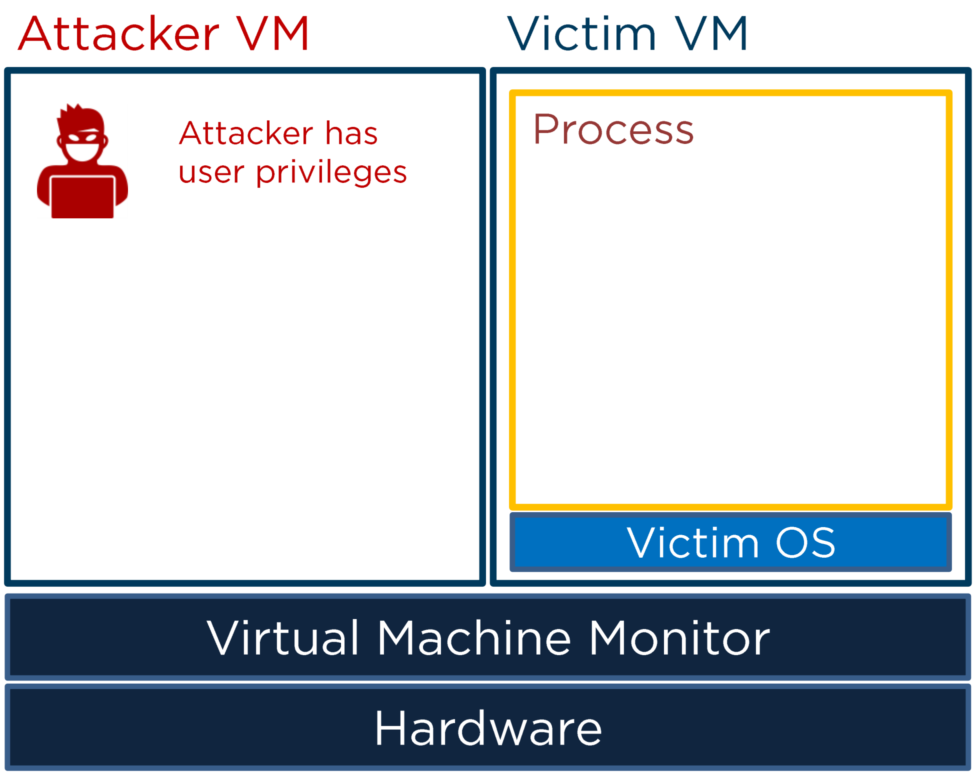
Agenda

- > Threat scenario
- > Memory deduplication
- > Side-channel
- > CAIN attack (Cross-VM Address Space Layout INtrospection)
- > Evaluation
- > Post-CAIN exploitation
- > Mitigations

