



MARCH 26-29, 2019

MARINA BAY SANDS / SINGAPORE

## **Winter Is Coming Back: Defeating the Most Advanced Rowhammer Defense to Gain Root/Kernel Privileges**

**Yueqiang Cheng, Zhi Zhang, Surya Nepal, Zhi Wang**

# About the speakers



- Dr. Yueqiang Cheng @ Baidu X-Lab



- Zhi Zhang @ University of New South Wales



- Dr. Surya Nepal @ CSIRO Data61, Australia



- Dr. Zhi Wang @ Florida State University

# Overview

- A brief introduction of rowhammer
- The most advanced rowhammer defense
- A proof-of-concept exploit
- Demo
- Mitigations
- Sound bytes

# What is rowhammer?



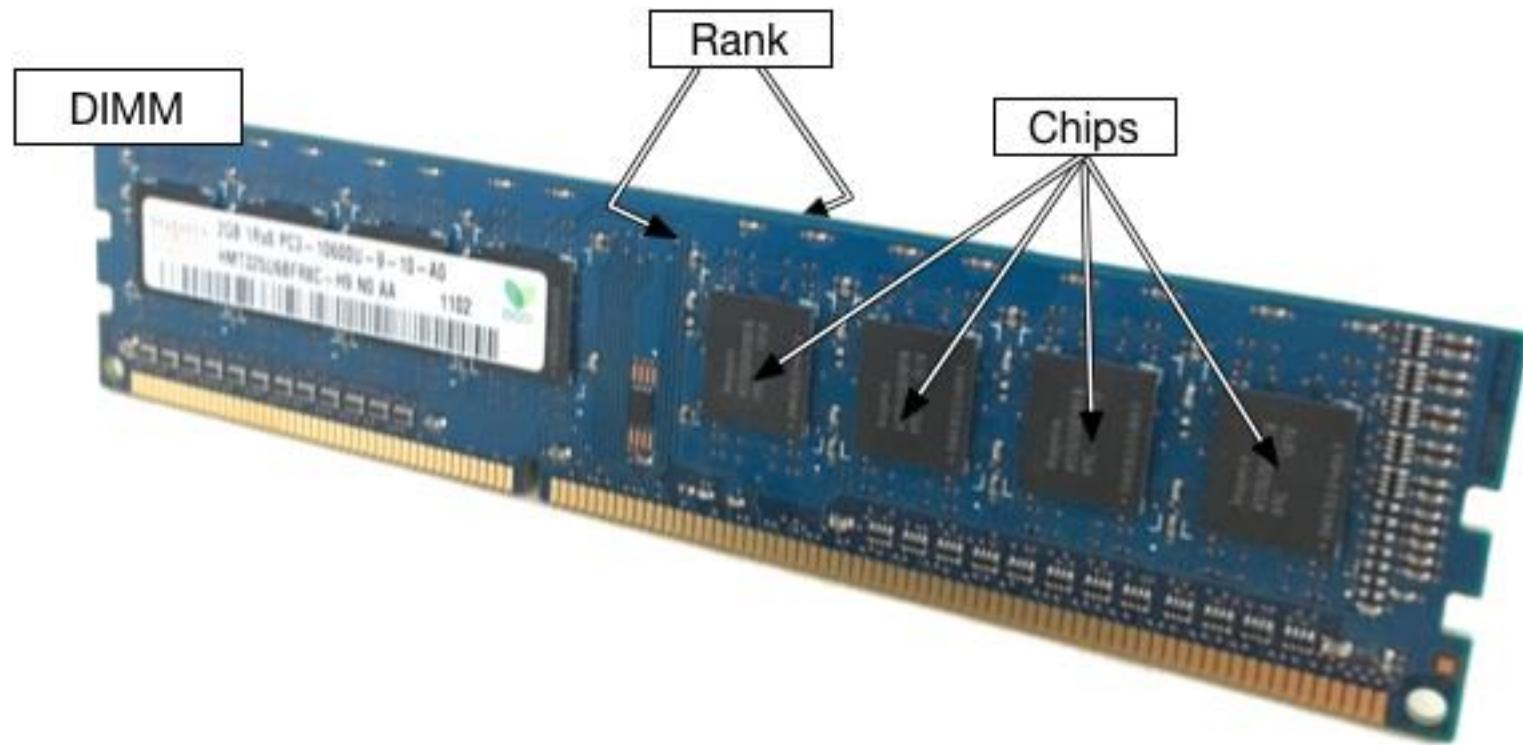
Rowhammer:

Frequently accessing

DRAM row

# Hierarchical structure of DRAM

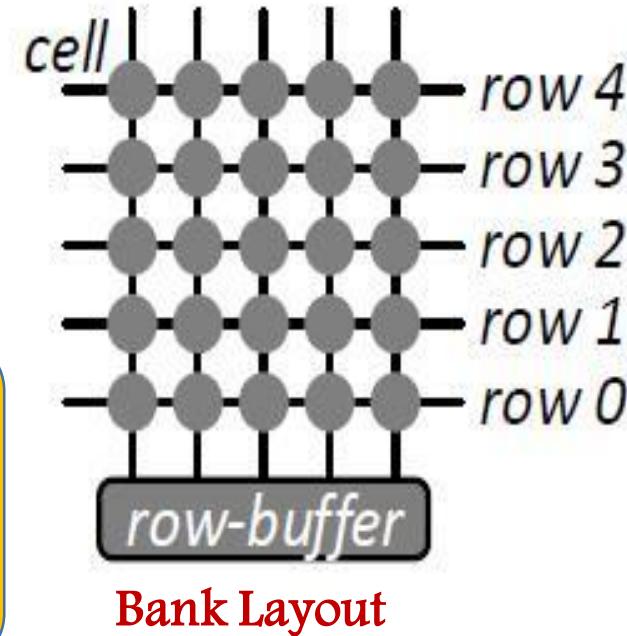
Diagram from Brasser et al.



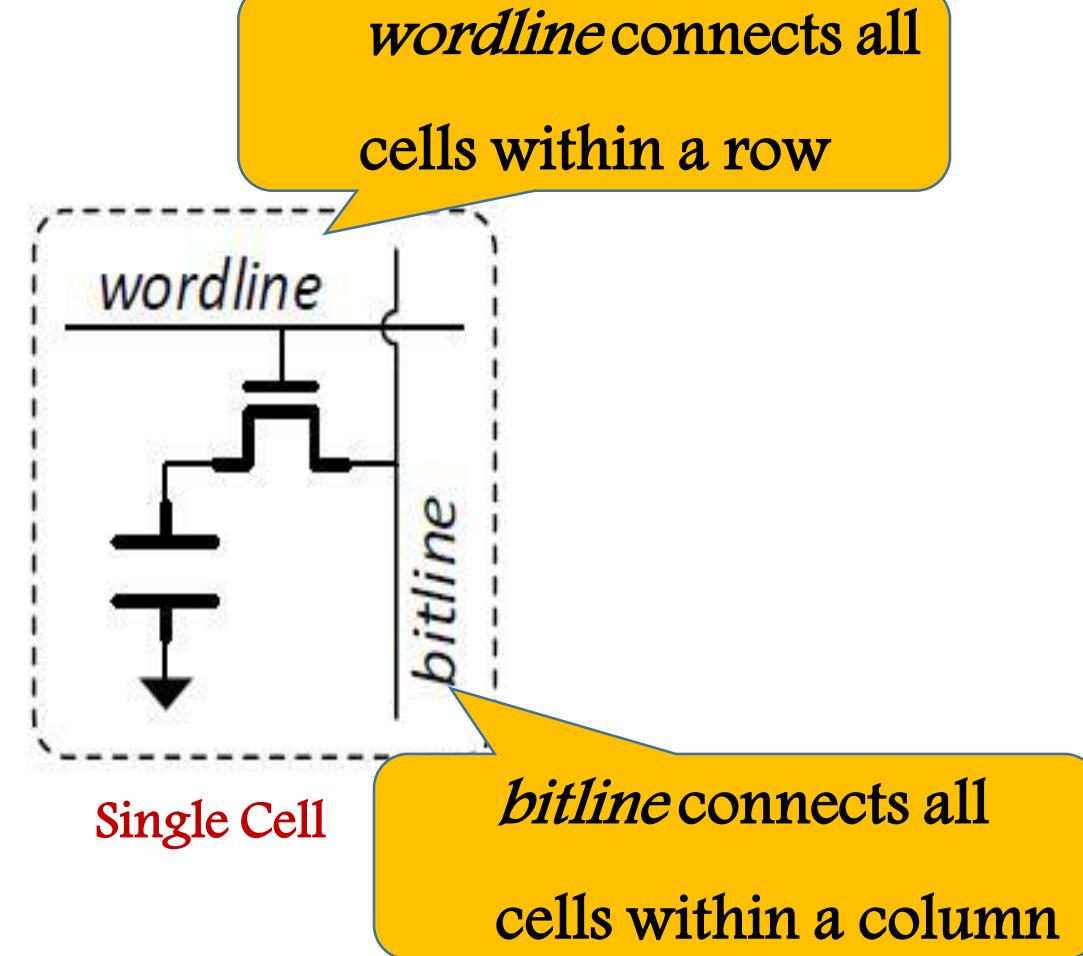
# Structure of DRAM bank

Diagram from Kim et al.

