Exploiting the DRAM row hammer bug to gain kernel privileges

Writer : MARK SEABORN @GOOGLE

Presenter : Jiwon Choi

Introduction

Exploit !

… without exploiting software bug

Row hammer

DRAM's row

repeated accesses

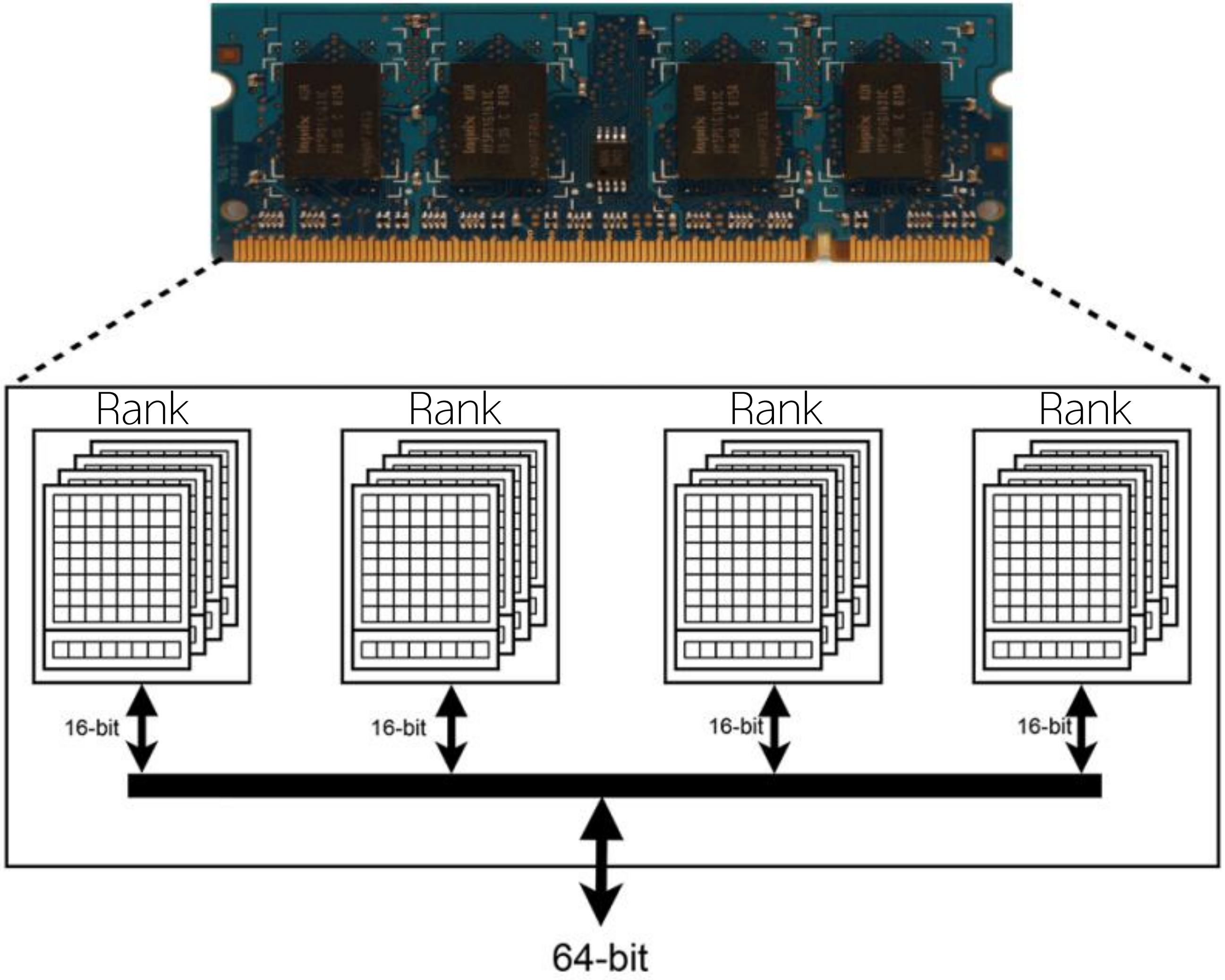