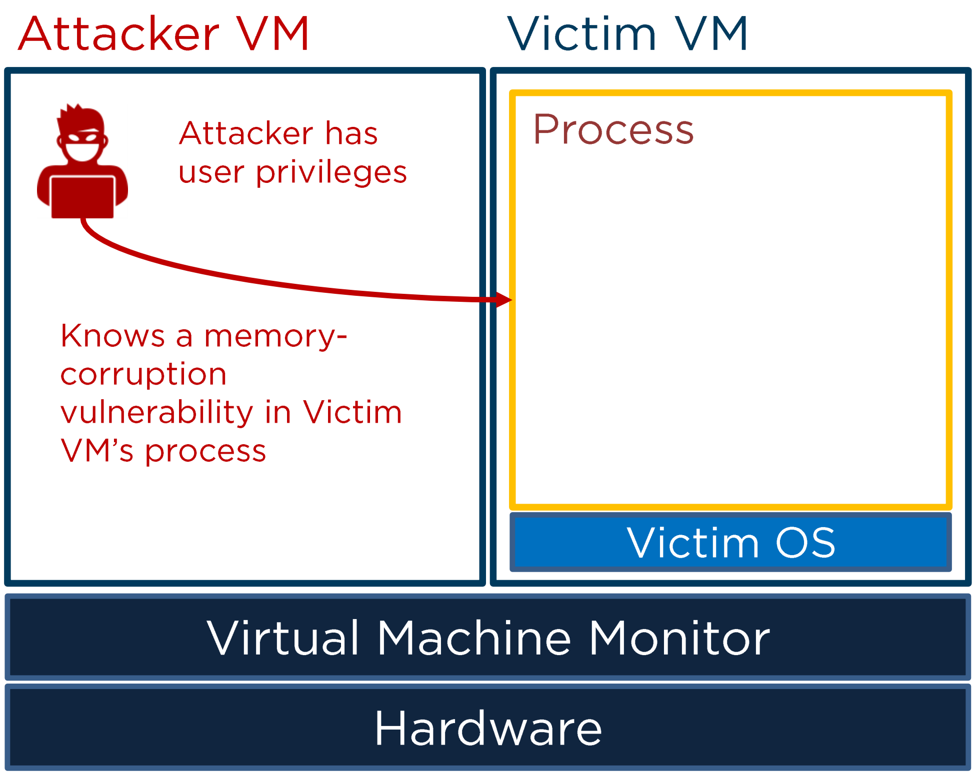
Threat scenario



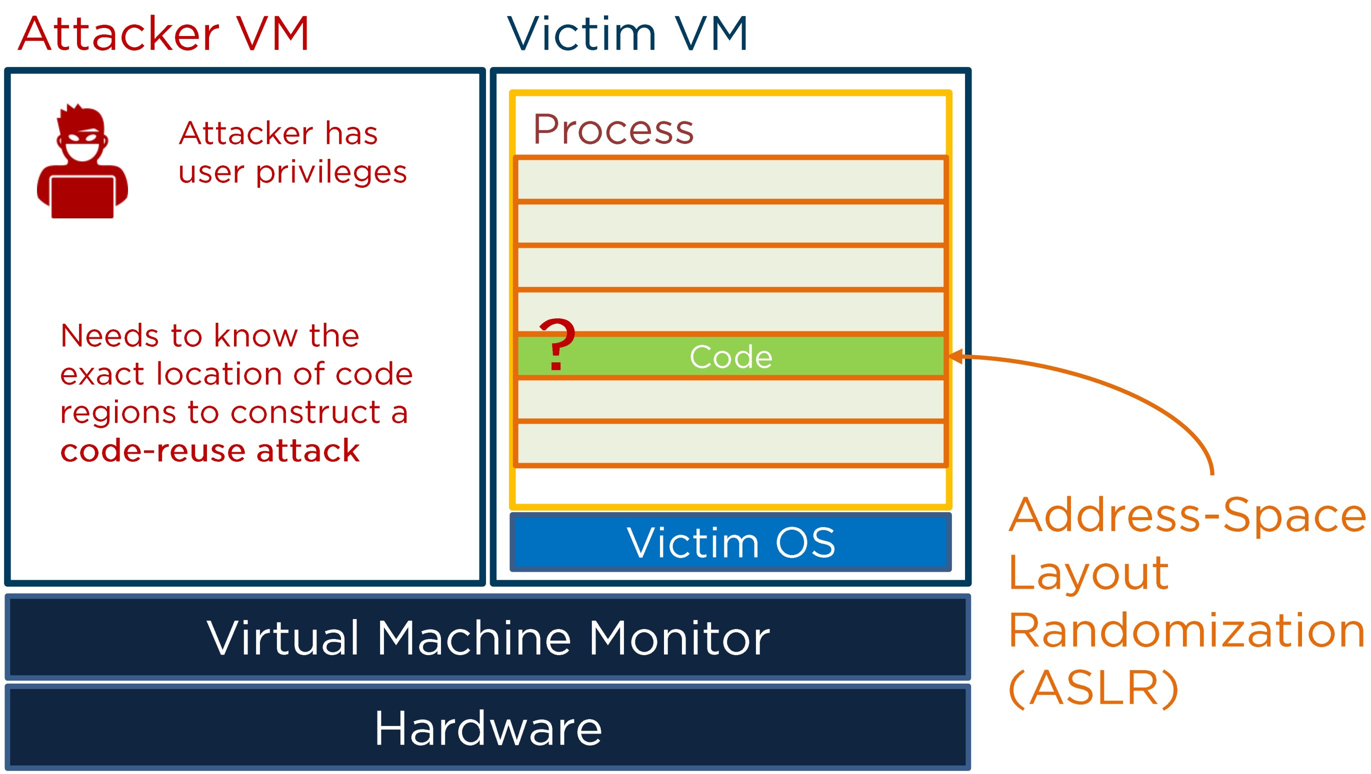
Threat scenario



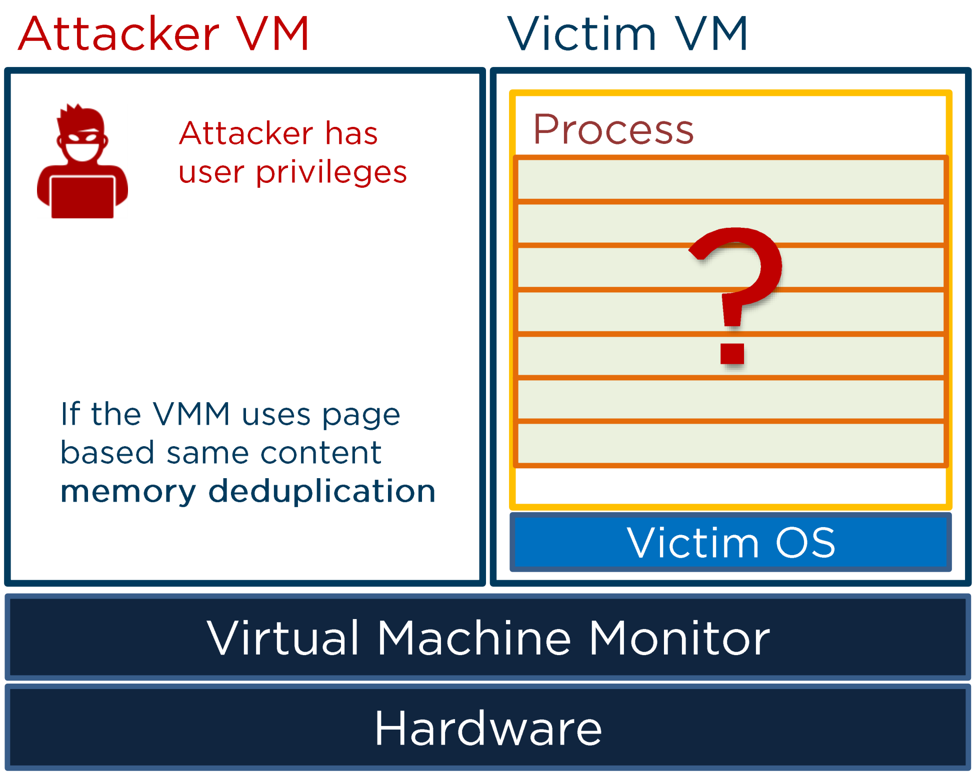
Threat scenario



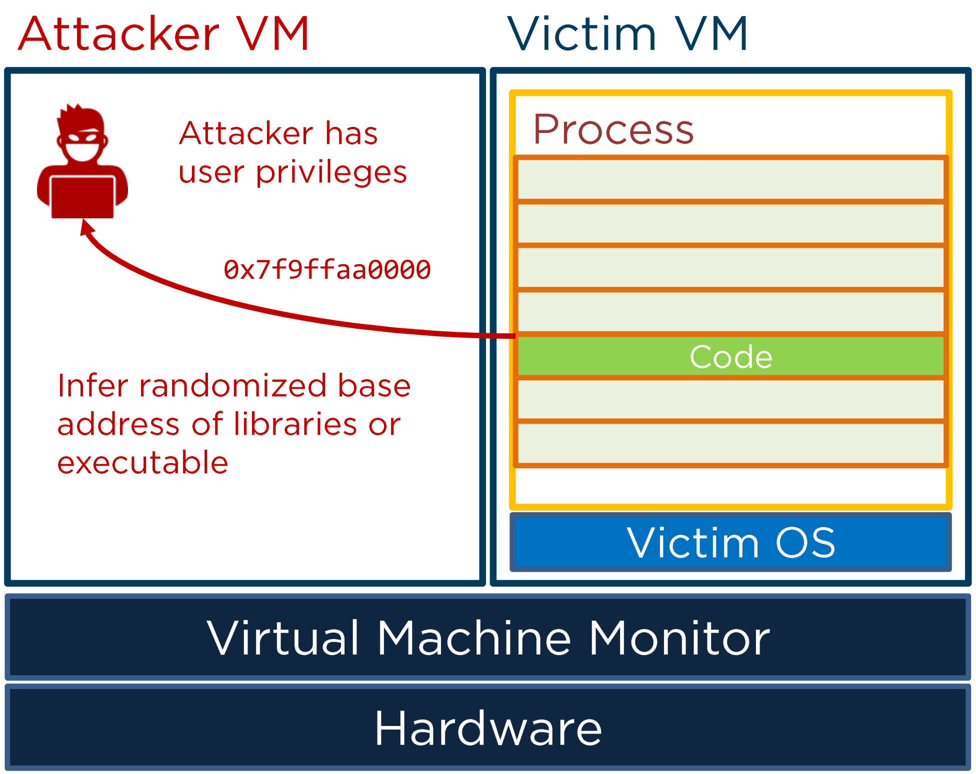
Threat scenario



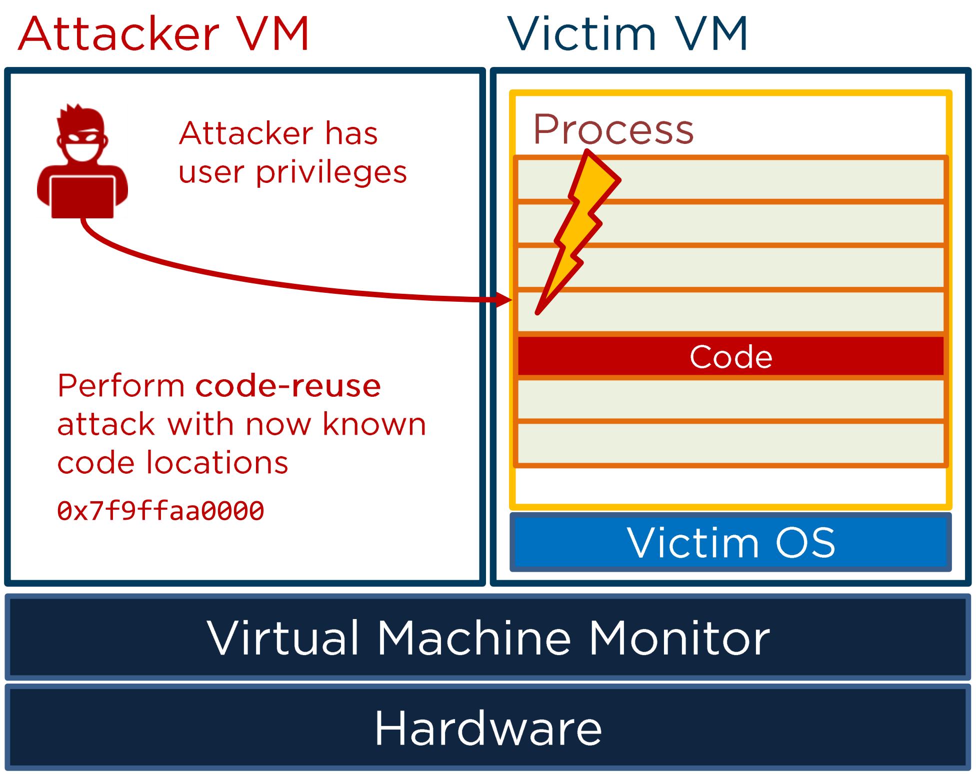
Threat scenario



Threat scenario



Threat scenario

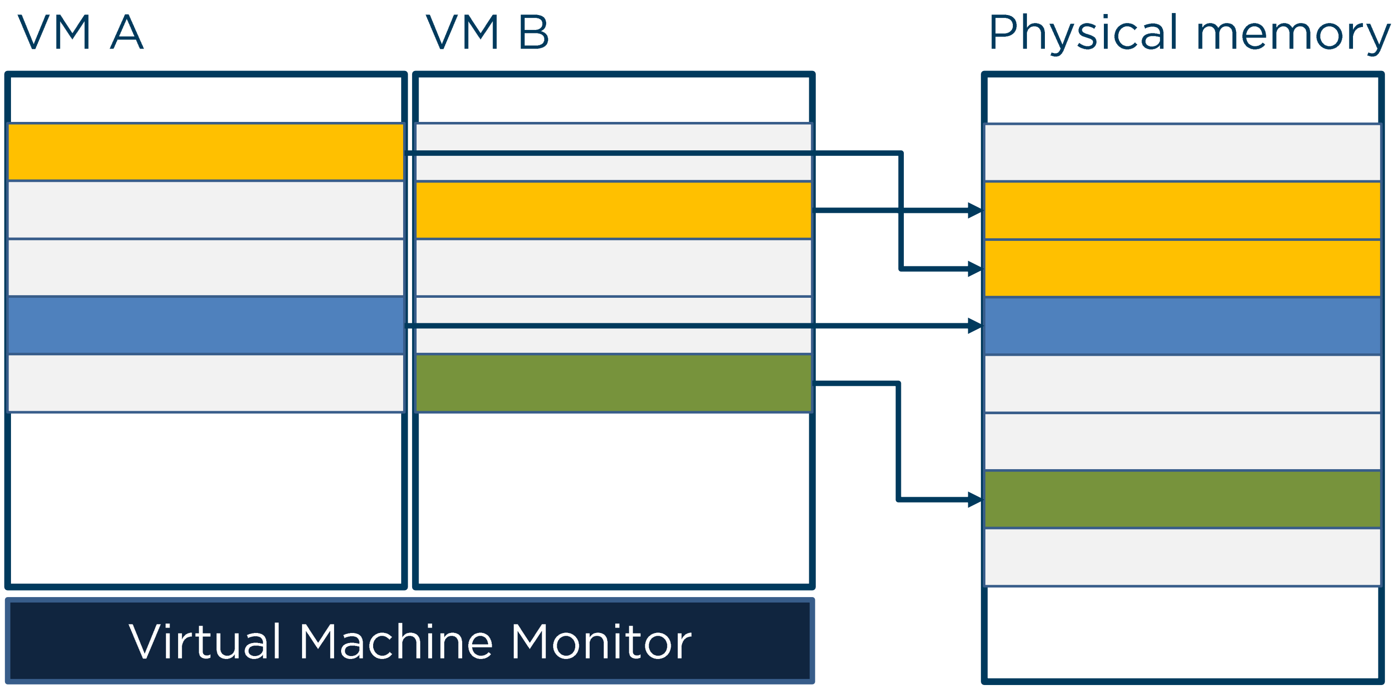


CAIN: Cross-VM ASL INtrospection

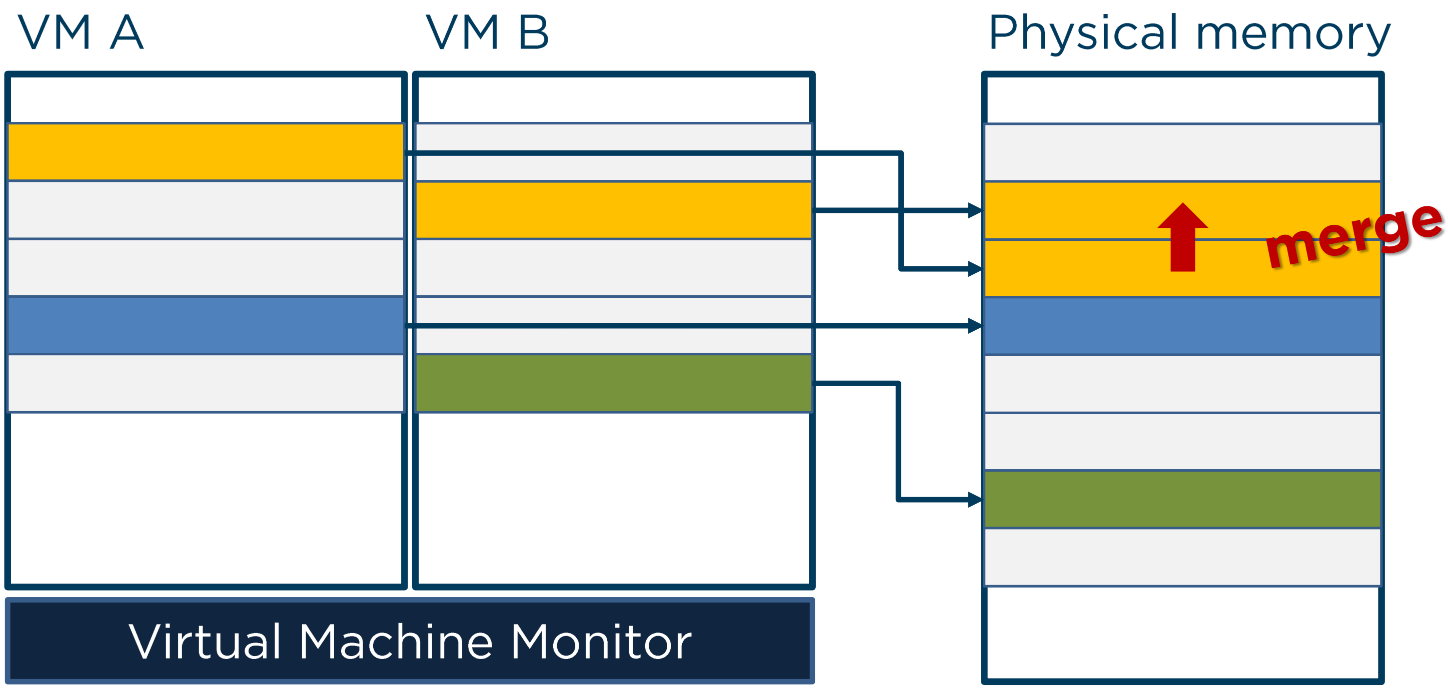
- New attack vector against memory deduplication
  - CVE-2015-2877
  - VU#935424 (https://www.kb.cert.org/vuls/id/935424)