Each bank has rows of cells

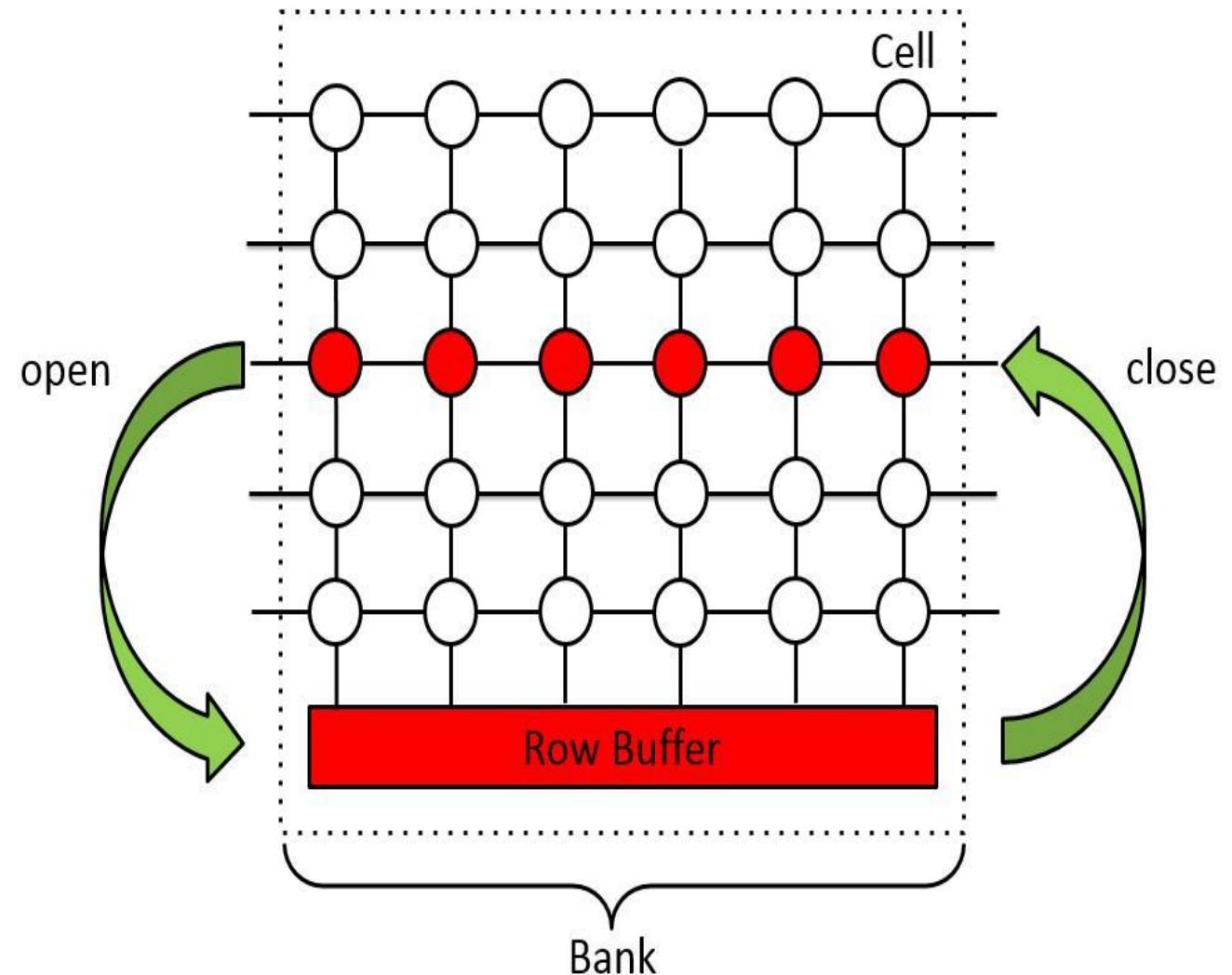


Each cell consists of a capacitor and an access-transistor



# Memory access

- Activating/opening a row
- Accessing row buffer
- Deactivating/closing the row



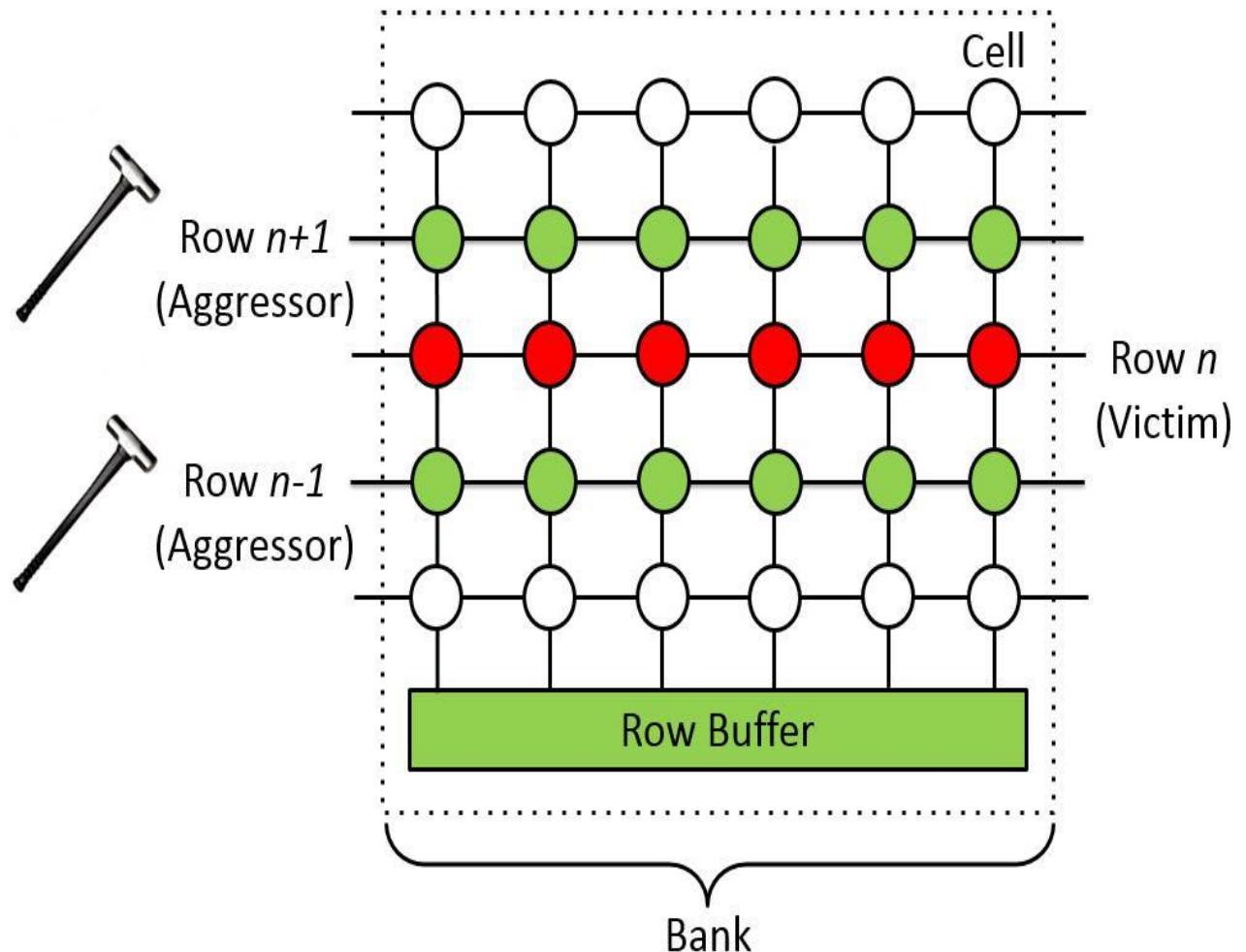
# DRAM refresh

- Capacitors of cells can **lose charge** over time
- A row of cells needs to be **periodically charged/refreshed**  
(refreshing and opening a row are **identical** for charging a row)
- The refresh interval is typically **64 ms**

# Rowhammer bug

Kim et al. *key observation* (ISCA'14)

Frequently opening rows  $n+1$   
&  $n-1$  cause **bit flips** in row  $n$

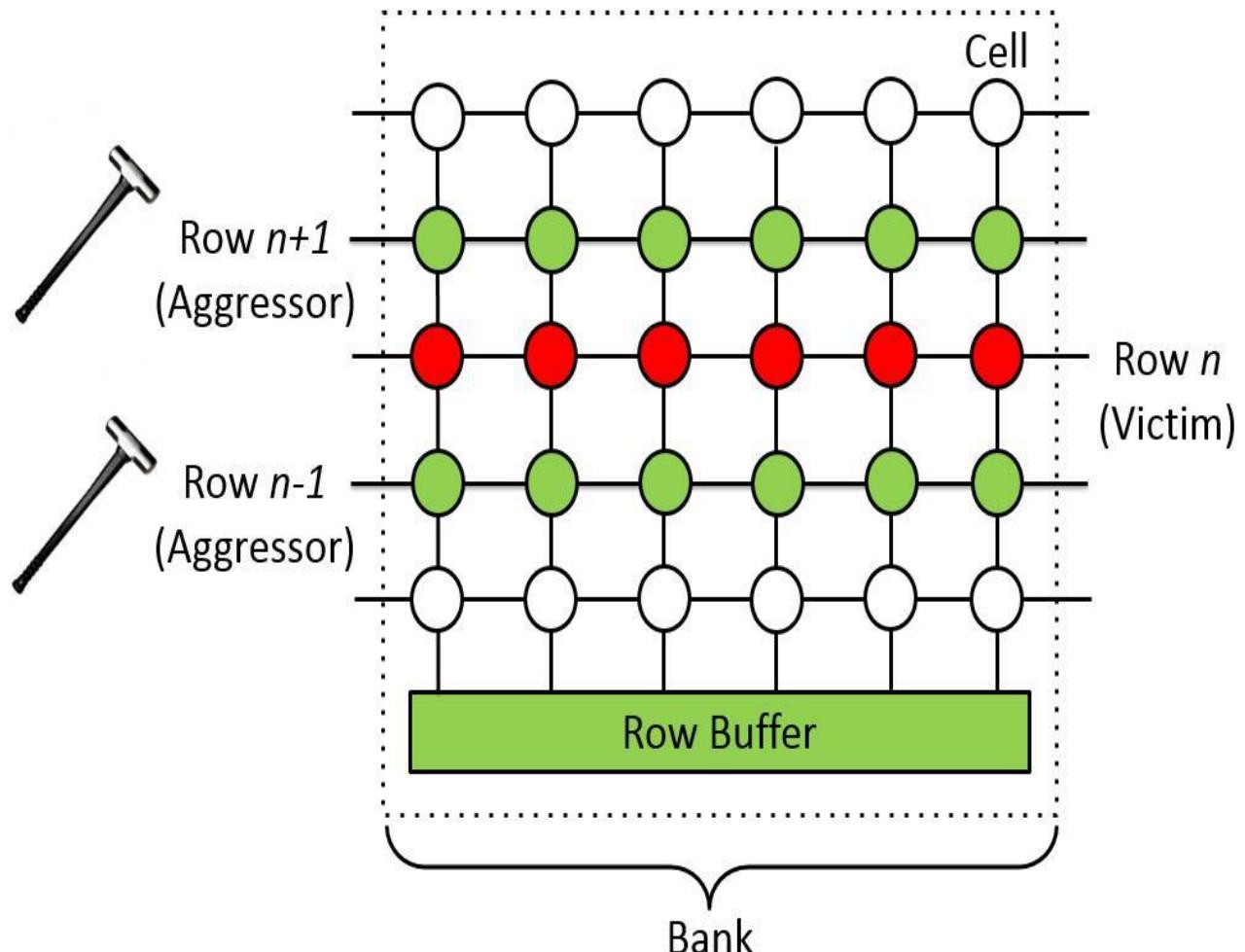


# Rowhammer bug

Kim et al. *key observation* (ISCA'14)

Motivated by Kim et al.,

Seaborn et al. can **compromise**  
**a bug-free kernel** (Blackhat'15)



# Rowhammer code

Code from Kim et al.

- ❖ Flush/Evict CPU cache (e.g., clflush)
  - Clear row buffer (e.g., alternate access)
- 1 loop:
- 2 mov (X), %eax
  - 3 mov (Y), %ebx
  - ❖ 4 clflush (X)
  - ❖ 5 clflush (Y)
  - 6 mfence
  - 7 jmp loop

# Rowhammer code

Code from Kim et al.

- ❖ Flush/Evict CPU cache (e.g., clflush)
- Clear row buffer (e.g., alternate access)

1 loop:

- 2 mov (X), %eax
- 3 mov (Y), %ebx
- ❖ 4 clflush (X)
- ❖ 5 clflush (Y)
- 6 ~~mfence~~
- 7 jmp loop

# Existing rowhammer methods

- **Double-sided hammer**: hammer two rows on each side of the target row
- **Single-sided hammer**: randomly select multiple addresses and hammer them
- **One-location hammer**: randomly select one address and hammer it

# Existing rowhammer methods

1 loop:

- 2        mov (X), %eax
- 3        mov (Y), %ebx
- 4        clflush (X)
- 5        clflush (Y)
- 6        jmp loop

## Double-sided hammer

- X and Y are adjacent to the target row

## Single-sided hammer

- Either X or Y is adjacent to the target row

## Common

- X and Y must be in the same bank to clear row buffer

# Existing rowhammer methods

1 loop:

```
2      mov (X), %eax  
3      mov (Y), %ebx  
4      clflush (X)  
5      clflush (Y)  
6      jmp loop
```

## One-location hammer

- X is adjacent to the target row
- DRAM controller actively clears the row buffer



## Previous rowhammer defenses

# Hardware defenses

Increase row-refreshing frequency

- frequency is not high enough

Introduce Error correcting code (ECC) memory

- multiple bit flips occur

Apply probabilistic adjacent row activation (PARA)

Utilize a target row refresh (TRR) capability

Specify maximum Activation Count (MAC)

- hardware changes

## Ad-hoc attempts

- limited to preventing specific rowhammer attacks

## General solutions

- analysed rowhammer-based binary (Irazoqui et al.)
- blacklisted vulnerable memory (Brasser et al.)
- utilized performance counters (Aweke et al.)