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

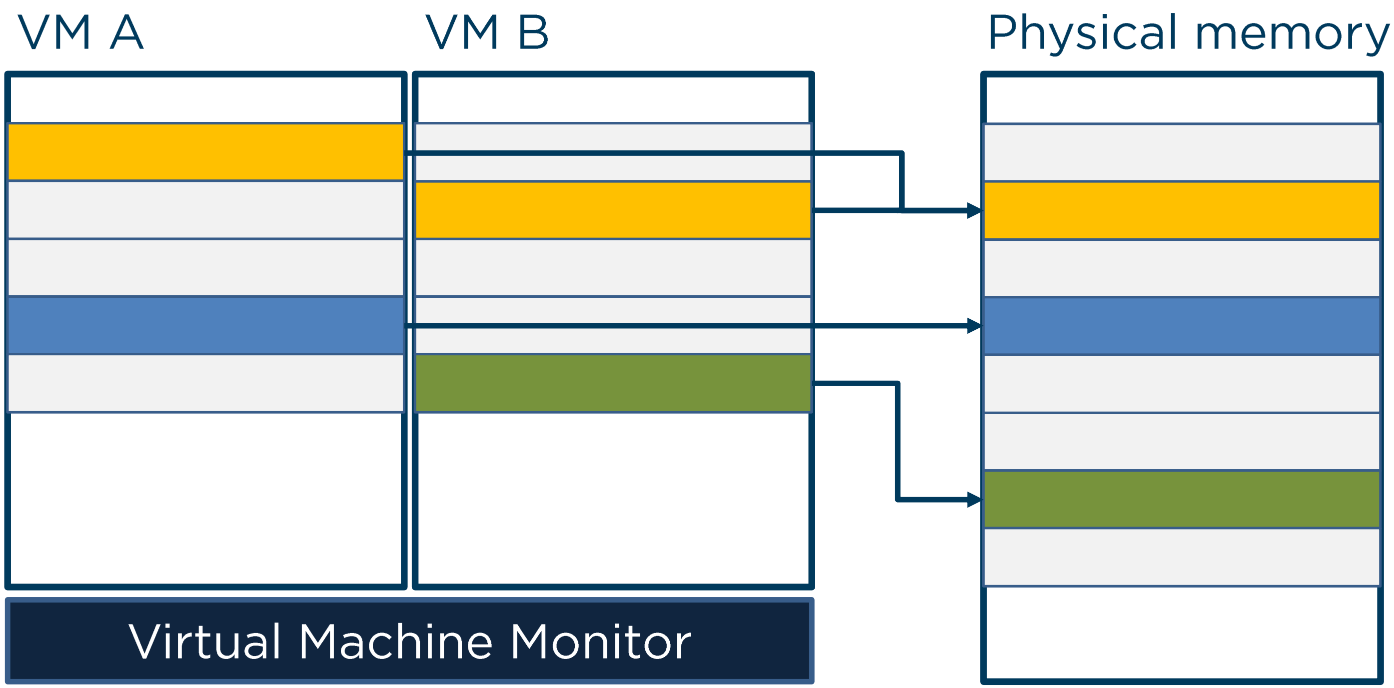
Memory deduplication



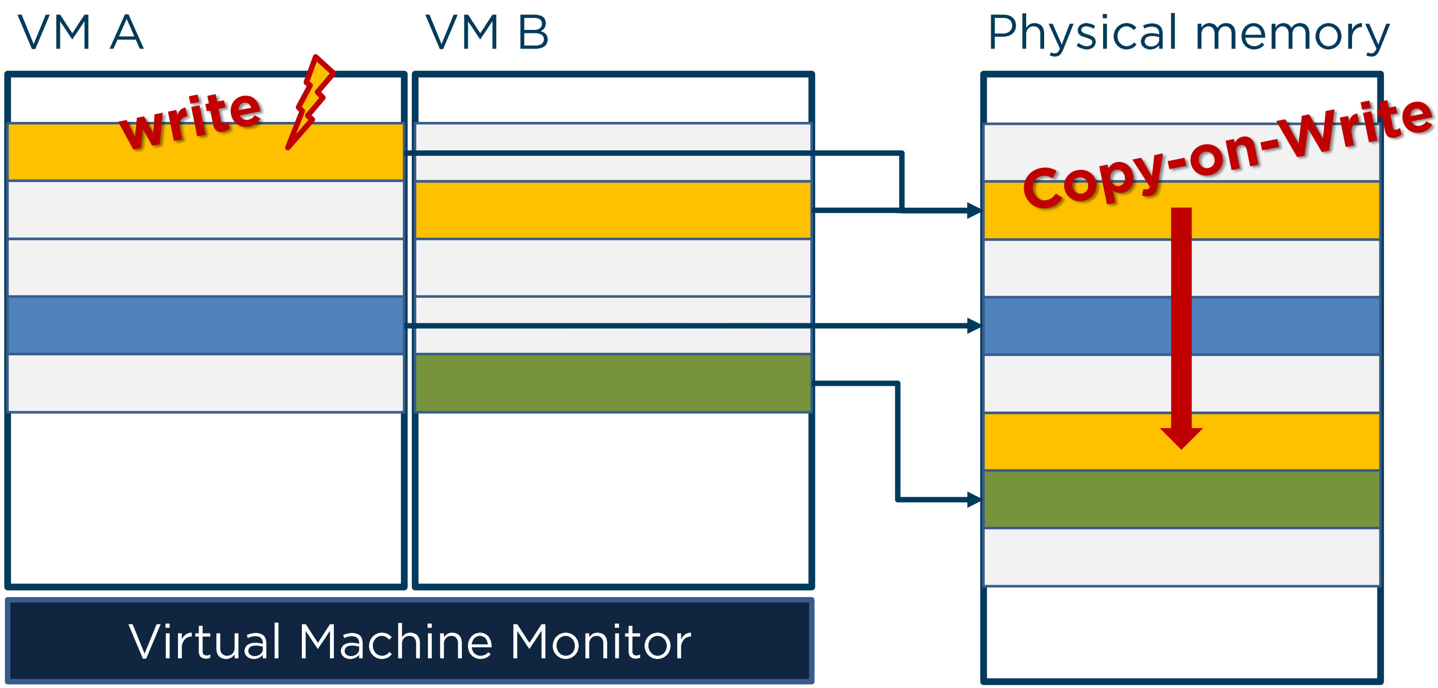
Memory deduplication



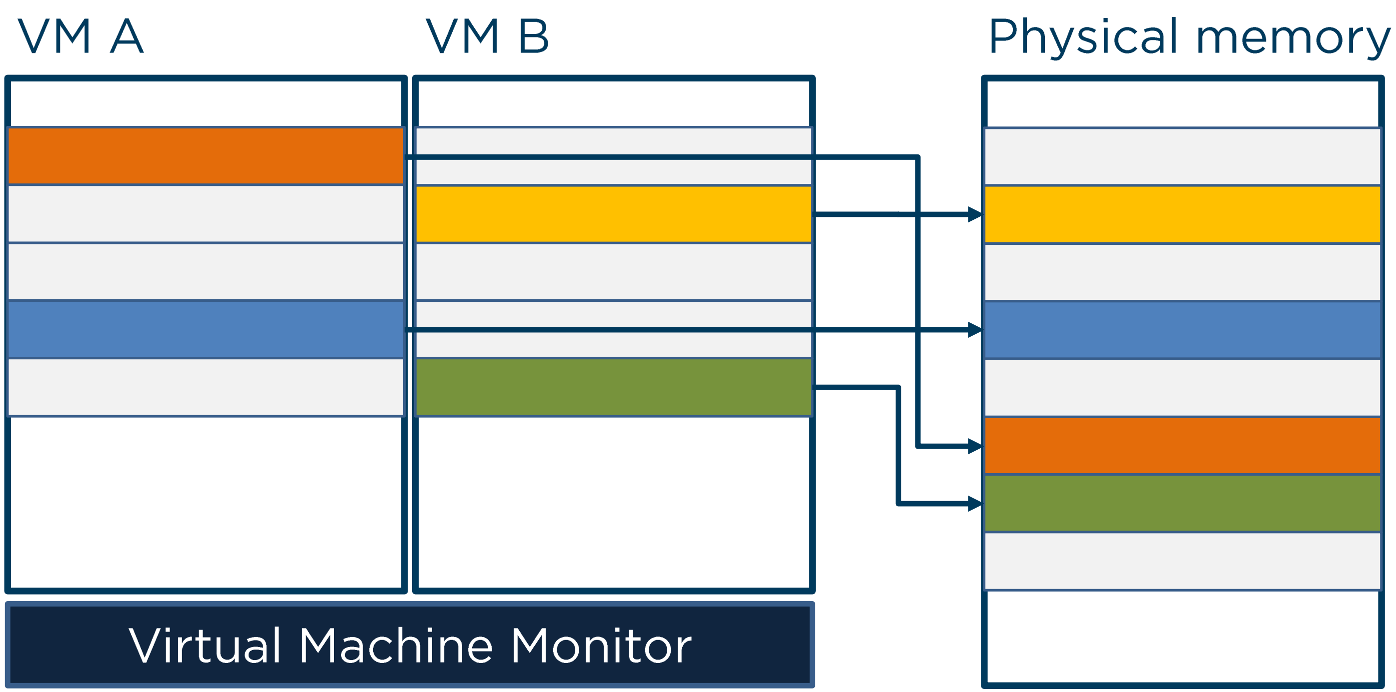
Memory deduplication



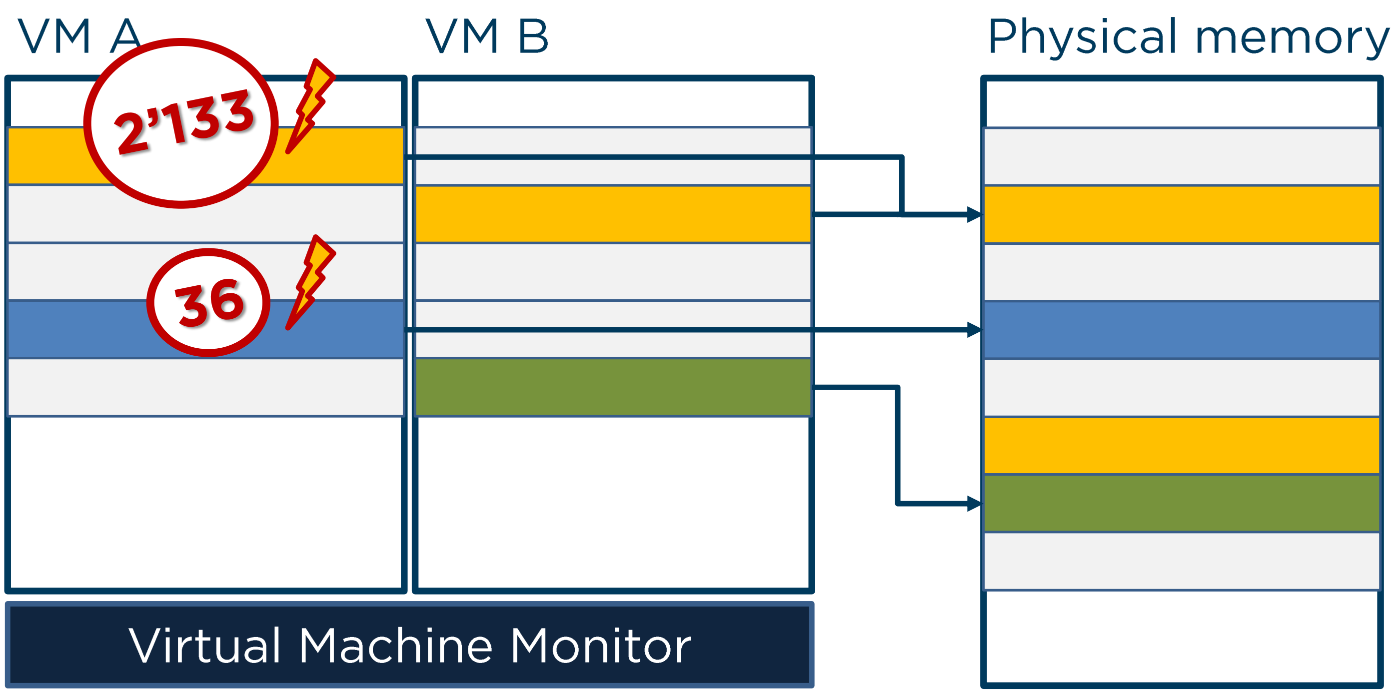
Memory deduplication



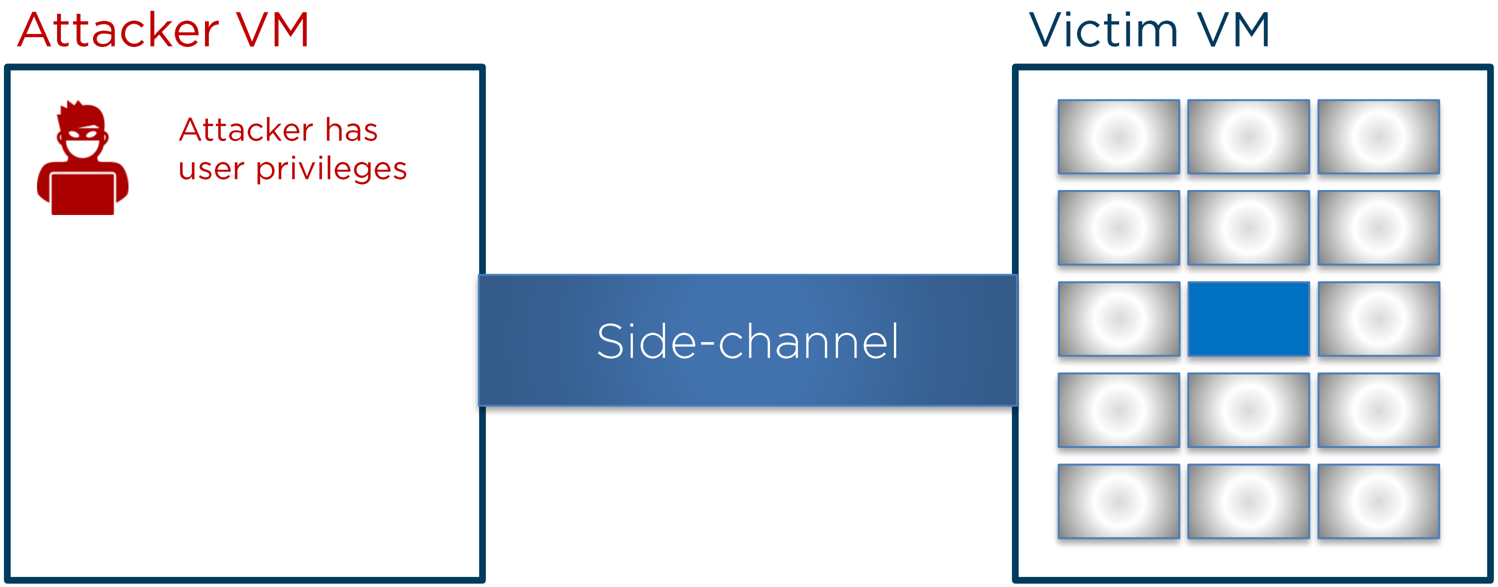
Memory deduplication



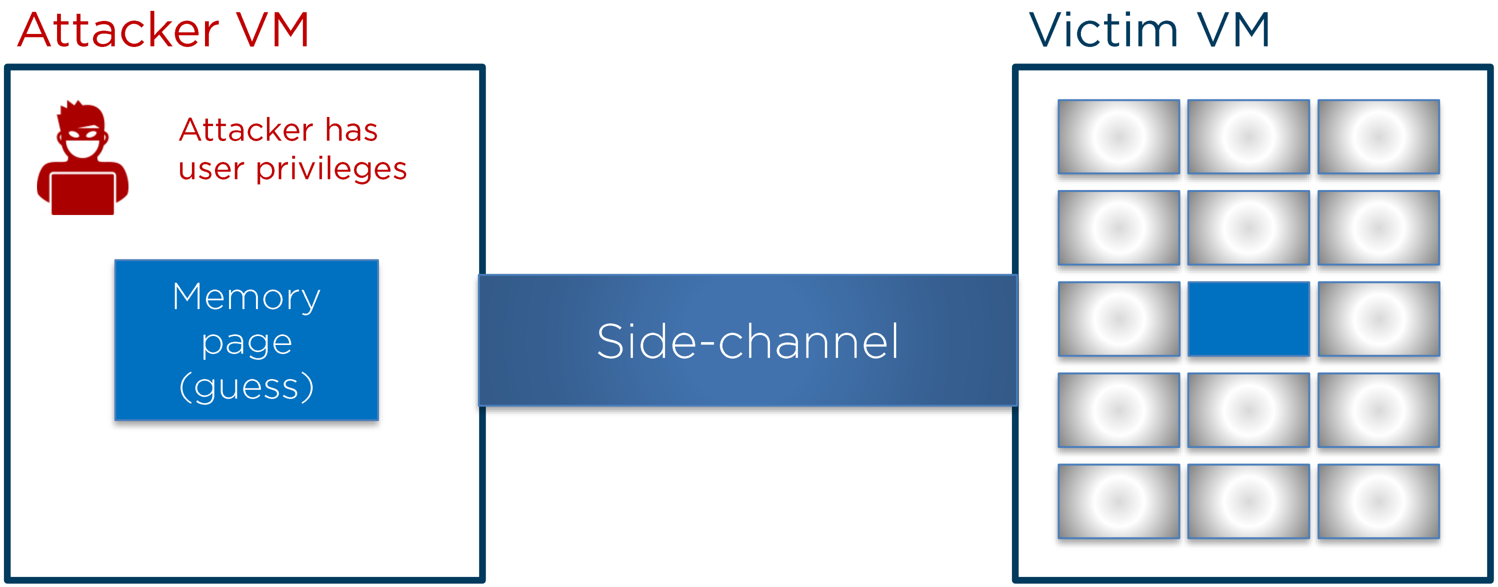
Memory deduplication



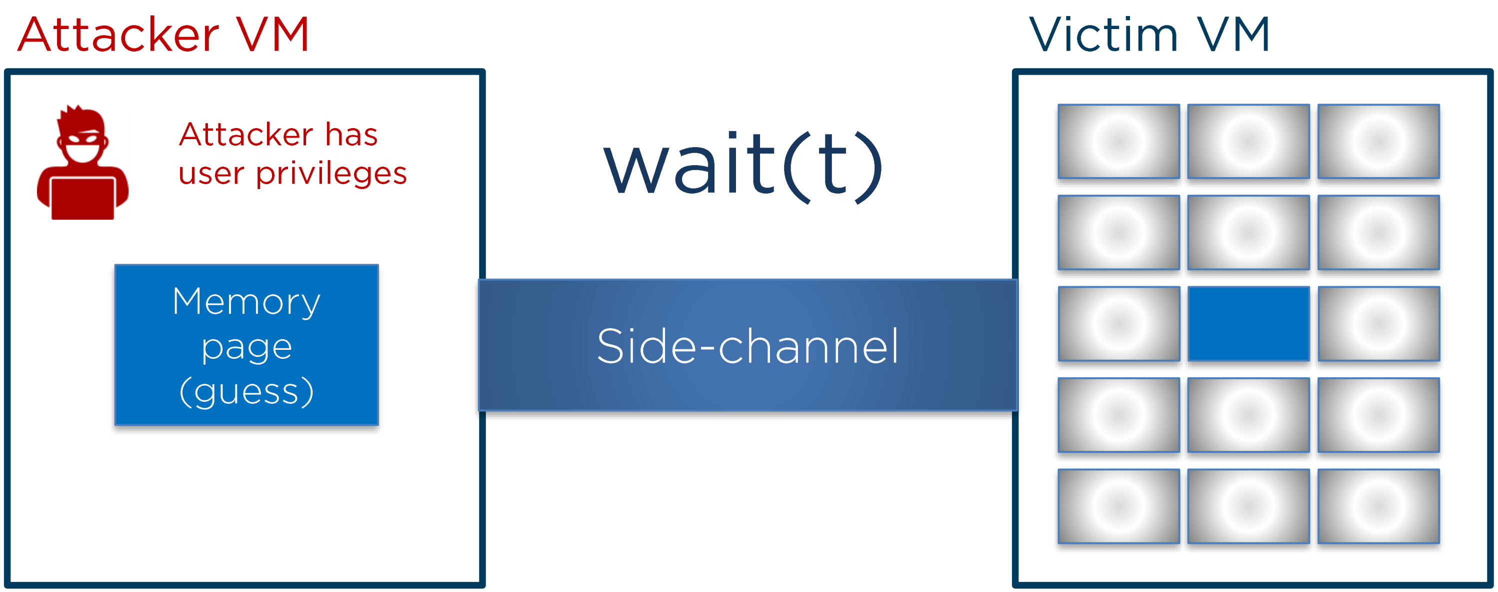
Memory deduplication side-channel



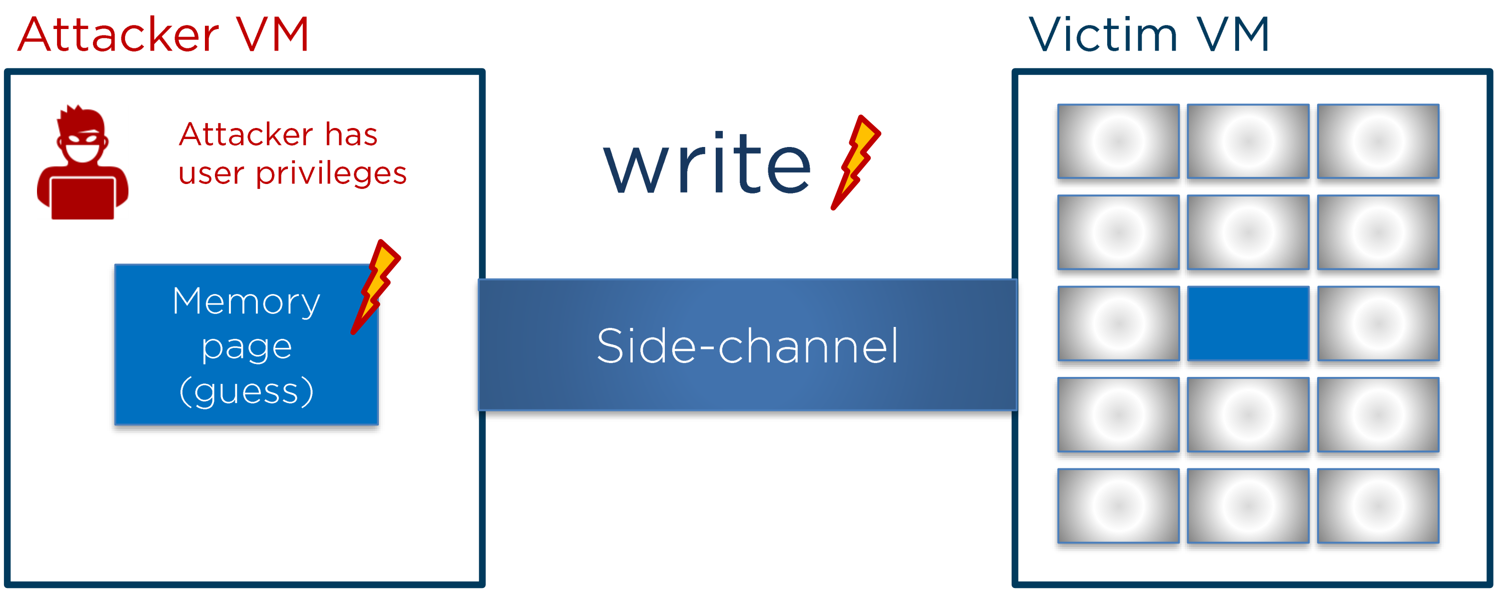
Memory deduplication side-channel



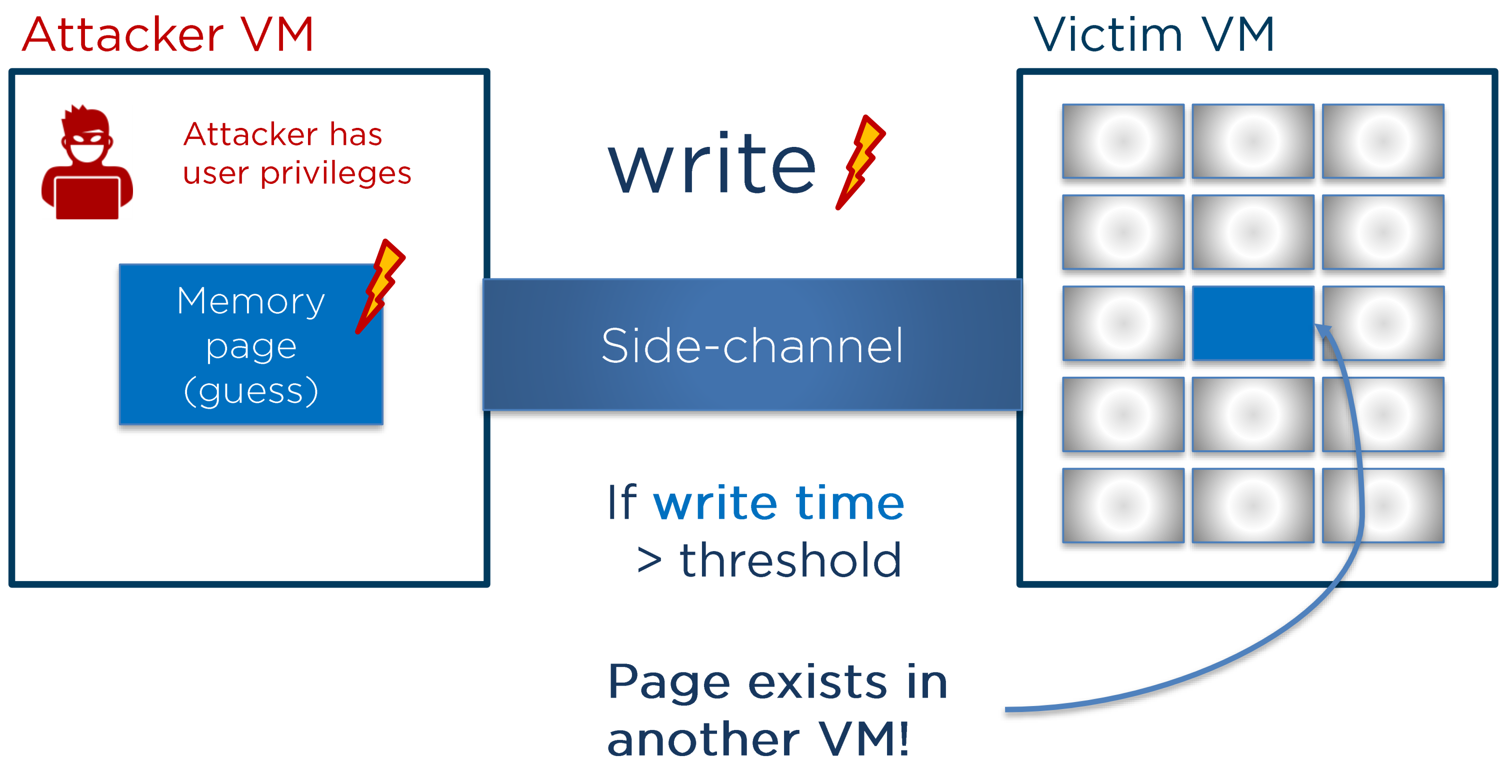
Memory deduplication side-channel



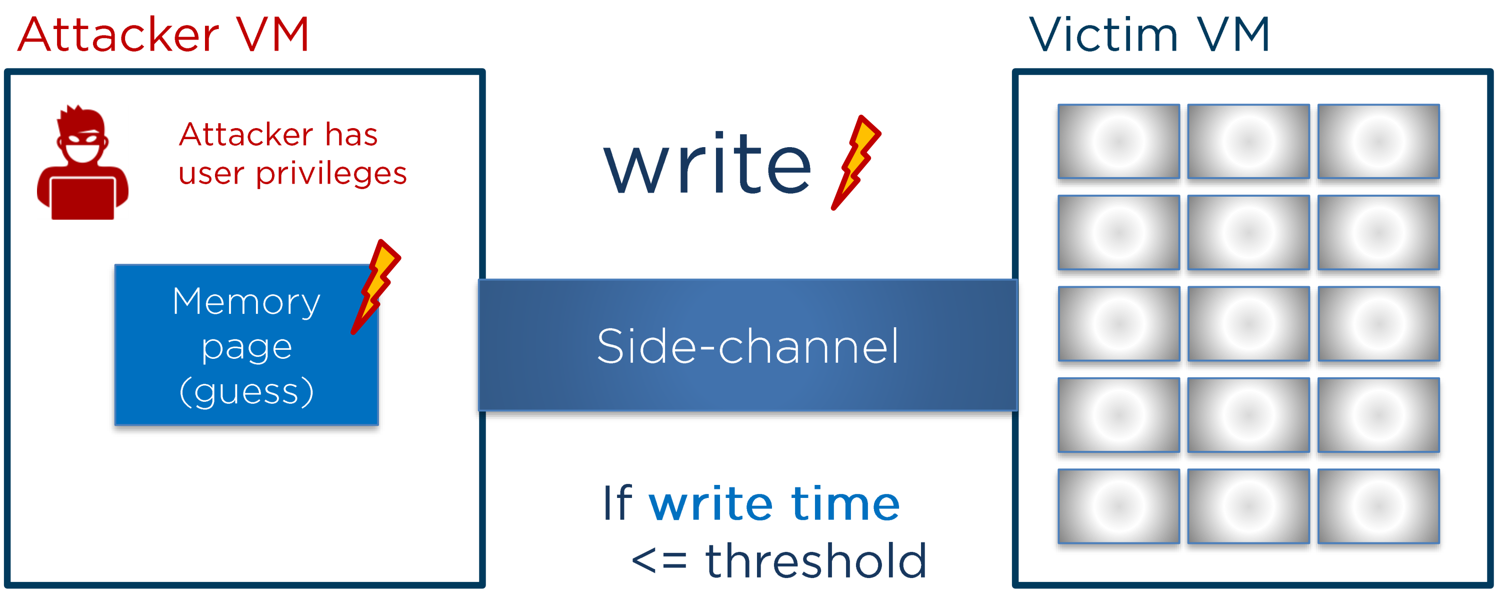
Memory deduplication side-channel



Memory deduplication side-channel



Memory deduplication side-channel



Page does not exist in another VM.

Memory deduplication side-channel

- Attacker can craft a page (guess)

Page contents?

Memory deduplication side-channel

- Attacker can craft a page (guess) **Page contents?**
- Wait for a certain amount of time **How long?**

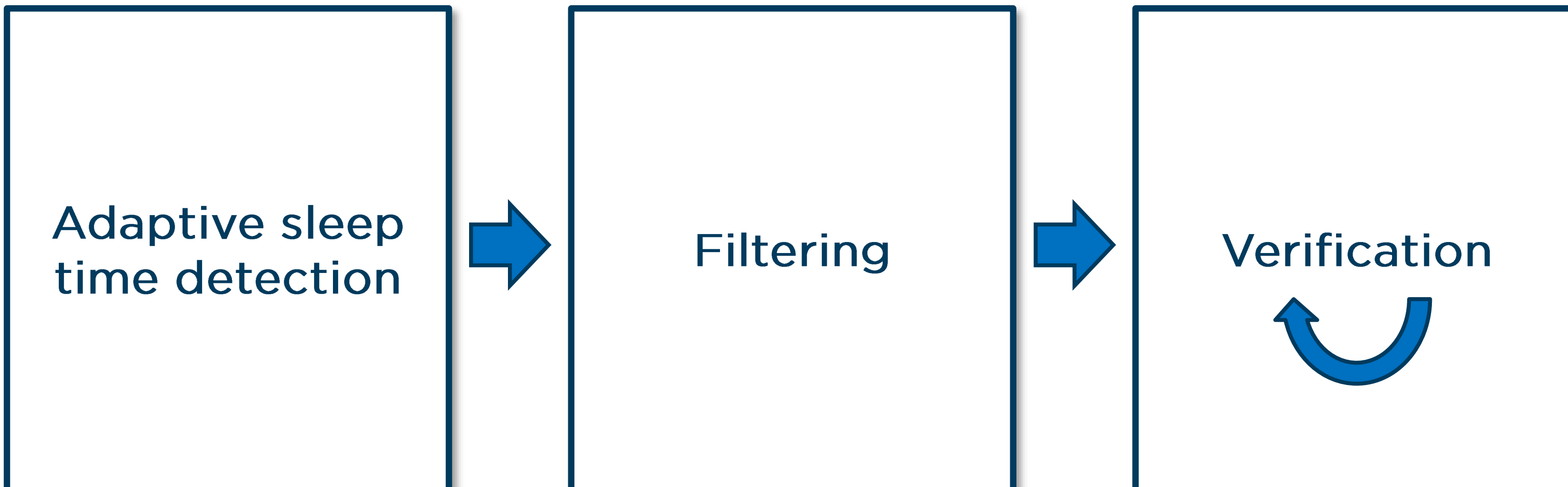
Memory deduplication side-channel

- Attacker can craft a page (guess) **Page contents?**
- Wait for a certain amount of time **How long?**
- Write to page and measure time **Noise? Threshold?**

Memory deduplication side-channel

- Attacker can craft a page (guess) **Page contents?**
- Wait for a certain amount of time **How long?**
- Write to page and measure time **Noise? Threshold?**
- Write time will reveal if page was shared

CAIN: Cross-VM ASL INtrospection



black hat EUROPE 2015

Suitable pages to break ASLR

Page contents?