The most<sup>\*</sup> advanced  
rowhammer defense: CATT

\* At the time of our submission



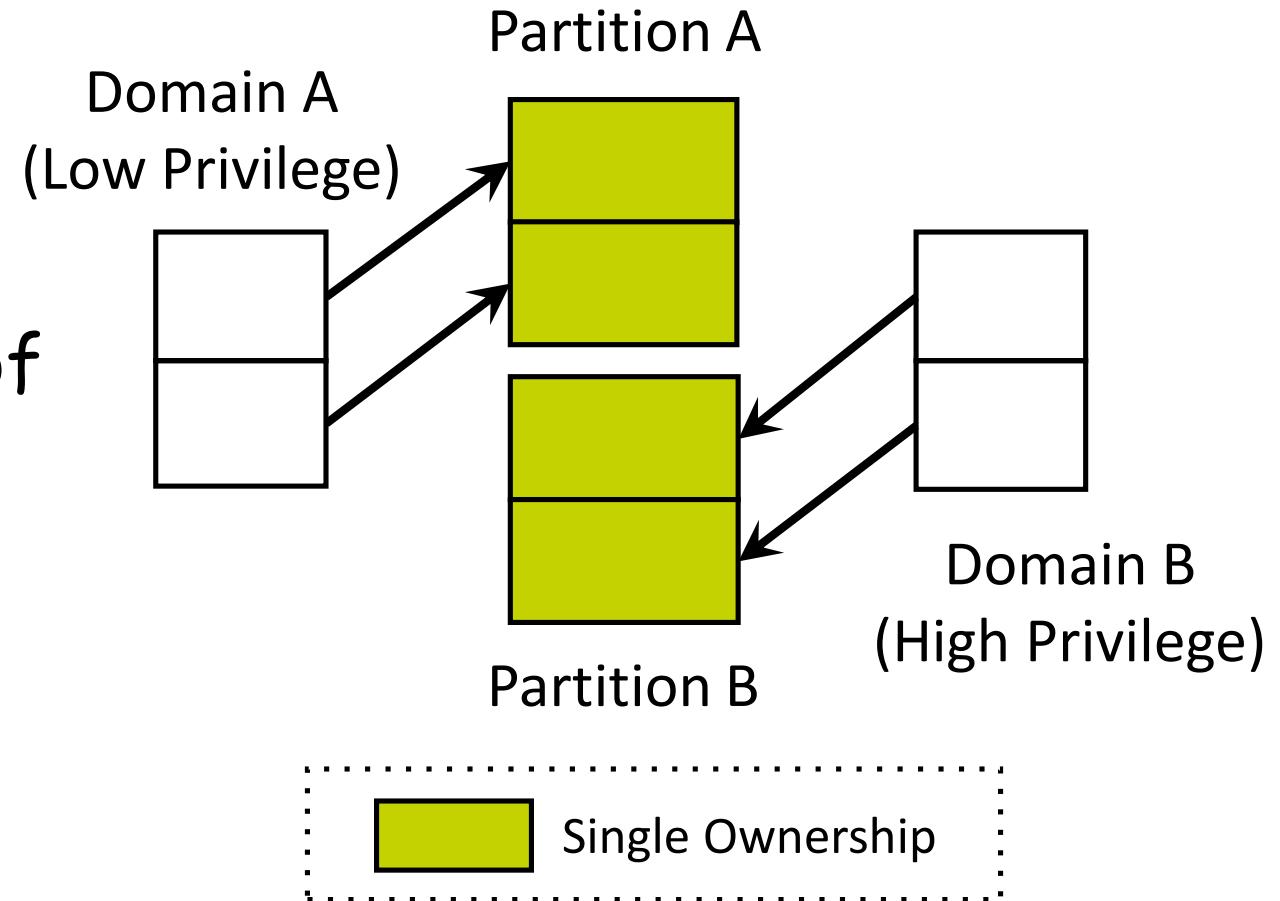
## CAnt Touch This (CATT)

- The island is split into two halves
- One half is physically isolated from the other

# To be specific

Fundamental dictation

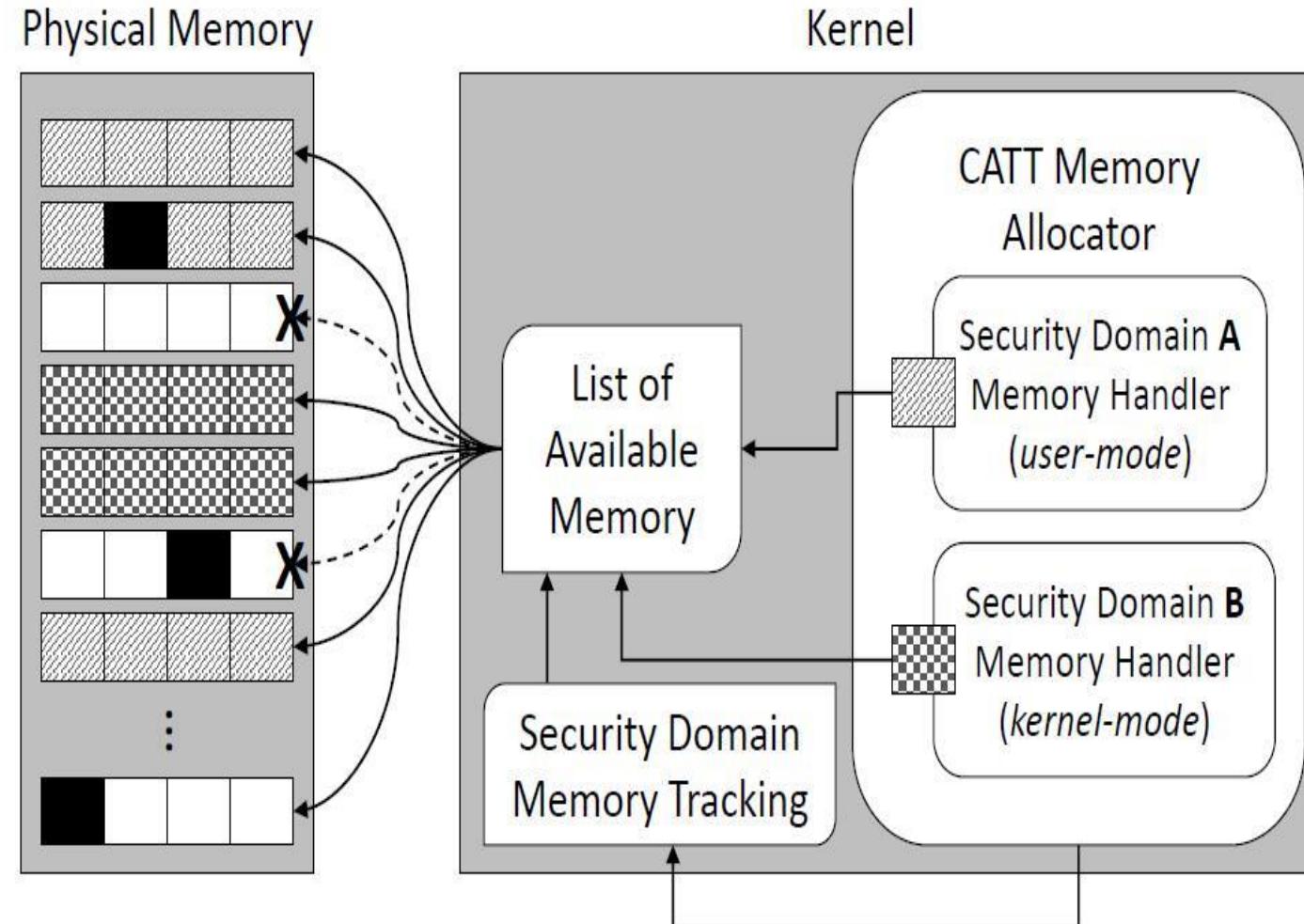
CATT views the ownership of each physical partition as *single*



# CATT's implementation

Diagram from Brasser et al.

Physical Isolation



# CATT's advantages

Picture from [www.wallpapermania.eu](http://www.wallpapermania.eu)

## High practicality

- lightweight kernel patch
- neglectable performance overhead

## High effectiveness

- mitigate previous rowhammer attacks

# CATT's advantages

Picture from [www.wallpapermania.eu](http://www.wallpapermania.eu)

## High practicality

- lightweight kernel patch
- neglectable performance overhead

## High effectiveness

- mitigate previous rowhammer attacks



# Summary of a rowhammer exploit

- ❖ Fill up a victim row with sensitive data structures
- Position attacker-accessible rows adjacent to the victim row
- ❖ Rowhammer

# Review of defenses

- ❖ Fill up a victim row with sensitive data structures
- ❖ Rowhammer



Previous rowhammer defenses stop

# Review of defenses

- ❖ Fill up a victim row with sensitive data structures



- ❖ Rowhammer

Previous rowhammer defenses stop

- ❑ Position attacker-accessible rows adjacent to the victim row



CATT (i.e., physical isolation) stops

# Threat model & Assumptions

Bug-free kernel +

Enabled CATT +

Unknown victim-row locations +

Disabled pagemap

# Threat model & Assumptions

No way to  
rowhammer kernel?



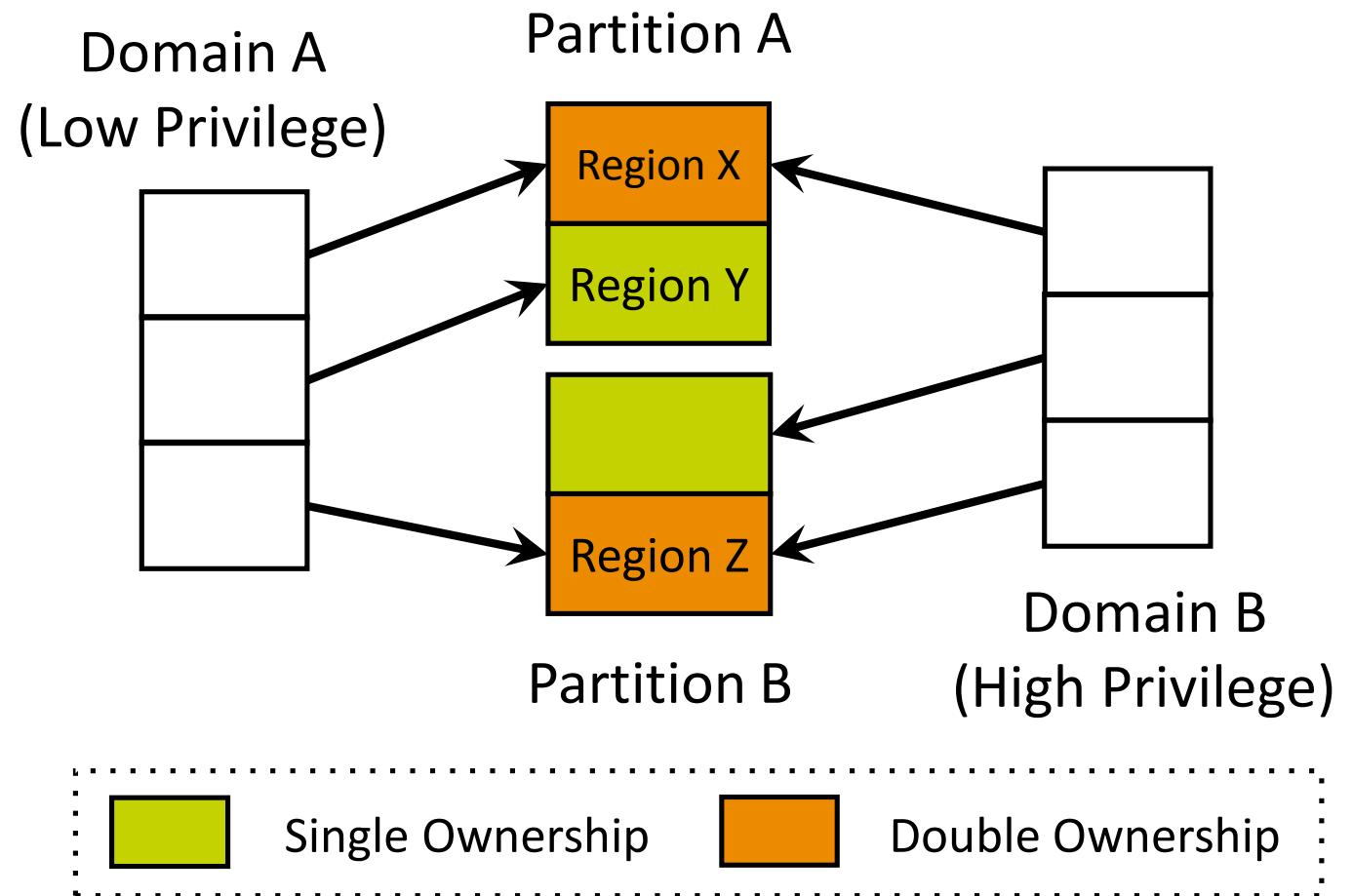
Bug-free kernel +  
Enabled CATT +  
Unknown victim-row locations +  
Disabled pagemap