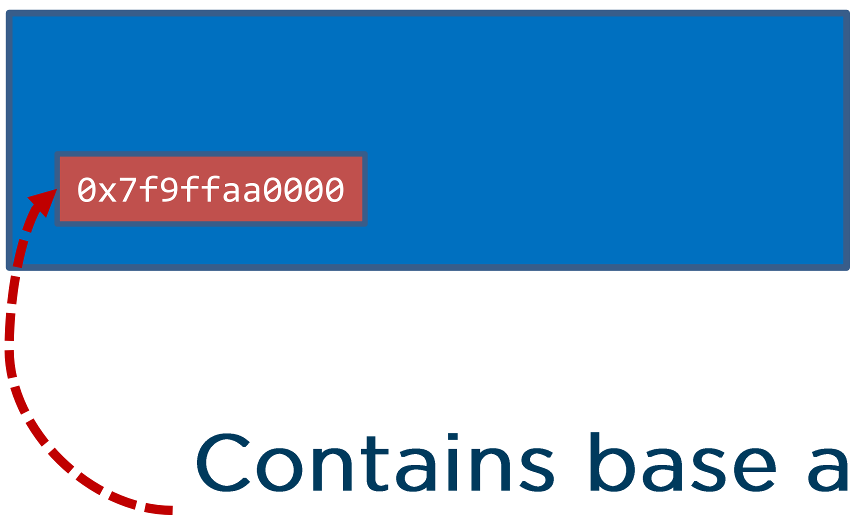
Suitable page
to break ASLR

Page aligned

- Mostly static
- Read-only in victim VM
- Known to exist

Suitable pages to break ASLR

Page contents?

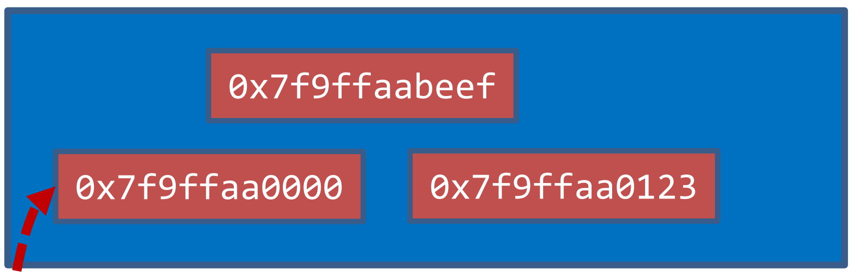


Page aligned

Contains base address
of an executable image

Suitable pages to break ASLR

Page contents?



Page aligned

Contains values derived from the **base address** of an executable image

Suitable pages to break ASLR

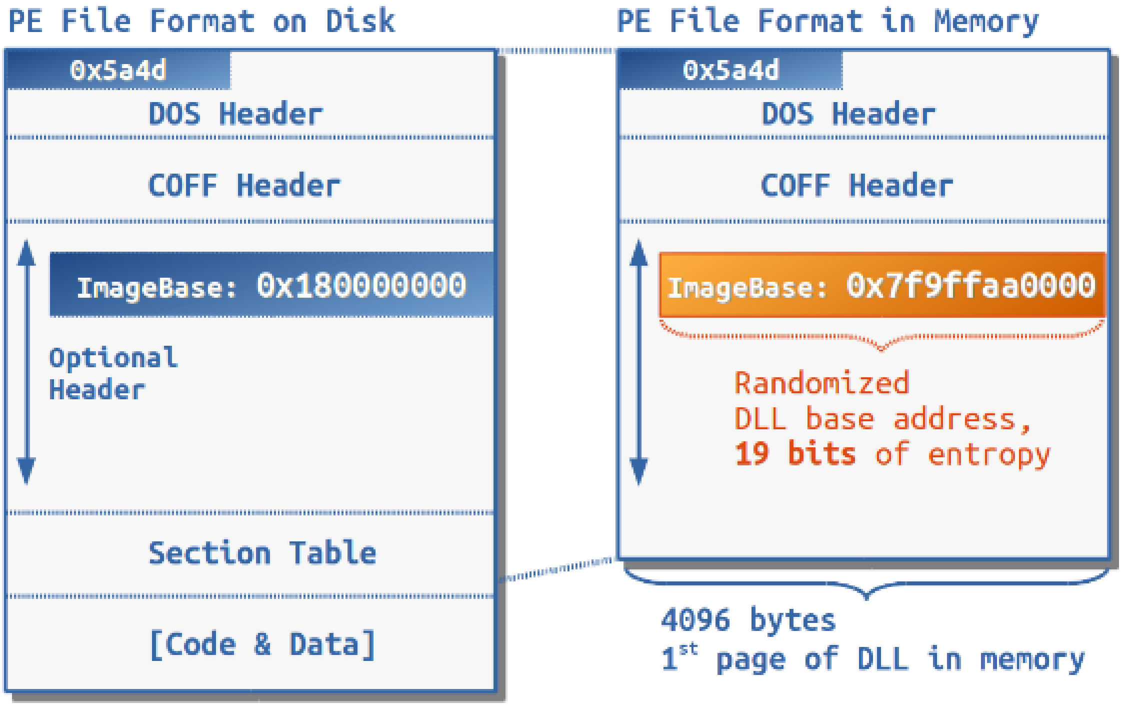
Page contents?

Page aligned

Entropy = ASLR entropy

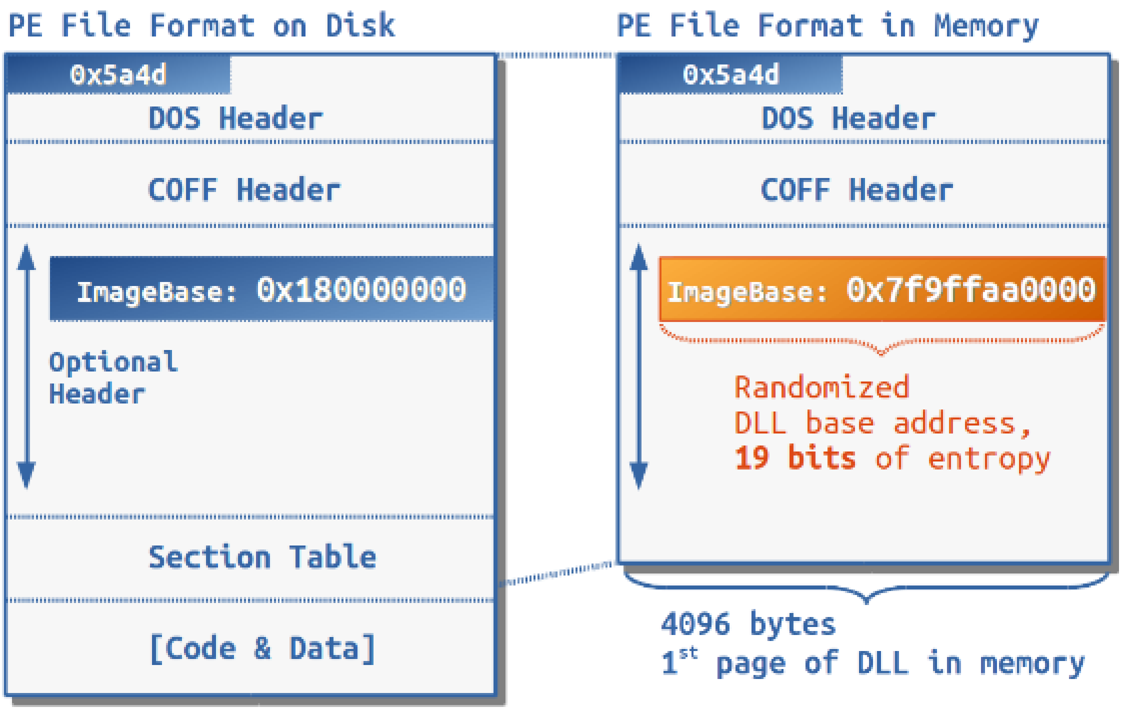
Suitable page under Windows

Page contents?



Suitable page under Windows

Page contents?



We also found a page under Linux x64...

Guessing the right address

Page contents?

> Well, you still have to guess

Guessing the right address

Page contents?

- Well, you still have to guess
  - 2^{19} base addresses for Windows x64
  - 524'288 guesses
  - One guess requires 1 page of memory

Based on http://sourceforge.net/projects/mpimd5bruteforc/

BRUTE FORCE

If it doesn't work, you're just not using enough.

Guessing the right address

Page contents?

- Well, you still have to guess
  - 2^{19} base addresses for Windows x64
  - 524’288 guesses
  - One guess requires 1 page of memory

- Attacker VM has much more memory
  - Fill up memory with all guesses
  - 2^{19} * 1 page of 4 KB = 2 GB

Brute-force all addresses

Page contents?

<Page with RBA guess>

0x7f9ffa70000
0x7f9ffa80000
0x7f9ffa90000
0x7f9ffaa0000
0x7f9ffab0000
0x7f9ffac0000
0x7f9ffad0000

.
.
.

Brute-force all addresses

Page contents?

<Page with RBA guess>

0x7f9ffa70000
0x7f9ffa80000
0x7f9ffa90000
0x7f9ffaa0000
0x7f9ffab0000
0x7f9ffac0000
0x7f9ffad0000

.
.
.

detect_shared_pages() → 0x7f9ffaa0000

Wait for how long?

How long?

- Depends on the memory deduplication implementation

Wait for how long?

How long?

- Depends on the memory deduplication implementation
- Varies depending on amount of memory used

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

Adaptive sleep-time detection

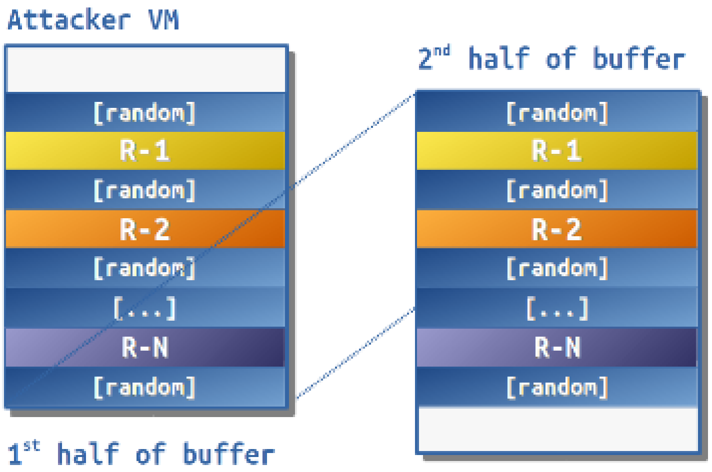
How long?

- Create random buffer

Adaptive sleep-time detection

How long?

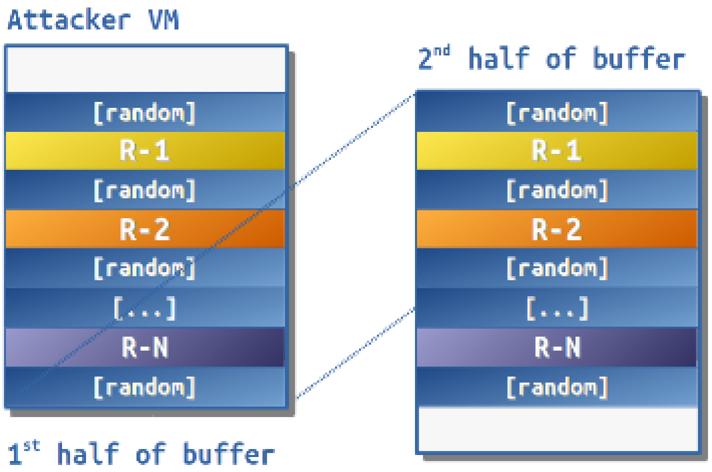
- Create random buffer
- Copy every second page of 1st half to the 2nd half



Adaptive sleep-time detection

How long?

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



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

Adaptive sleep-time detection

How long?

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

```
t_start = t_1 = 10 min, loop:

test_time(t_n)

If detection rate > 95%
      return t_n*1.2

If detection rate < 50%
      t_n+1 = t_n*2

Else
      t_n+1 = t_n*(1/[detection rate])
```

Detect merged pages

Noise?
Threshold?

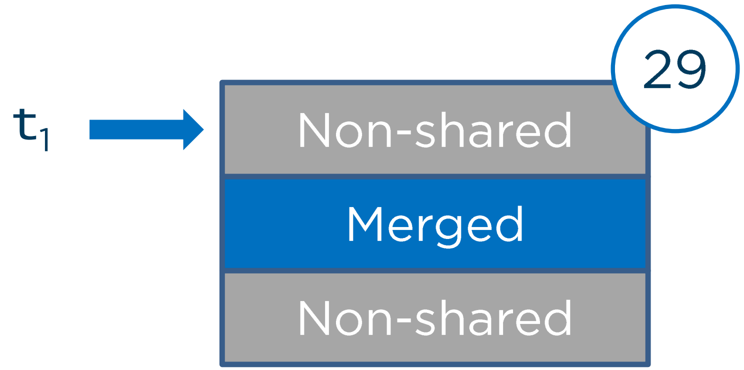
Non-shared

Merged

Non-shared

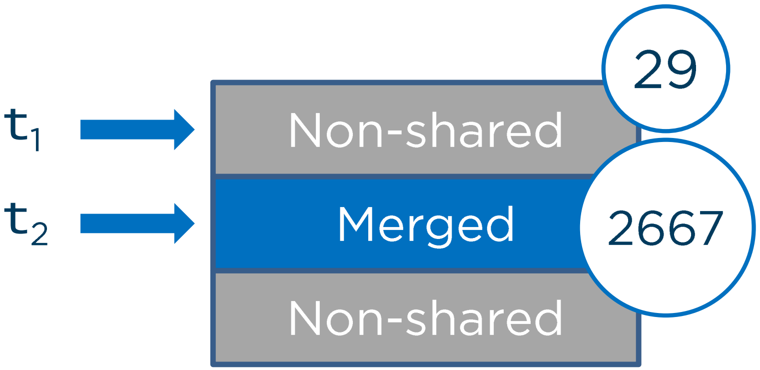
Detect merged pages

Noise?
Threshold?