# Key steps & Challenges

- 000. Clearly identify CATT's weakness
- 001. Stealthily position attacker-accessible memory adjacent to kernel objects
- 010. Efficiently perform hammering
  
- 011. Verify whether “exploitable” bit flips have occurred (Seaborn et al.)
- 100. If yes, kernel privilege ✓
  - root privilege (scan kernel memory, flush tlb and change uid to 0) ✓

# Clearly identify CATT's weakness

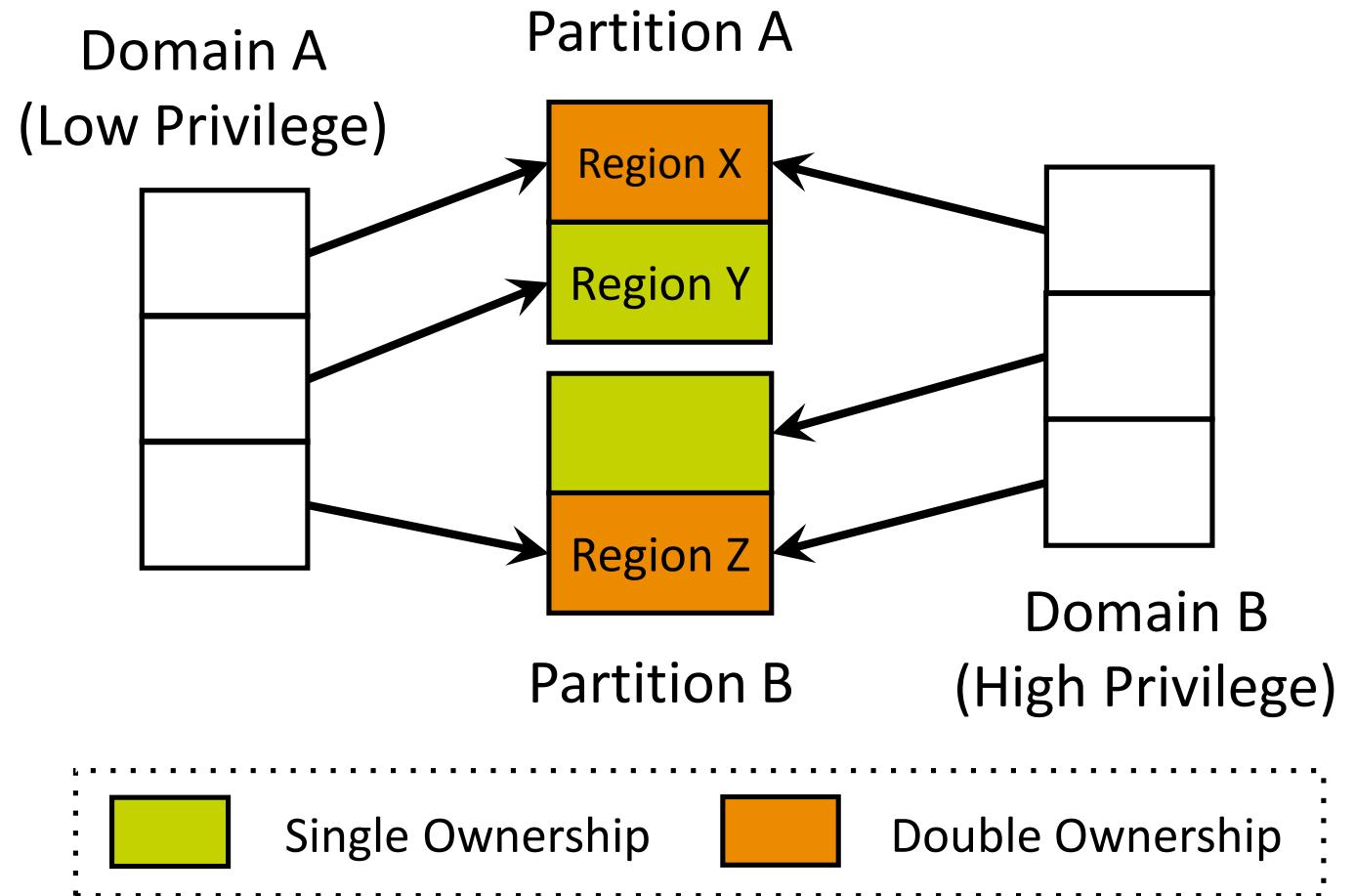
In theory, **single ownership**  
does hold



# Clearly identify CATT's weakness

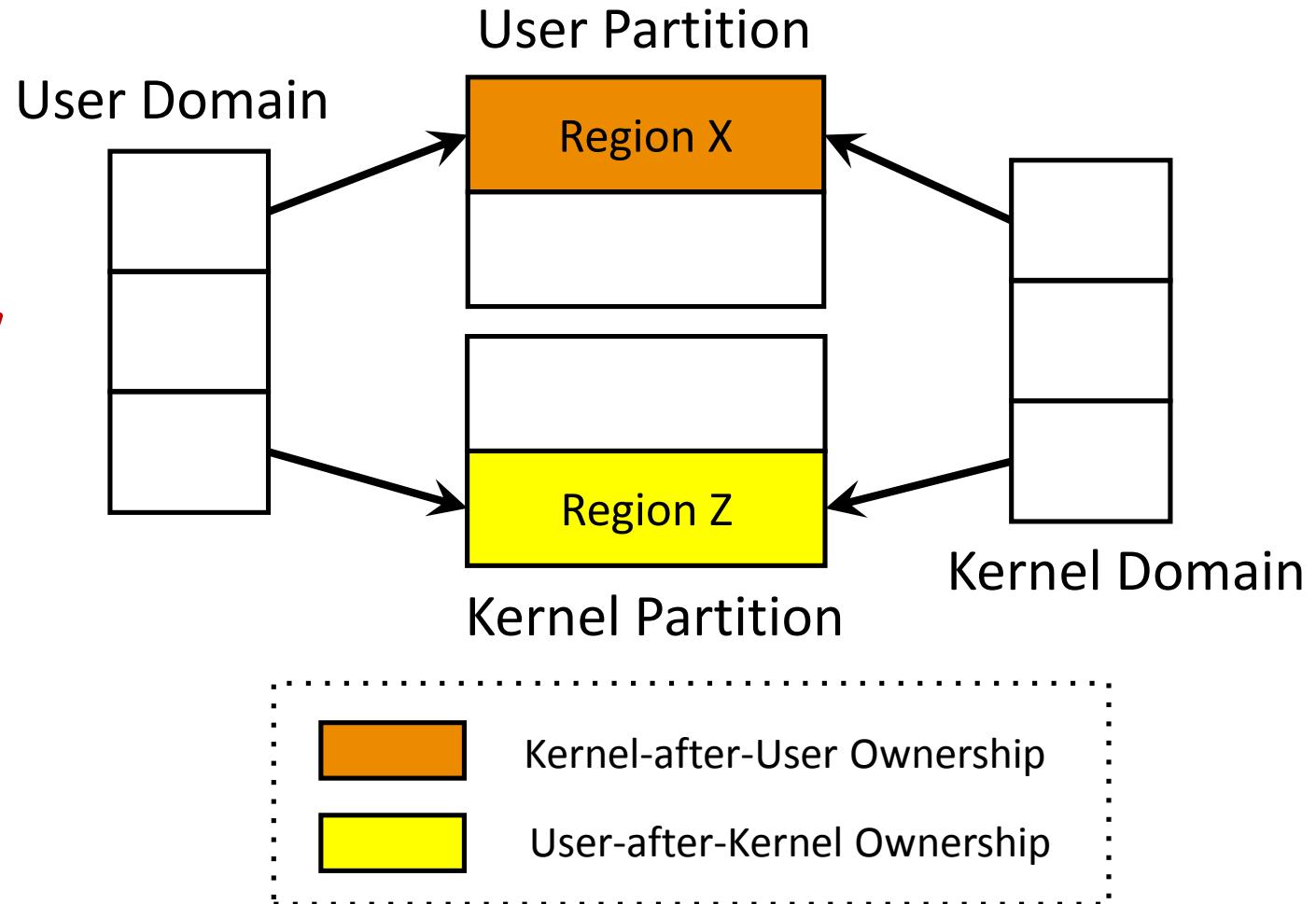
In theory, **single ownership**  
does hold

However, in real-world,  
modern systems view the  
ownership as **dynamic**



# Clearly identify CATT's weakness

Region Z is “hammerable”



- Challenge
  - Identify “**hammerable**” double-owned memory
- Properties
  - Hammerable memory is **initially owned** by the kernel
  - The user can **access** the memory

# Clearly identify CATT's weakness

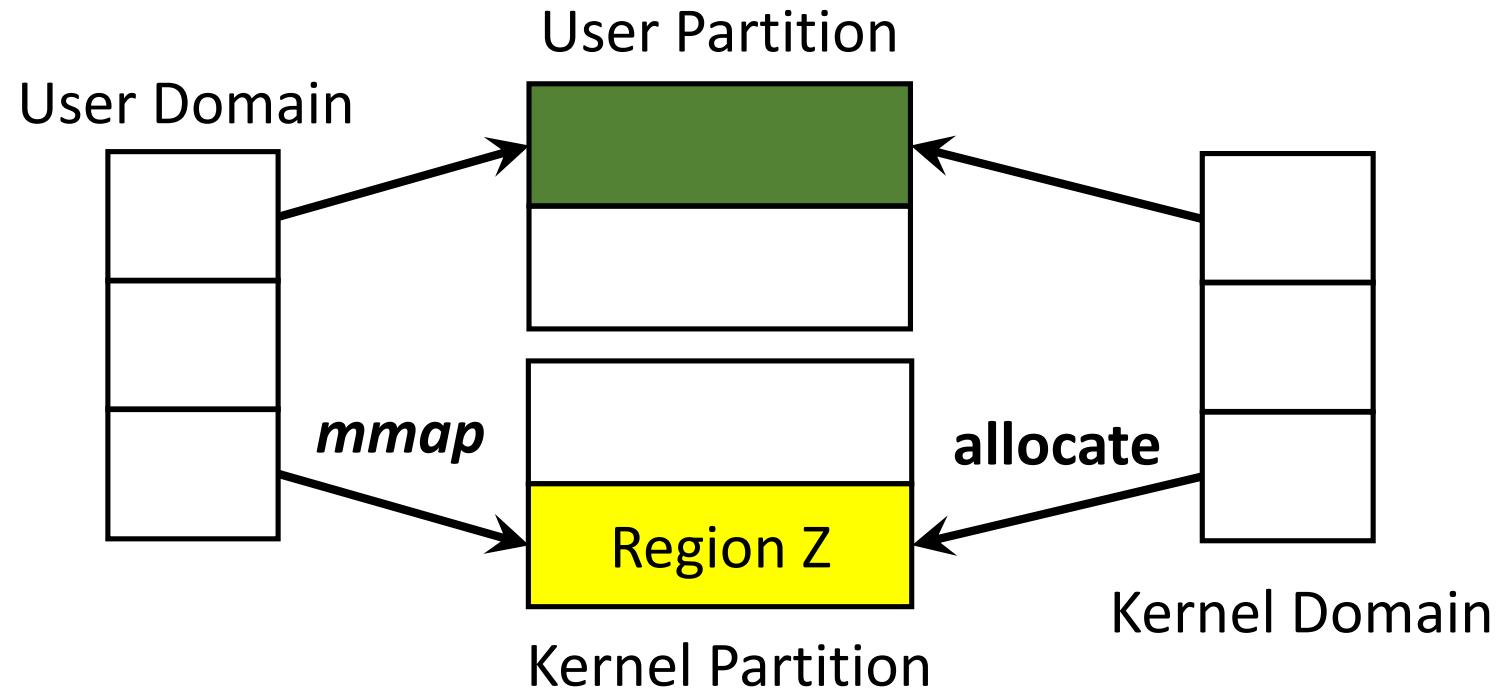


*For the split islands*

*Bridge interface* is introduced to  
counteract the physical isolation

# Clearly identify CATT's weakness

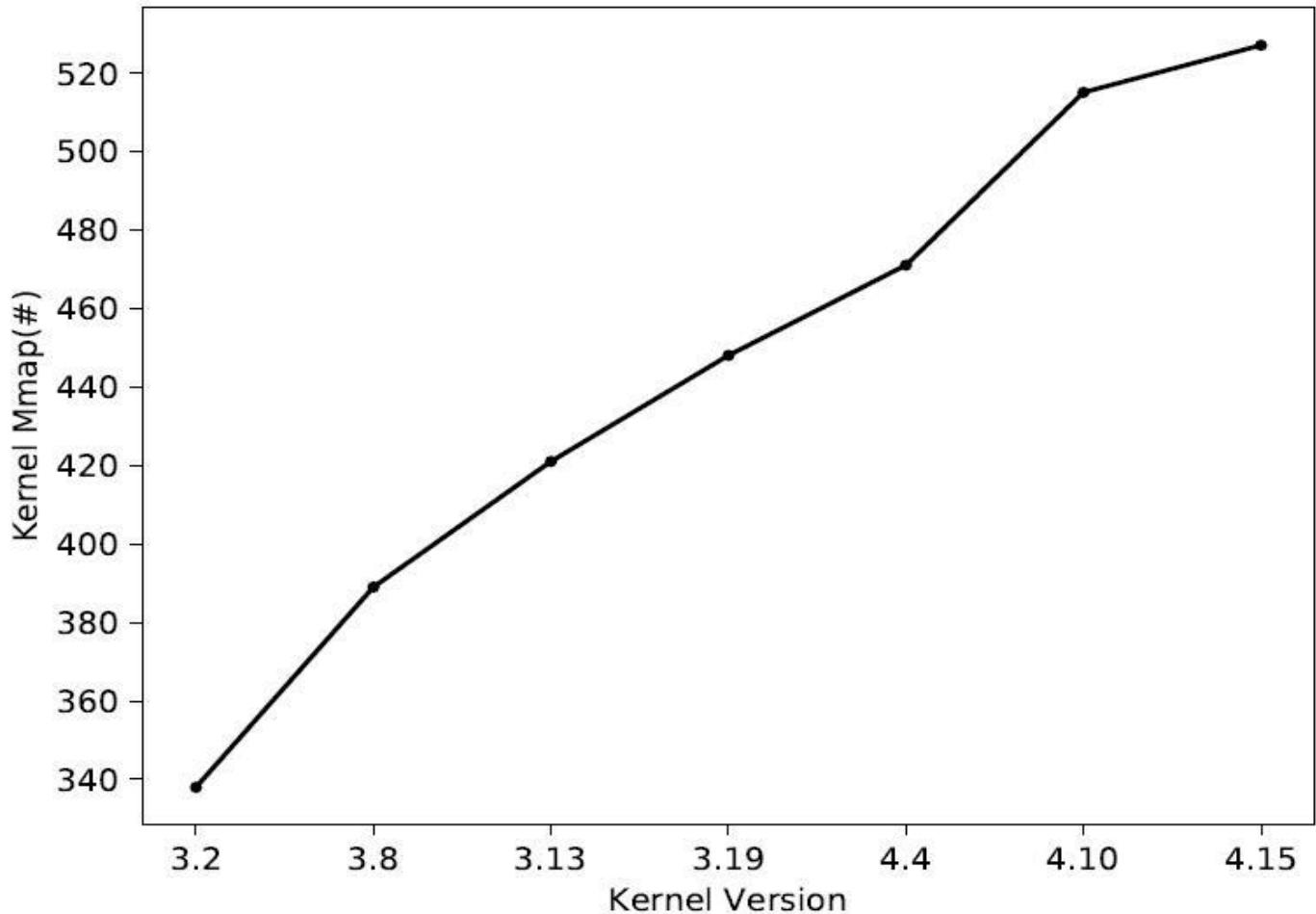
*For the split partitions*  
**mmap interface** can  
be abused to hammer the  
kernel partitions



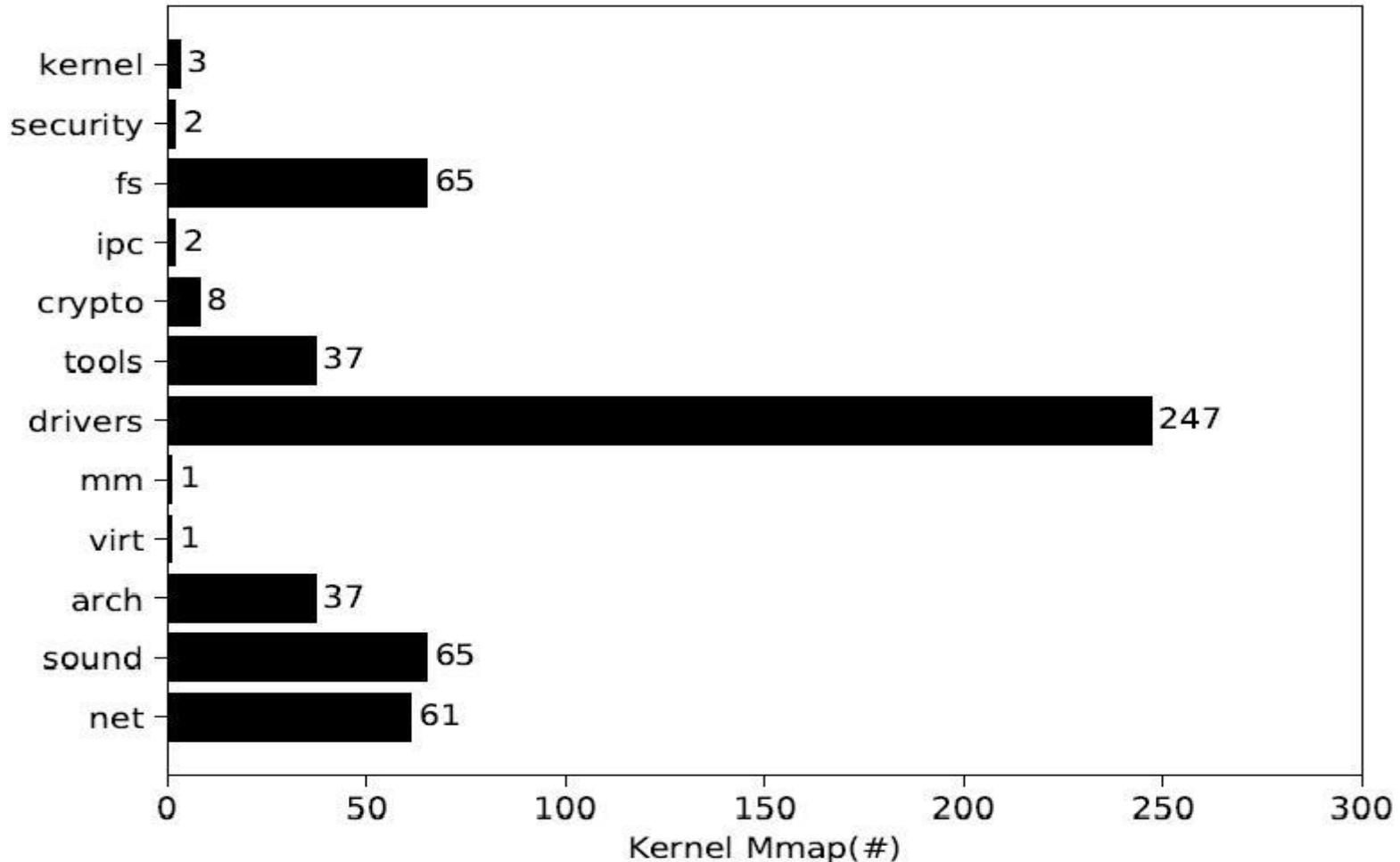
User-after-Kernel Ownership

# Clearly identify CATT's weakness

The number of kernel *mmap* operations **increases significantly** as Linux kernel evolves.



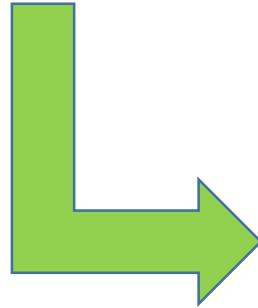
## *mmap* distribution (Linux kernel 4.17)



# Clearly identify CATT's weakness

## Identify hammerable buffer

Identify hammerable buffer



**SCSI Generic buffer** in the Linux SCSI subsystem

# Stealthily position hammerable buffer and kernel page table

Double-owned buffer makes it possible  
to rowhammer the kernel again

# Stealthily position hammerable buffer and kernel page table

Double-owned buffer makes it possible  
to rowhammer the kernel again

Next, how to make the double-owned hammerable buffer exploitable



How to stealthily position the hammerable buffer  
next to page tables

# Stealthily position hammerable buffer and kernel page table

Double-owned buffer makes it possible  
to rowhammer the kernel again

Next, how to make the double-owned hammerable buffer exploitable



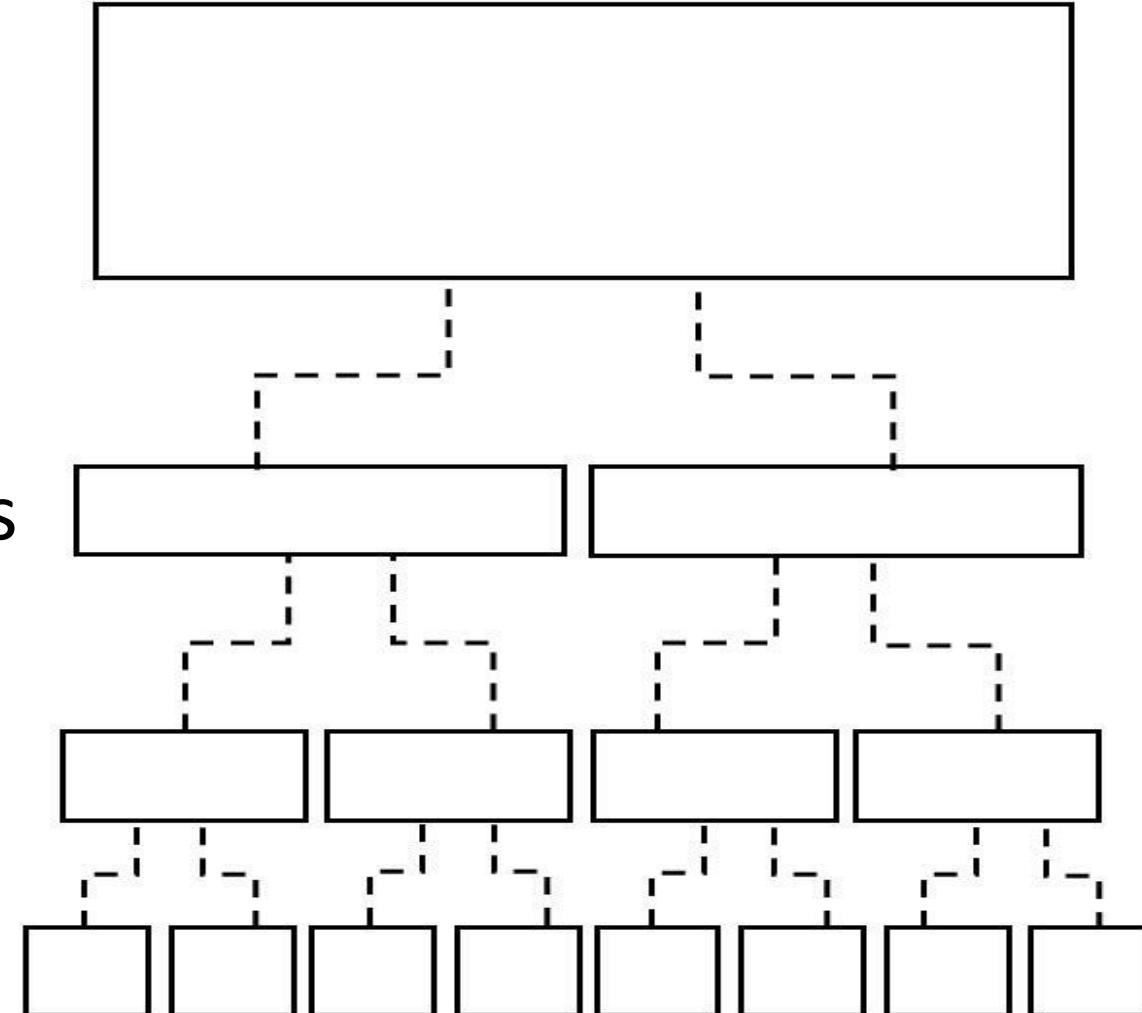
How to stealthily position the hammerable buffer  
next to page tables