Detect merged pages

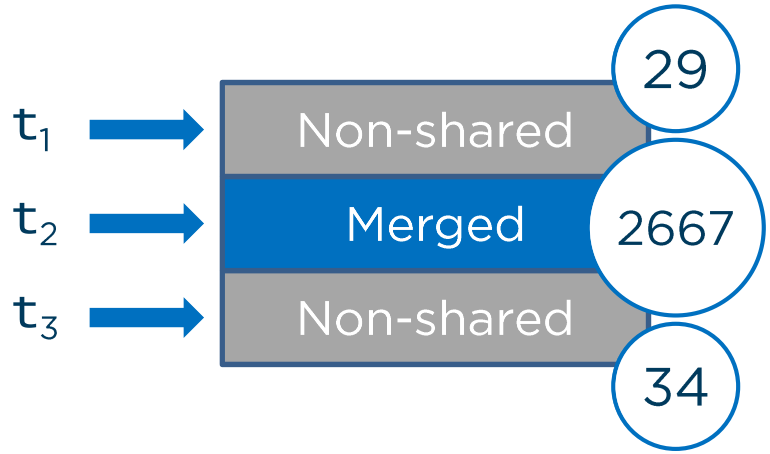
Noise?
Threshold?



Detect merged pages

Noise?
Threshold?

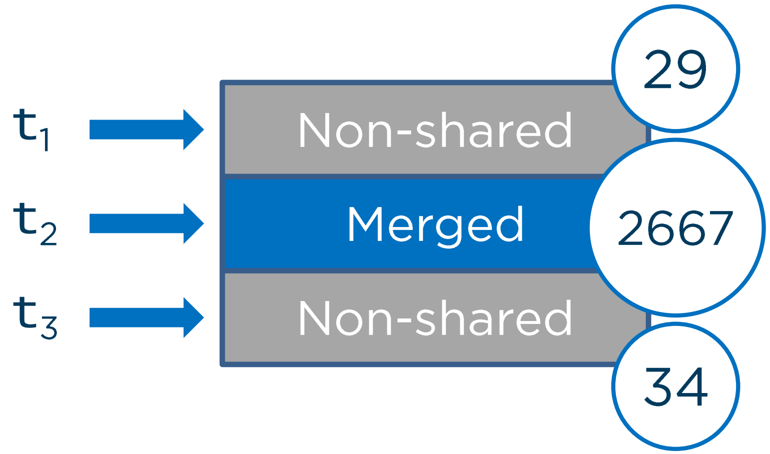
Measure write time with `rdtsc`
(Read Time Stamp Counter)



Detect merged pages

Noise?
Threshold?

Measure write time with rdtsc
(Read Time Stamp Counter)



$t_2 > 2 * (t_1+t_3)/2$ $\quad$ $t_{1,3} < M = 1000$ $\quad$ $t_1 < t_3,\ (t_3-t_1) < t_3/3$

Detect merged pages

se?
eshold?

These heuristics worked for different HW configurations

$t_2 > 2$ 1000 $t_1 < t_3$, $(t_3\text{-}t_1) < t_3/3$

Handling noise

Noise?
Threshold?

- Noise can affect write time
  - High write time for non-shared page
  - Missed 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	
	G1
	G2
	G3
	…
	Gn

Handling noise

Noise?
Threshold?

Handling noise

Noise?
Threshold?

Round 1

G1
G2
G3
...
Gn

detect()

G22
G37
G98

False positive
False positive

Round 2

G22
G37
G98

Handling noise

Noise?
Threshold?

Round 1

G1
G2
G3
...
Gn

detect()

G22 False positive
G37
G98 False positive

Round 2

G22
G37
G98

detect()

G22 False positive
G37

Handling noise

Noise?
Threshold?

Round 1

G1
G2
G3
…
Gn

detect() →

G22
G37
G98

False positive
False positive

Round 2

G22
G37
G98

detect() →

G22
G37

False positive

Round 3

G22
G37

Handling noise

Noise?
Threshold?

Round 1

G1
G2
G3
…
Gn

detect() →

G22
G37
G98

False positive
False positive

Round 2

G22
G37
G98

detect() →

G22
G37

False positive

Round 3

G22
G37

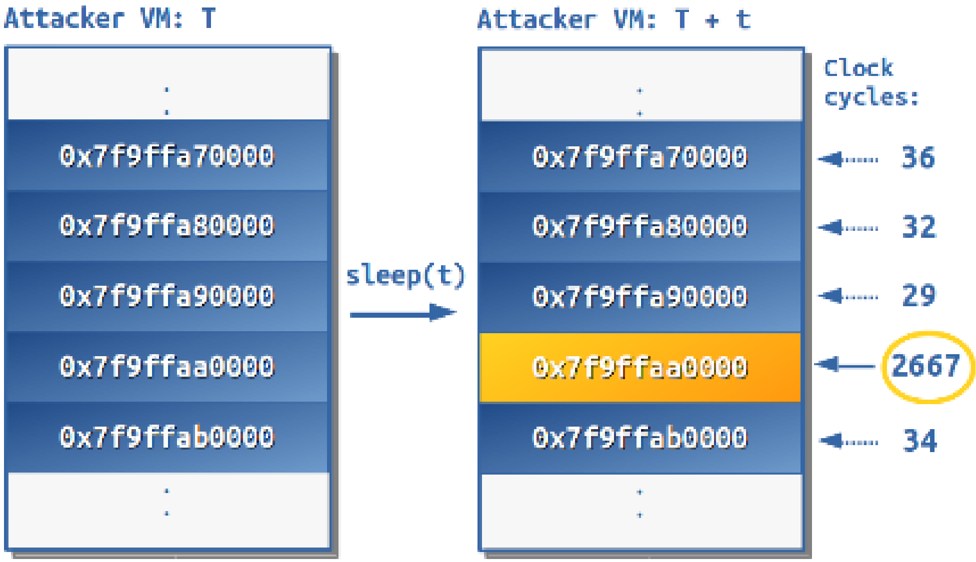
detect() →

G37

Filtering

Noise?
Threshold?

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



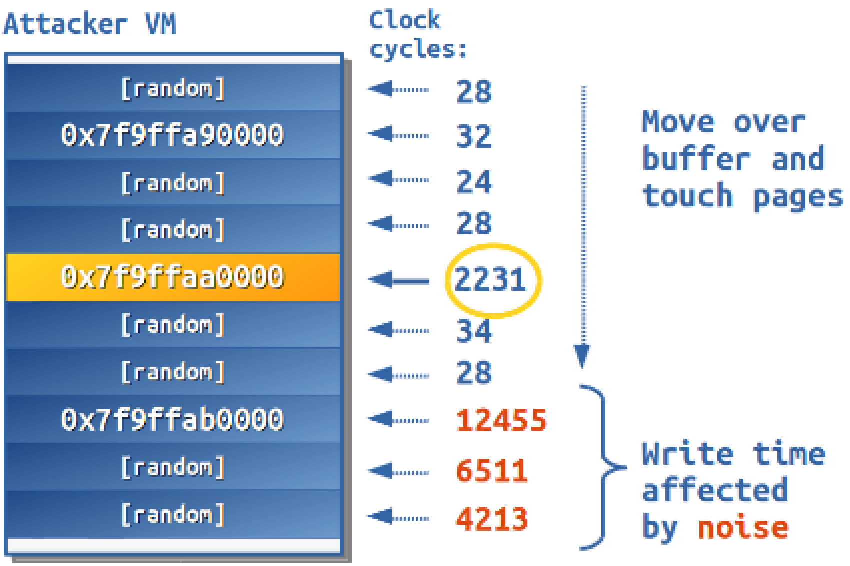
Verification

> Verify remaining potential candidates

> Use more memory per candidate

> Eliminate remaining false positives

Noise? Threshold?



Does it work?

Does it work?

We implemented CAIN and evaluated the PoC

CAIN requirements

- Side-channel exists (memdedup is on)
- Suitable page exists (for breaking ASLR)
  - Other «secrets» might be interesting too
- Entropy is within brute-forceable range
  - For given amount of attacker memory

CAIN attack phases

CAIN attack phases

Adaptive sleep time detection

Goal:
detect a wait time t required to merge pages

CAIN attack phases

Adaptive sleep time detection

Goal:
detect a wait time t required to merge pages

Filtering

Goal:
reduce number of candidates with as little memory

1 round

CAIN attack phases

Adaptive sleep time detection	Filtering	Verification
Goal: detect a wait time t required to merge pages	Goal: reduce number of candidates with as little memory	Goal: verify remaining candidates and remove false positives
	1 round	N rounds

Evaluation setup

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
- 6 victim VMs with Windows Server 2012 (6.2.9200 Build 9200)
  - 4 vCPUs, 4 GB per VM

Kernel same-page Merging (KSM)

- Enabled by default for KVM (Ubuntu Server)
  - Out-of-band Content Based Page Sharing (CBPS)

```
/sys/kernel/mm/ksm/run                  ‘1’ or ‘0’
/sys/kernel/mm/ksm/sleep_millisecs      e.g., 200 ms
/sys/kernel/mm/ksm/pages_to_scan        e.g., 100
```

Kernel same-page Merging (KSM)

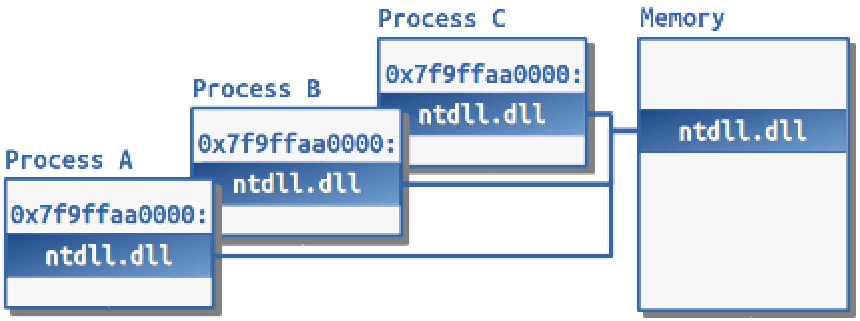
- Enabled by default for KVM (Ubuntu Server)
  - Out-of-band Content Based Page Sharing (CBPS)

```
/sys/kernel/mm/ksm/run                    ‘1’ or ‘0’
/sys/kernel/mm/ksm/sleep_millisecs        e.g., 200 ms
/sys/kernel/mm/ksm/pages_to_scan          e.g., 100

1000/sleep_millisecs * pages_to_scan = pages per second
e.g., (1000/200ms) * 100 = 500 pages/sec
```

ASLR in Windows x64

- High Entropy ASLR



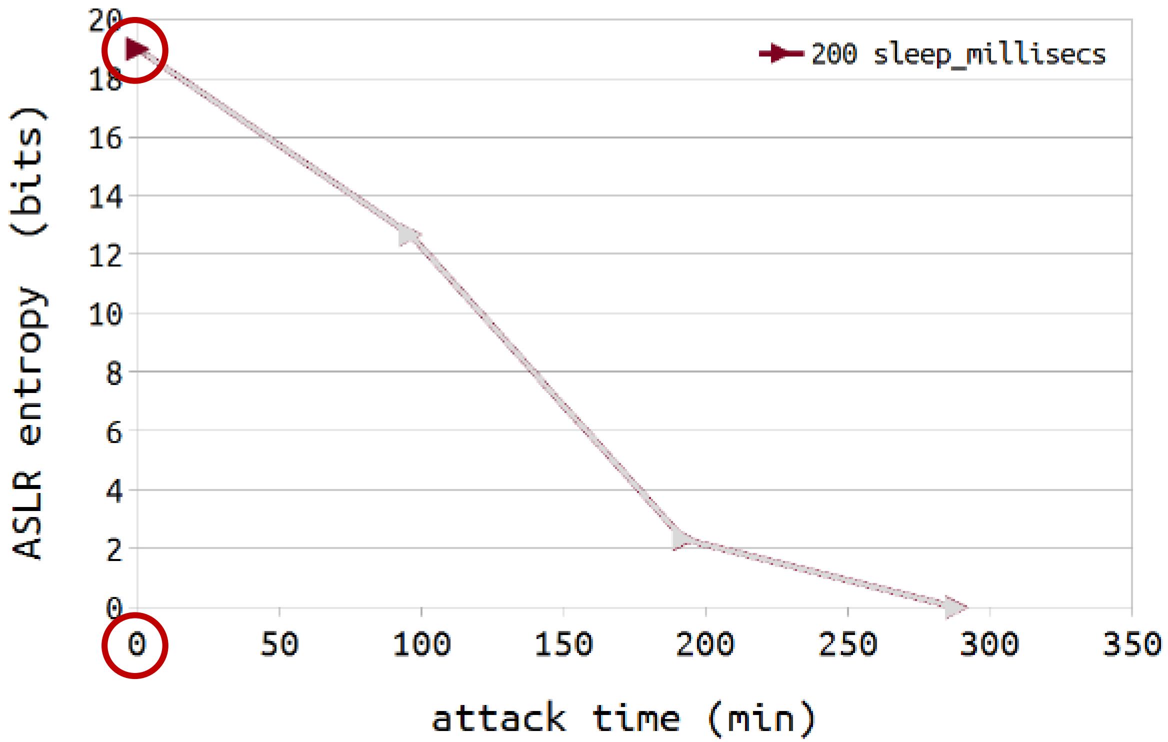
- 33 bits for stacks
- 24 bits for heaps

- 17 bits for executables
- 19 bits for DLLS

System-wide
at boot-time
for certain images

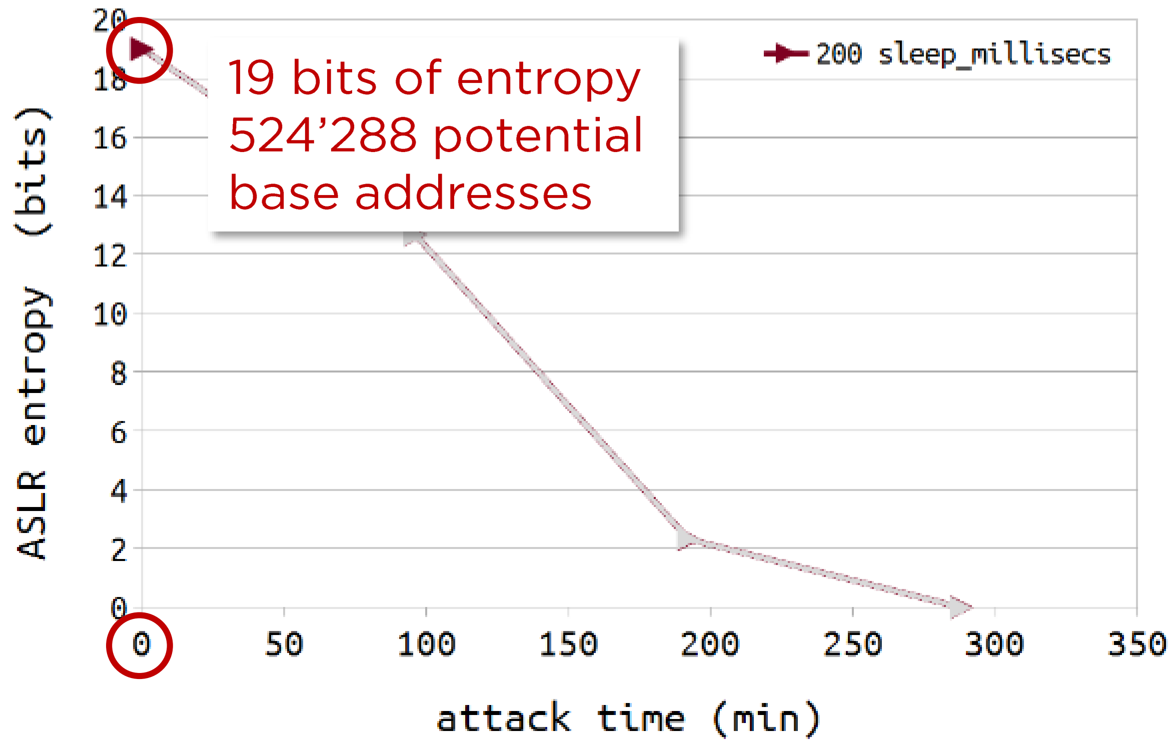
http://www.nynaeve.net/?p=198
https://media.blackhat.com/bh-us-12/Briefings/M_Miller/BH_US_12_Miller_Exploit_Mitigation_Slides.pdf