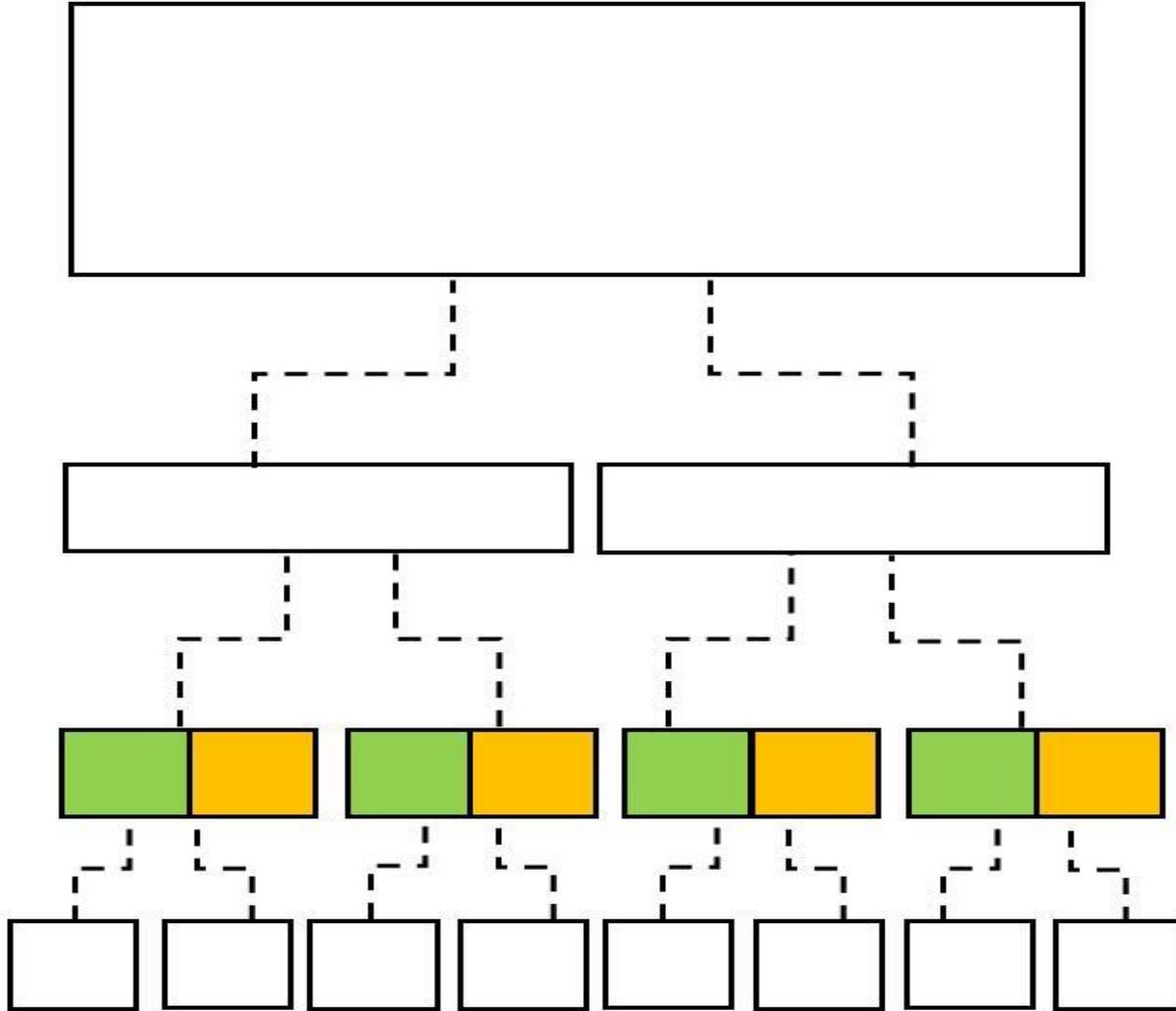
A new technique (memory ambush)

## Linux buddy allocator

A list of blocks



# Memory ambush



If they share the same block

Are they adjacent to each other?

# Memory ambush



Two adjacent physical addresses **do not imply** two adjacent dram rows

Say: 0x1000 and 0xFFFF are in the same row

# Memory ambush



Two adjacent physical addresses **do not imply** two adjacent dram rows

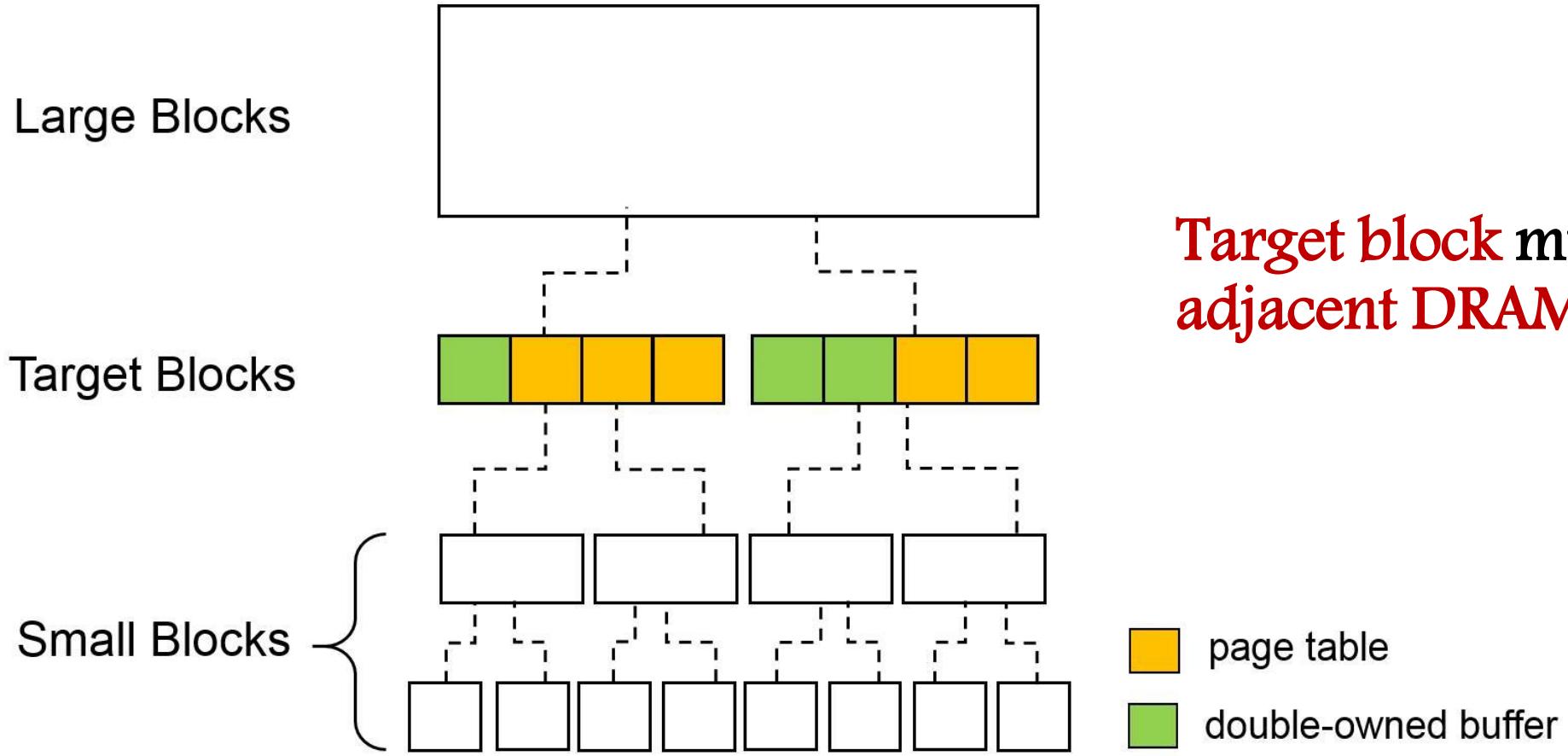
Say: 0x1000 and 0xFFFF are in the same row



Two **row-aligned** adjacent physical addresses **indicate** two **adjacent** dram rows (Xiao et al.)

Say: **b<sub>18</sub>** to **b<sub>32</sub>** on Sandy Bridge, **b<sub>18</sub>** to **b<sub>31</sub>** on Ivy Bridge, **b<sub>23</sub>** to **b<sub>34</sub>** on Haswell are row indexes

# Memory ambush



# Memory ambush

## How to calculate target block size

$$\text{TargetBlockSize} = \text{RowsSizePerRowIndex} \bullet 2$$
$$\text{RowsSizePerRowIndex} = \text{DIMMs} \bullet \text{BanksPerDIMM RowSize}$$
$$\text{BanksPerDIMM} = \text{BanksPerRank} \bullet \text{RanksPerDIMM}$$