Attacking a single Windows VM



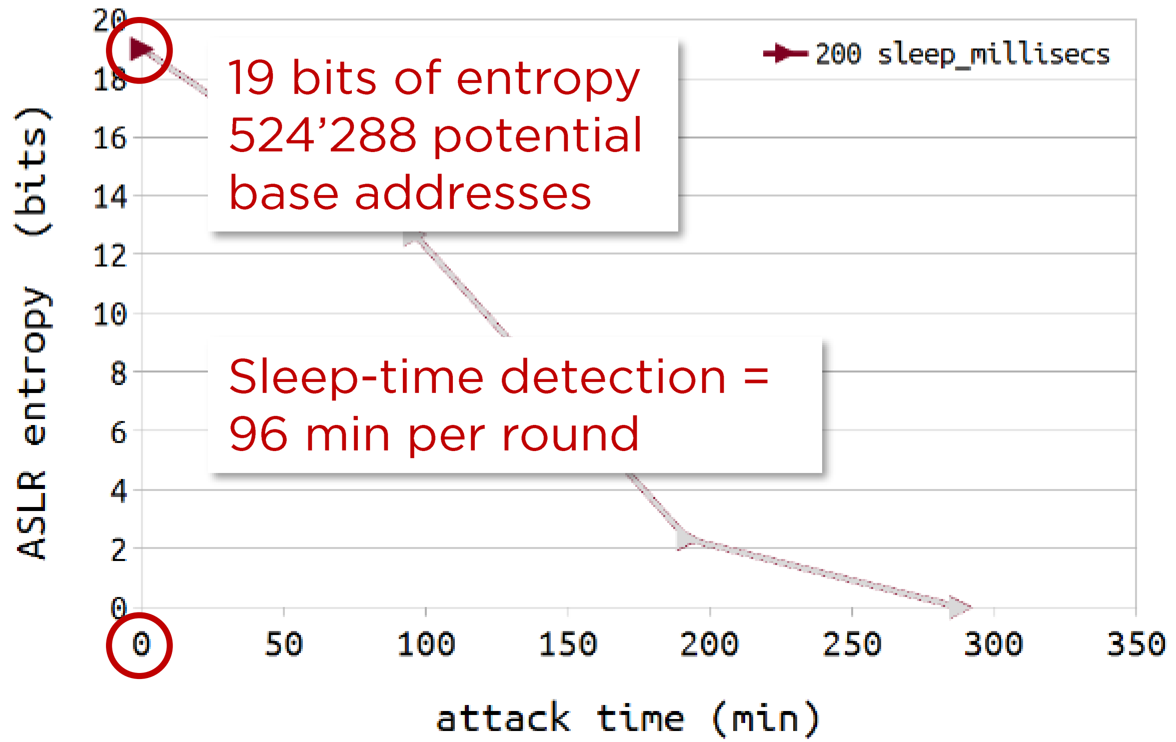
time t = 0 min

Attacking a single Windows VM



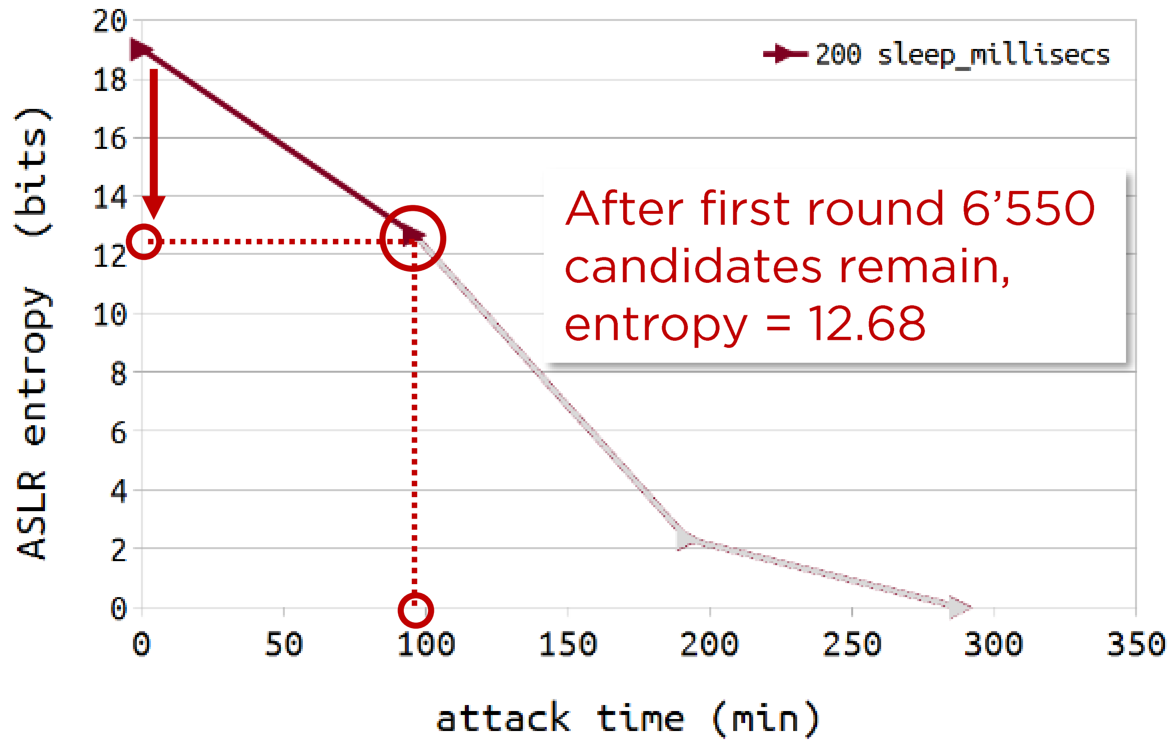
time t = 0 min

Attacking a single Windows VM



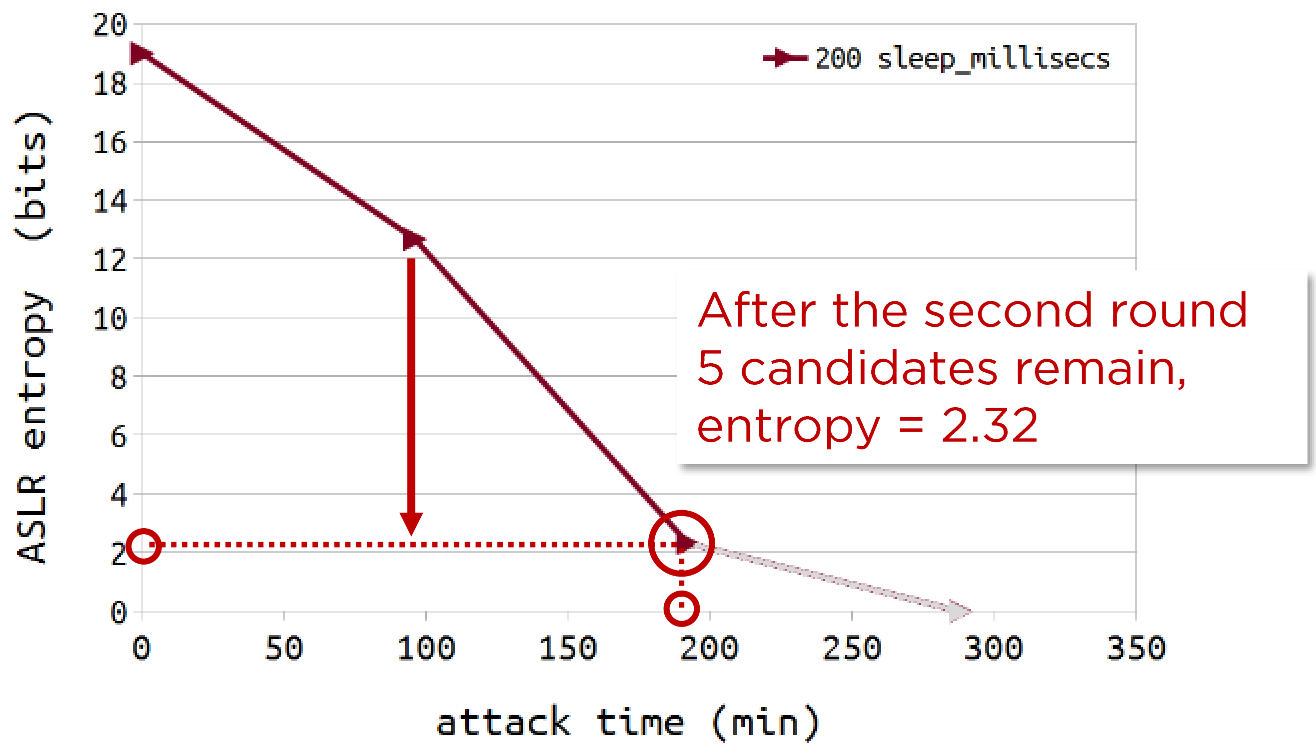
time t = 0 min

Attacking a single Windows VM



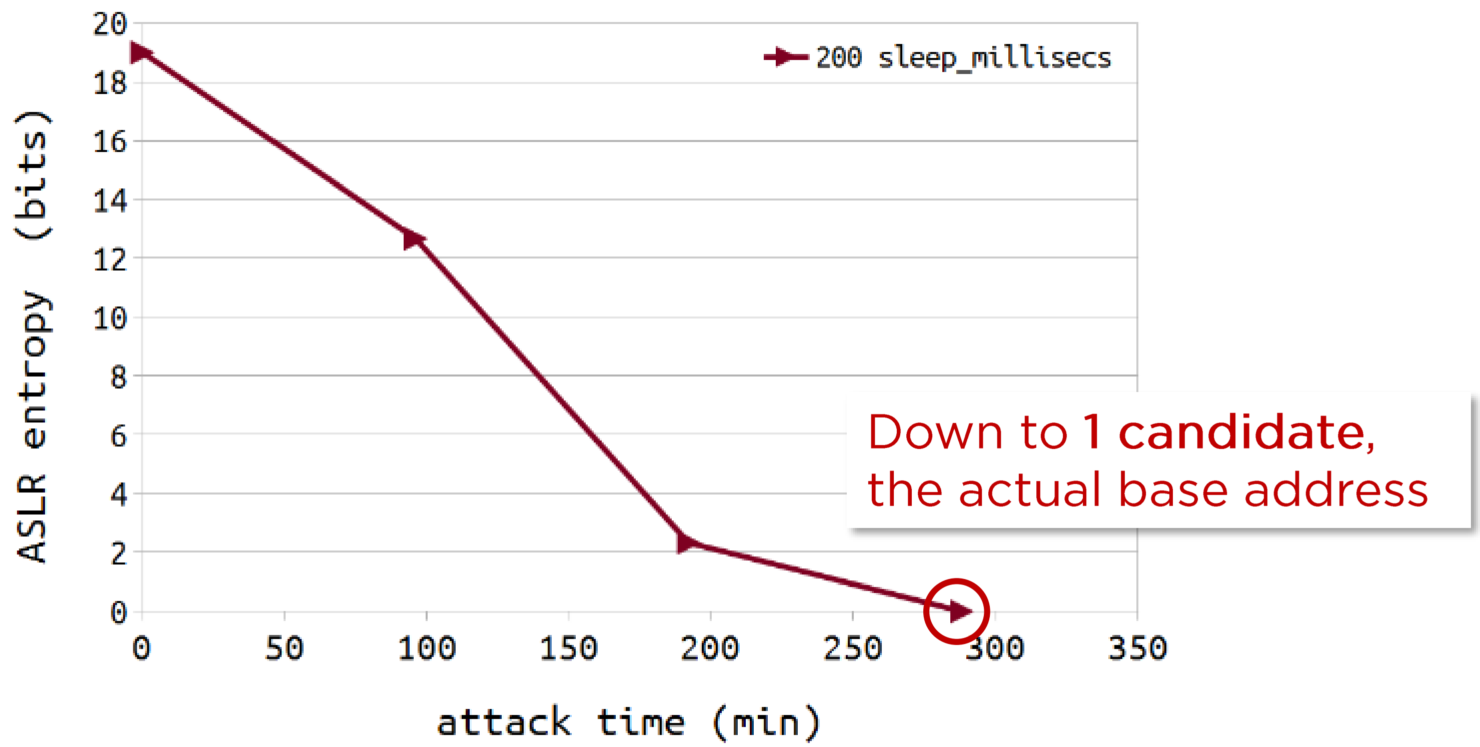
time t = 96 min

Attacking a single Windows VM



time t = 192 min

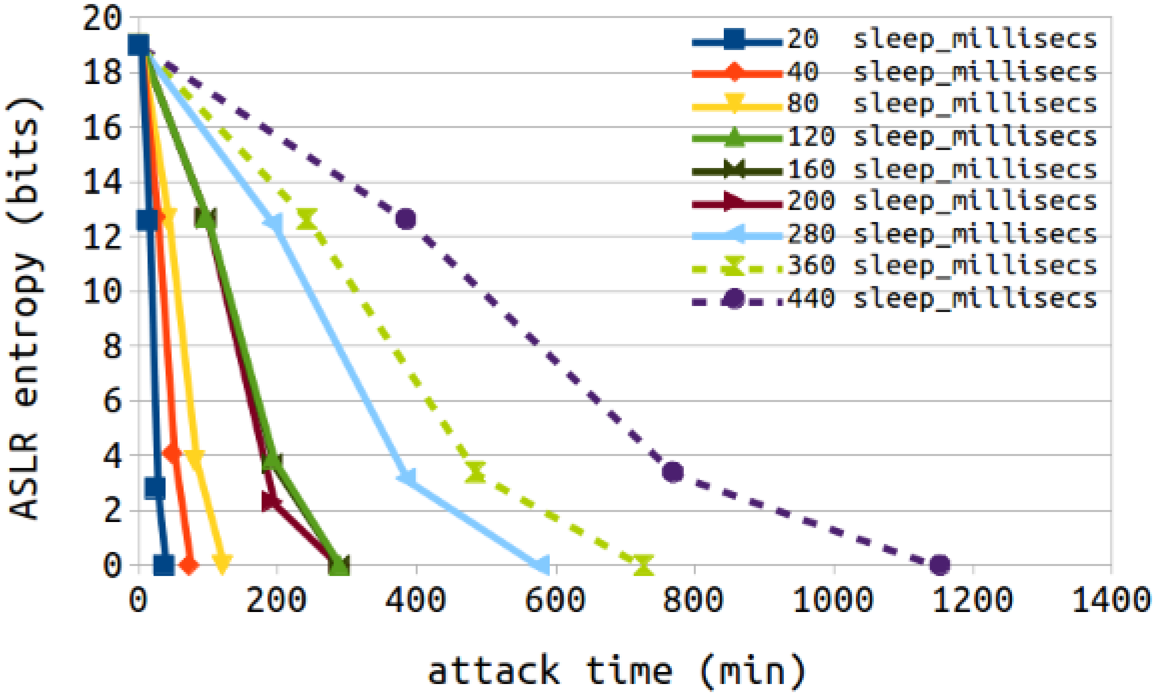
Attacking a single Windows VM



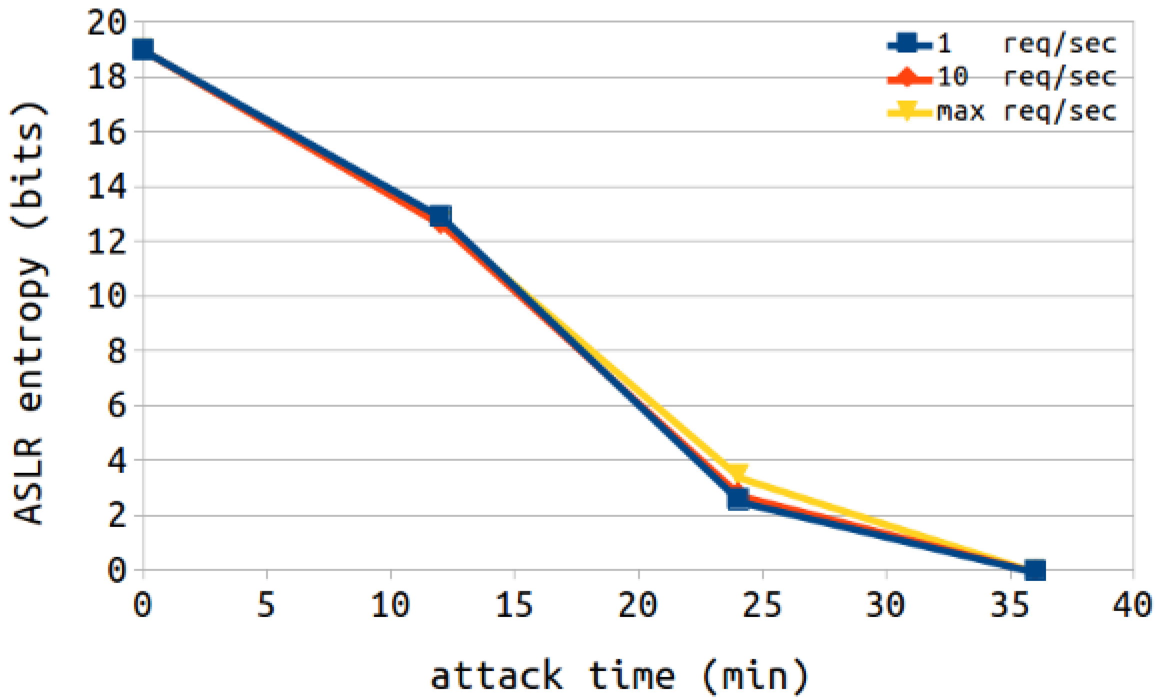
time t = 288 min

Attacking a single Windows VM

and various merge times



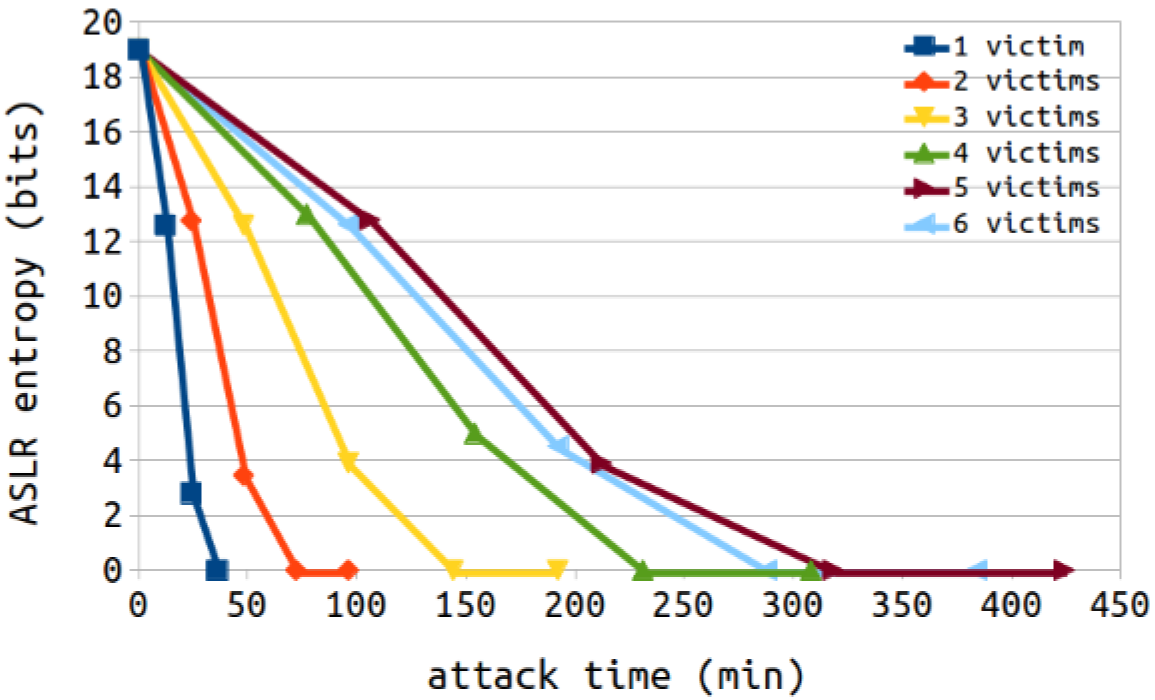
Attacking a Windows VM under load



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

Demo

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…

Post-CAIN exploitation

> What will the attacker really get?

Post-CAIN exploitation

- What will the attacker really get?

- Assume a vulnerability where:
  - Attacker can reliably hijack EIP/RIP
  - Attacker has some control over data
  - No infoleak for ASLR (code) bypass

Post-CAIN exploitation

Title : The Art of Exploitation: MS IIS 7.5 Remote Heap Overflow

Author : redpantz

```
                    ==Phrack Inc.==

      Volume 0x0e, Issue 0x44, Phile #0x0c o

|=----------------------------------------------------
|=-------------=[       The Art of Exploitation
|=----------------------------------------------------
|=-------------------=[   Exploiting MS11-004    ]=----
|=----------=[ Microsoft IIS 7.5 remote heap buffer ov
|=----------------------------------------------------
|=------------------------=[  by redpantz  ]=---------
|=----------------------------------------------------

--[ Table of Contents

1  - Introduction
2  - The Setup
3  - The Vulnerability
4  - Exploitation Primitives
5  - Enabling the LFH
6  - FreeEntryOffset Overwrite
7  - The Impossible
8  - Conclusion
9  - References
10 - Exploit (thing.py)


--[ 1 - Introduction

Exploitation of security vulnerabilities has greatly i
difficulty since the days of the Slammer worm. There h
exploitation mitigations implemented since the early 2000's. Many of these
mitigations were focused on the Windows heap; such as Safe Unlinking and
Heap Chunk header cookies in Windows XP Service Pack 2 and Safe Linking,
expanded Encoded Chunk headers, Terminate on Corruption, and many others in
Windows Vista/7 [1].
```

Phrack #68 from 2012-04-14

Exploiting MS11-004

Microsoft IIS 7.5 remote heap buffer overflow

by **redpantz**

Post-CAIN exploitation

```
--[ 8 - Conclusion

Although Microsoft and many others claimed that this vulnerability would be
impossible to exploit for code execution, this paper shows that with the
correct knowledge and enough determination, impossible turns to difficult.
```

http://phrack.org/issues/68/12.html#article

Post-CAIN exploitation

```
--[ 8 - Conclusion

Although Microsoft and many others claimed that this vulnerability would be
impossible to exploit for code execution, this paper shows that with the
correct knowledge and enough determination, impossible turns to difficult.
```

.....

```
    10) This will obtain EIP with multiple registers pointing to
        user-controlled data. From there ASLR and DEP will need to be
        subverted to gain code execution. Take a look at
        DATA_STREAM_BUFFER.Size, which will determine how many bytes are
        sent back to a user in a response

Although full arbitrary code execution wasn't achieved in the exploit, it
still proves that a remote attacker can potentially gain control over EIP
via a remote unauthenticated FTP connection that can be used to subvert the
security posture of the entire system, instead of limiting the scope to a
denial of service.
```

http://phrack.org/issues/68/12.html#article

Real-world environments

- Memory deduplication is used in commercial public cloud environments
  - some providers disable it
  - some VMMs do not support it at all
  - usually not apparent to customers if used or not

- If KVM is used, it is likely enabled

- Most providers offer 4 GB and more
  - 10 – 30 VMs per host is usual

Affected VMMs

Vendor	Memdedup	cert.org*	We think	Enabled by default
Linux KVM	KSM	Affected	Yes	n.a.
Ubuntu (KVM)	KSM	-	Yes	Yes
Red Hat (KVM)	KSM	Affected	Yes	Yes
Parallels	-	Affected	Not sure	-
Microsoft Hyper-V	None	Not Affected	Not Affected	n.a.
Xen	None (yet?)	Not Affected	Not Affected	n.a.
Oracle VirtualBox	PageFusion	Unknown	Probably	No
VMware	TPS	Unknown	Probably	No

* https://www.kb.cert.org/vuls/id/935424

Disclosure & vendor responses

Disclosure & vendor responses

> Well, who should fix it?

Disclosure & vendor responses

- Well, who should fix it?

- We informed several VMM vendors that we thought might be affected
  - Advisory sent on June 4, 2015, all replied
  - Overall perception: **low severity issue**

Disclosure & vendor responses

- Well, who should fix it?
- We informed several VMM vendors that we thought might be affected
  - Advisory sent on June 4, 2015, all replied
  - Overall perception: **low severity issue**

= won’t fix

Mitigations

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

Mitigations

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

Black Hat Sound Bytes

> Memory deduplication considered harmful

> CAIN breaks ASLR of co-located VMs

> Don’t use memory dedup in public clouds

Memory deduplication considered harmful

CAIN breaks ASLR of co-located VMs

Don’t use memory dedup in public clouds

BLACK HAT EUROPE 2015

Antonio Barresi
antonio.barresi@xorlab.com
@AntonioHBarresi

Kaveh Razavi
kaveh@cs.vu.nl