# Memory ambush



**How** to place more page-tables  
on the target blocks

# Memory ambush

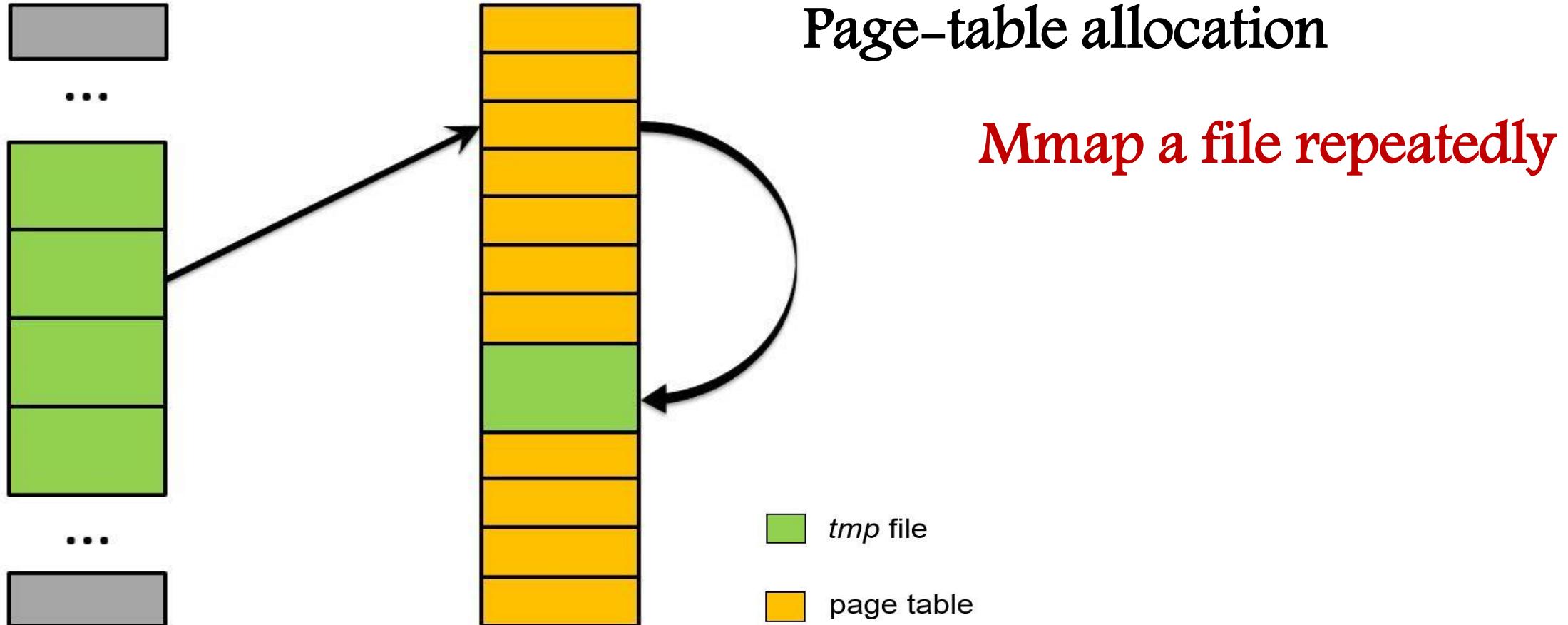


**How** to place more page-tables  
on the target blocks

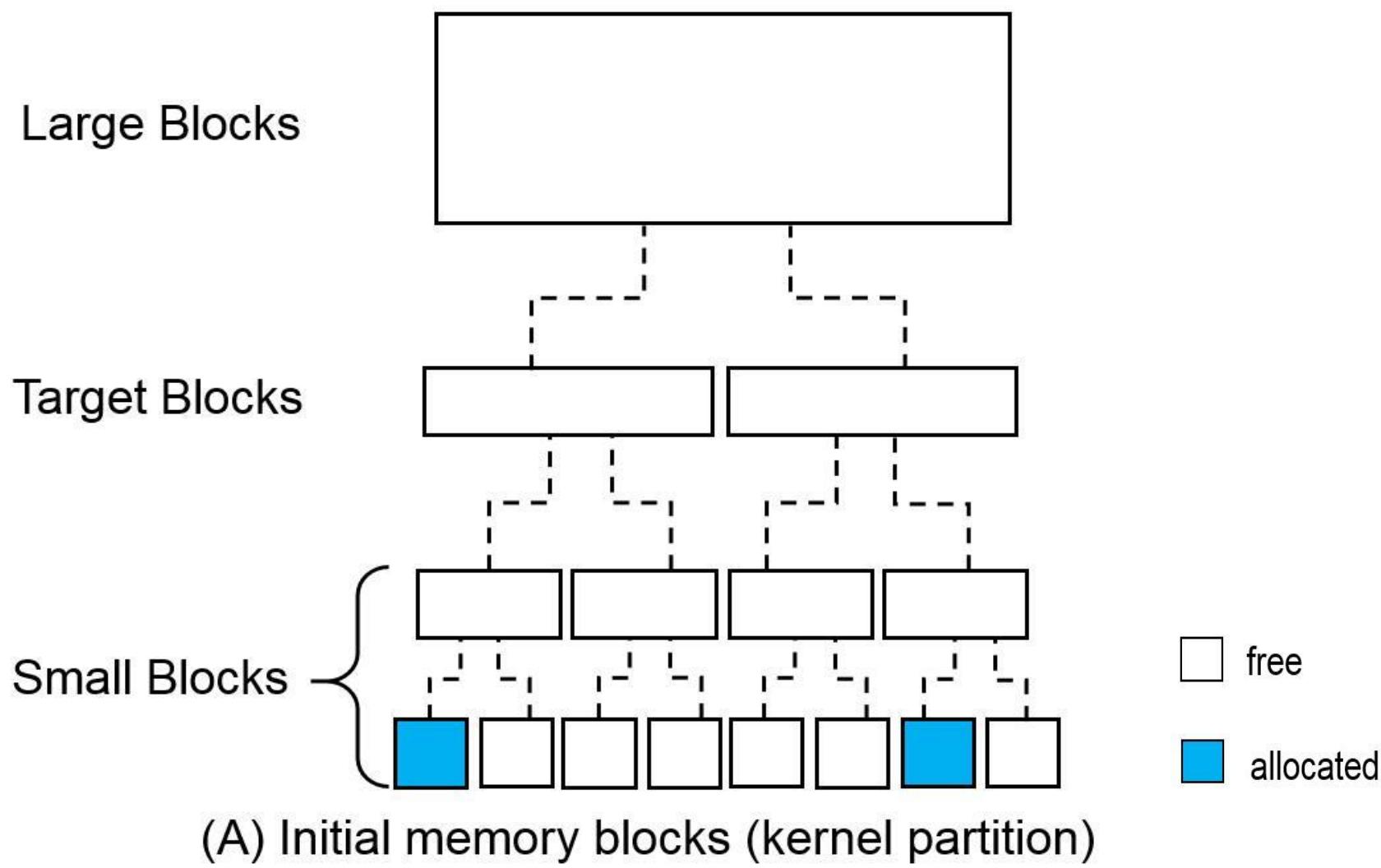


Abuse ***mmap*** to do page-table allocation (Seaborn et al.)

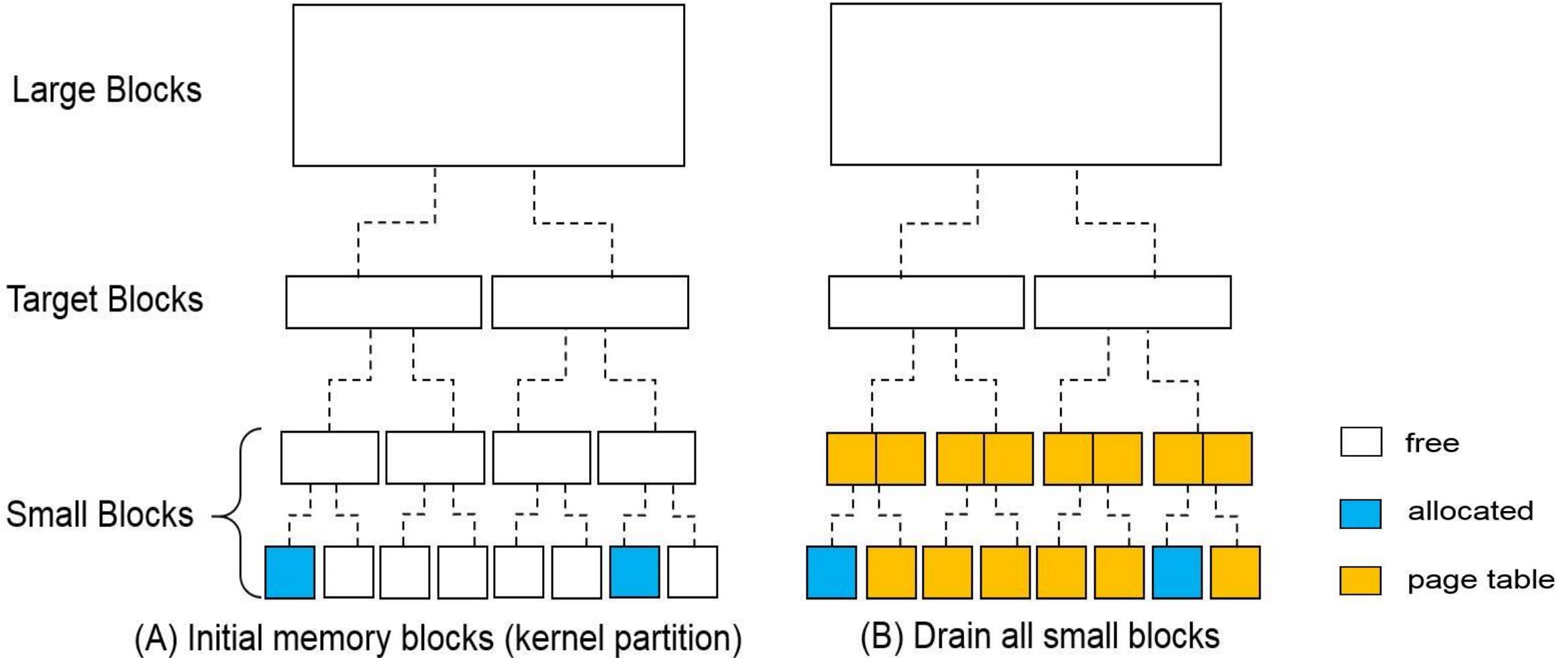
# Memory ambush



# Memory ambush

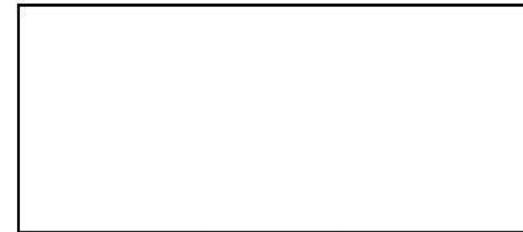


# Memory ambush

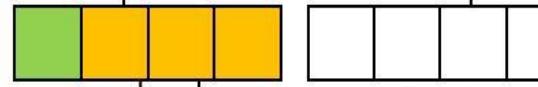


# Memory ambush

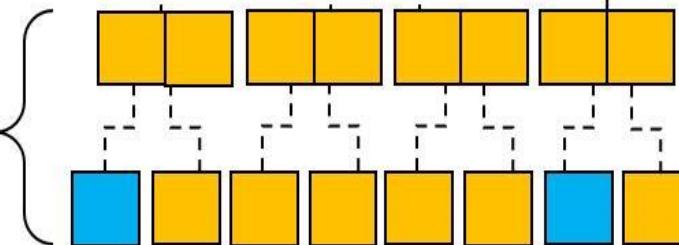
Large Blocks



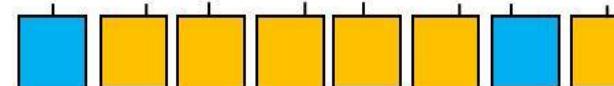
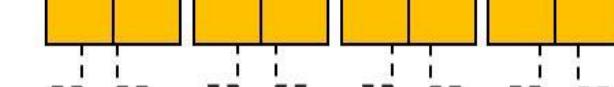
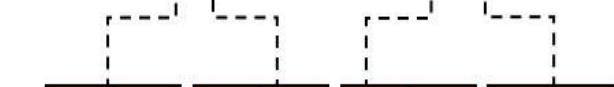
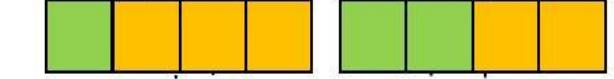
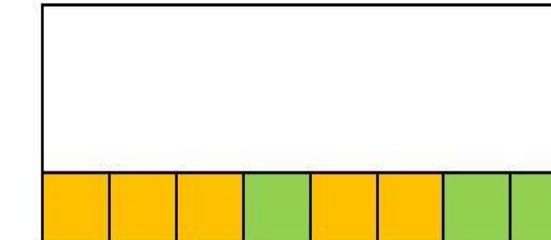
Target Blocks



Small Blocks



(C) Allocate double-owned buffers and page tables from target blocks



(D) Stop until a specified memory threshold is reached

-  free
-  allocated
-  double-owned buffer
-  page table

# Efficiently single-sided rowhammer



No *pagemaps* since Linux 4.0

# Efficiently single-sided rowhammer



No *pagemaps* since Linux 4.0



A **timing channel** (Moscibroda et al.)

Diagram from Kim et al.

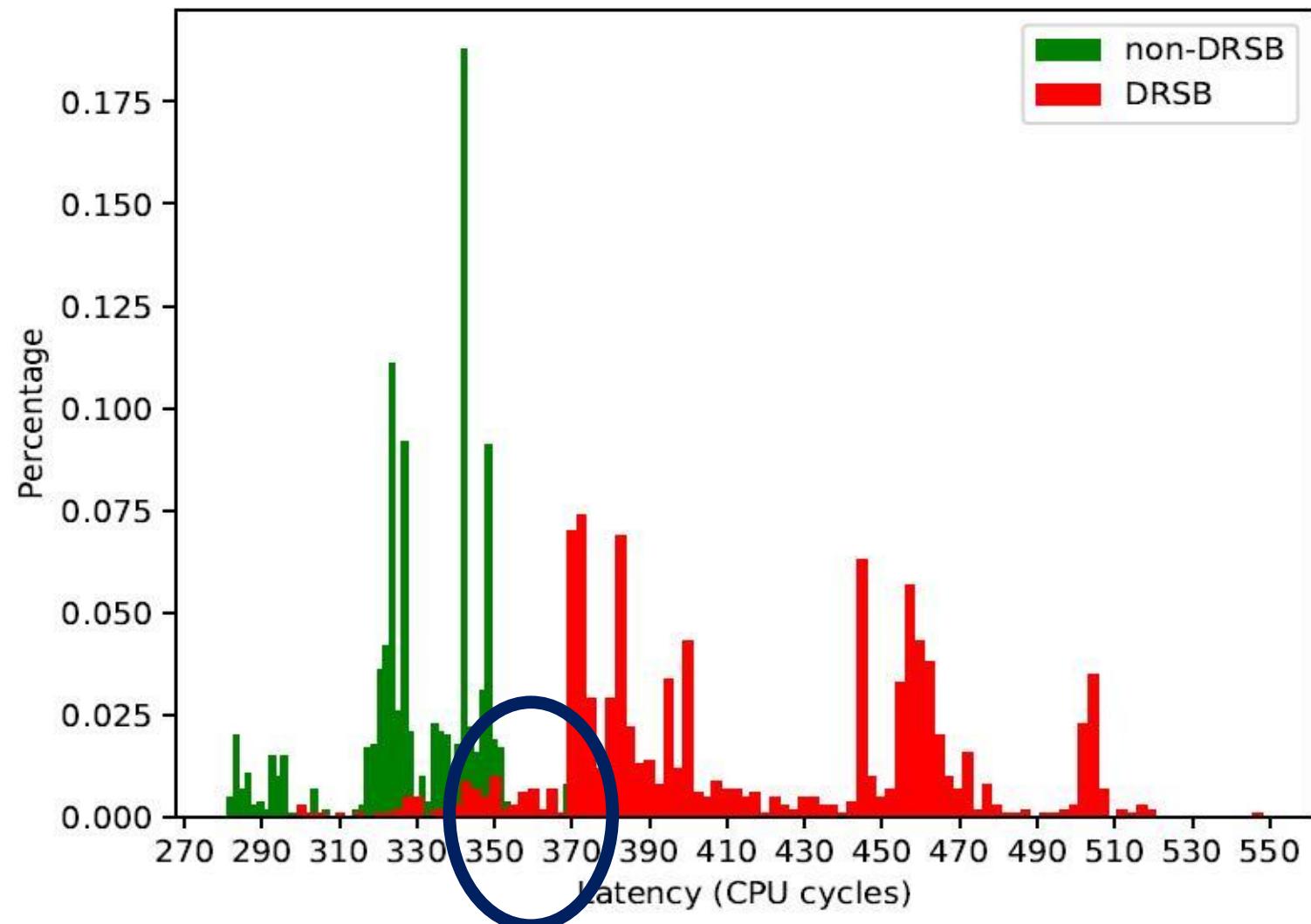
## A timing channel

- X and Y are in different rows of same bank (i.e., DRSB), causing *row conflict*
- *row conflict* leads to higher memory-access latency

1 loop:

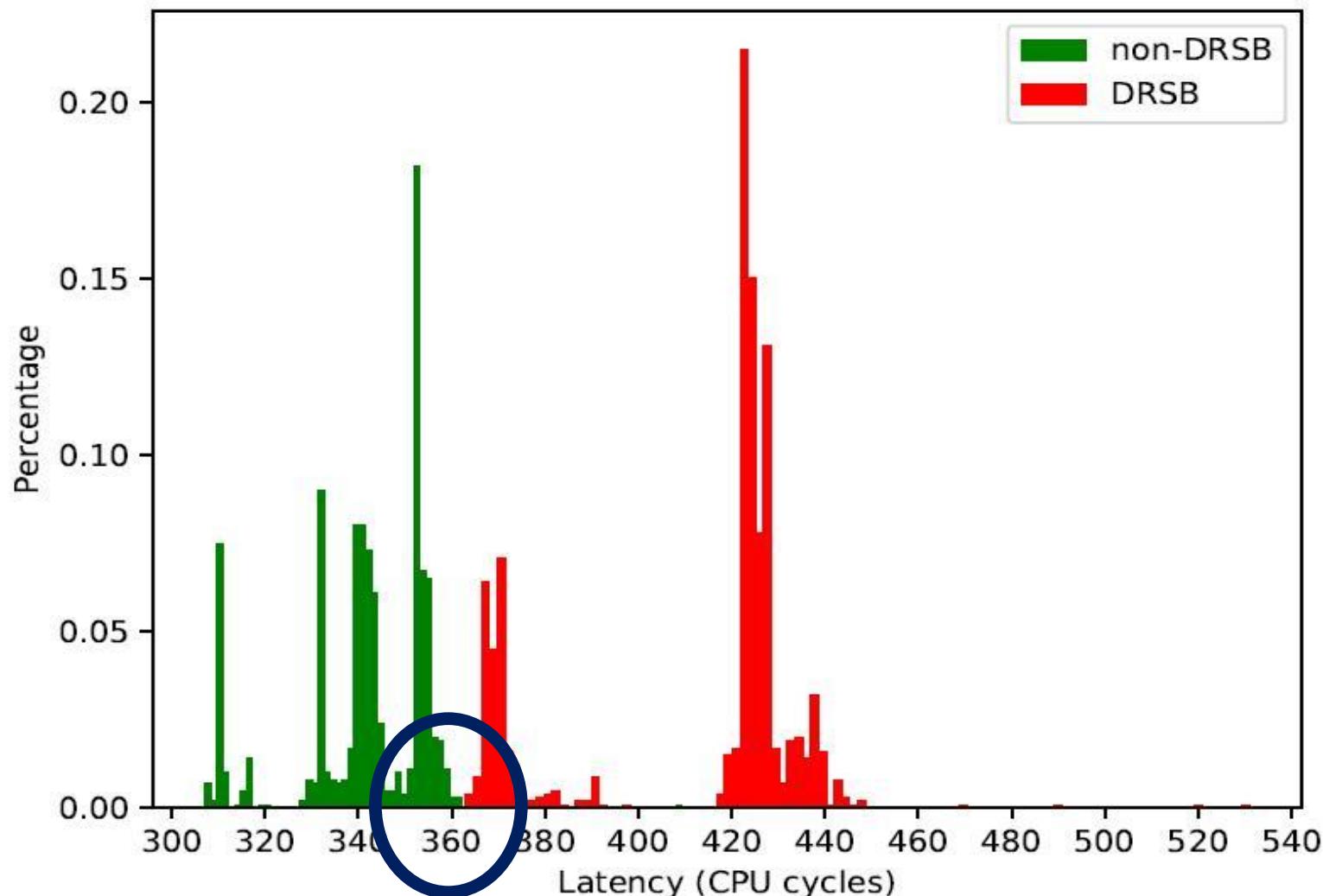
- 2      mov (X), %eax
- 3      mov (Y), %ebx
- 4      clflush (X)
- 5      clflush (Y)
- 6      jmp loop

# Efficiently single-sided rowhammer



DRSB: different row same bank

# Efficiently single-sided rowhammer



DRSB: different row same bank

- 000. Clearly identify CATT's weakness
- 001. Stealthily position attacker-accessible memory adjacent to kernel objects
- 010. Efficiently perform hammering
  
- 011. Verify whether “exploitable” bit flips have occurred (Seaborn et al.)
- 100. If yes, kernel privilege and then root privilege ✓

Picture from [https://cartooncharacters.fandom.com/wiki/Jerry\\_Mouse](https://cartooncharacters.fandom.com/wiki/Jerry_Mouse)



**Kernel Privilege Escalation**

**Root Privilege Escalation**



# SCSI Generic buffer is not alone

Disabling the SCSI Generic driver is **enough?**

# SCSI Generic buffer is not alone

Disabling the SCSI Generic driver is **enough?**

Far from enough

**Video buffer** in Video4Linux subsystem is also **exploitable**

Please help yourself

- ✓ /dev
- ✓ /sys
- ✓ ...

# Mitigations

- Allocate double-owned buffer from user partition
- Separate data with two guarding rows (Konoth et al., published after our submission)
- Protect Page Tables from being rowhammered (Wu et al., published after our submission)

# Mitigations

## Weaknesses

- Allocate double-owned buffer from user partition  
**expose sensitive buffer to user space**
- Separate data with two guarding rows (Konoth et al., published after our submission)
- Protect Page Tables from being rowhammered (Wu et al., published after our submission)

# Mitigations

## Weaknesses

- Allocate double-owned buffer from user partition  
*expose sensitive buffer to user*
- Separate data with two guarding rows (Konoth et al., published after our submission)  
*suffer from dram row remapping & bit flips in multiple rows*
- Protect Page Tables from being rowhammered (Wu et al., published after our submission)

# Mitigations

## Weaknesses

- Allocate double-owned buffer from user partition  
*expose sensitive buffer to user*
- Separate data with two guarding rows (Konoth et al., published after our submission)  
*suffer from dram row remapping & bit flips in multiple rows*
- Protect Page Tables from being rowhammered (Wu et al., published after our submission)  
*suffer from exploitable bit flips*

# Mitigations

## Weaknesses

- Allocate double-owned buffer from user partition  
**expose sensitive buffer to user**
- Separate data with two guarding rows (Konoth et al., published after our submission)  
**suffer from dram row remapping & bit flips in multiple rows**
- Protect Page Tables from being rowhammered (Wu et al., published after our submission)  
**suffer from exploitable bit flips**

## Ongoing work

**Can we break all of them ???  
Follow up our next rowhammer talk please ☺**

# Black Hat Sound Bytes

- Physical isolation of different security domains is powerful against current rowhammer attacks.
- Physical isolation is hard to achieve in practice due to double-owned buffers, which make rowhammer bug still exploitable.
- The double-owned buffers are used for the sake of performance and functionality and thus it will be challenging to remove them.
- Our exploit is stealthy to gain root/kernel privileges given the presence of physical isolation.



- Yueqiang Cheng ([chengyueqiang@baidu.com](mailto:chengyueqiang@baidu.com))
- Zhi Zhang ([zhi.zhang@data61.csiro.au](mailto:zhi.zhang@data61.csiro.au))
- Surya Nepal ([Surya.Nepal@data61.csiro.au](mailto:Surya.Nepal@data61.csiro.au))
- Zhi Wang ([zwang@cs.fsu.edu](mailto:zwang@cs.fsu.edu))

# References

- Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, “Flipping bits in memory without accessing them: An experimental study of dram disturbance errors,” in ACM SIGARCH Computer Architecture News, 2014.
- M. Seaborn and T. Dullien, “Exploiting the dram rowhammer bug to gain kernel privileges,” in Black Hat’15.
- F. Brasser, L. Davi, D. Gens, C. Liebchen, and A.-R. Sadeghi, “Can’t touch this: Software-only mitigation against rowhammer attacks targeting kernel memory,” in USENIX Security Symposium, 2017.
- K. A. Shutemov, “Pagemap: Do not leak physical addresses to nonprivileged userspace,” <https://lwn.net/Articles/642074/>, 2015.
- T. Moscibroda and O. Mutlu, “Memory performance attacks: Denial of memory service in multi-core systems,” in USENIX Security Symposium, 2007.
- Z. B. Aweke, S. F. Yitbarek, R. Qiao, R. Das, M. Hicks, Y. Oren, and T. Austin, “ANVIL: Software-based protection against next-generation Rowhammer attacks,” in International Conference on Architectural Support for Programming Languages and Operating Systems, 2016.
- X. C. Wu, T. Sherwood, F. T. Chong and Y. J. Li, “ANVIL: Software-based protection against next-generation Rowhammer attacks,” In International Conference on Architectural Support for Programming Languages and Operating Systems, 2019.
- R. K. Konoth, M. Oliverio, A. Tatar, D. Andriesse, H. Bos, C. Giuffrida, and K. Razavi. 2018. ZebRAM: Comprehensive and Compatible Software Protection Against Rowhammer Attacks. In USENIX Symposium on Operating Systems Design and Implementation, 2018.