DRAM chipset

DRAM Structure

DRAM chipset

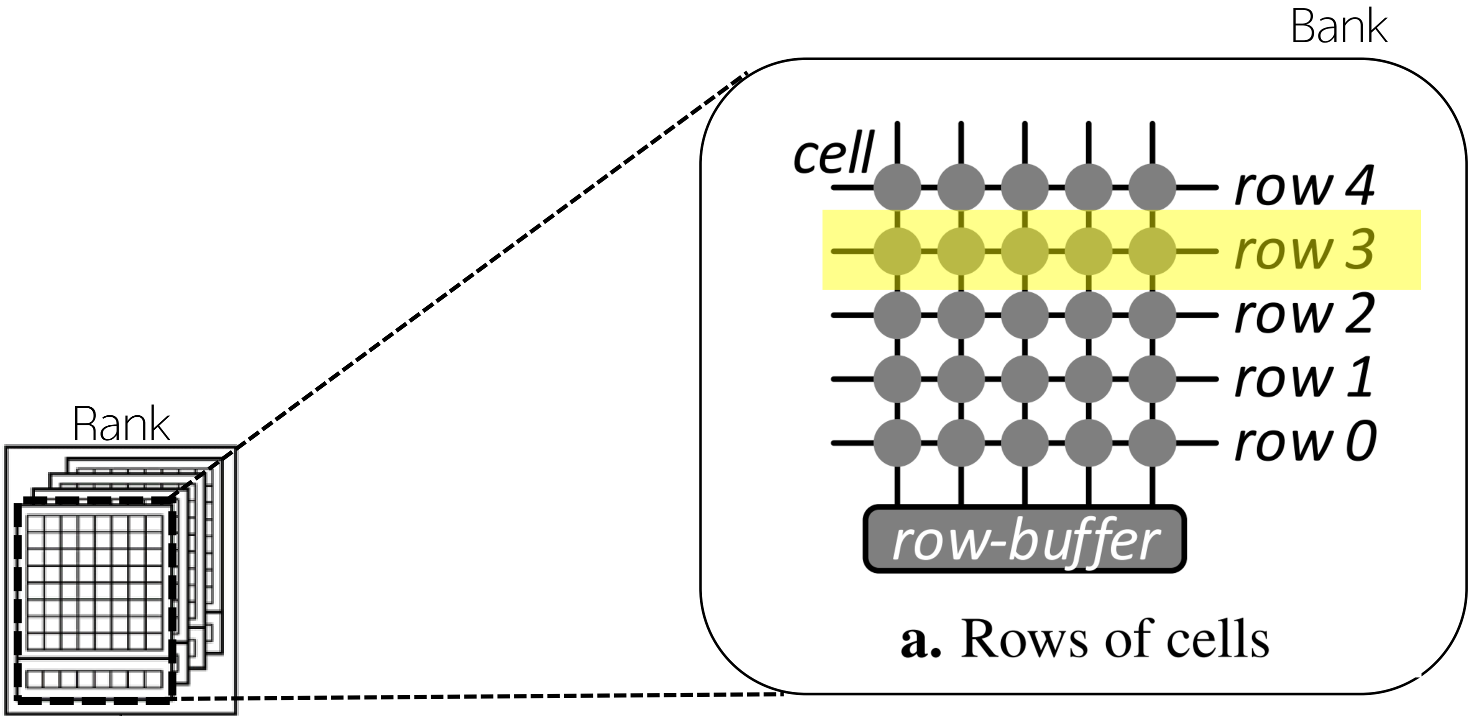
DRAM Structure

DRAM chipset



(Diagram from ARMOR project, University of Manchester)

DRAM Structure

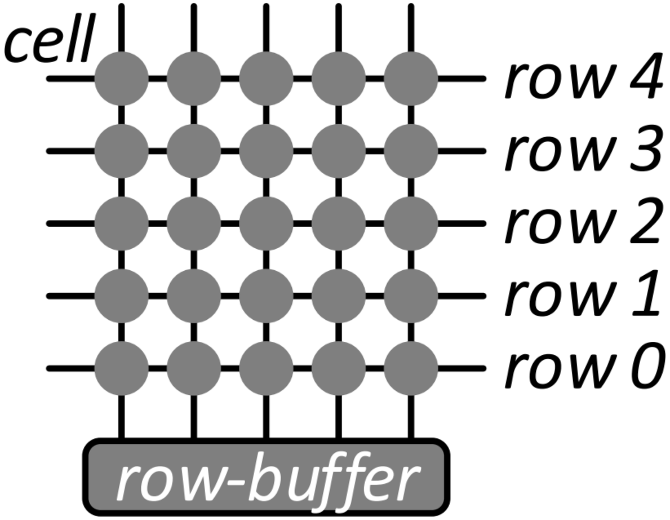


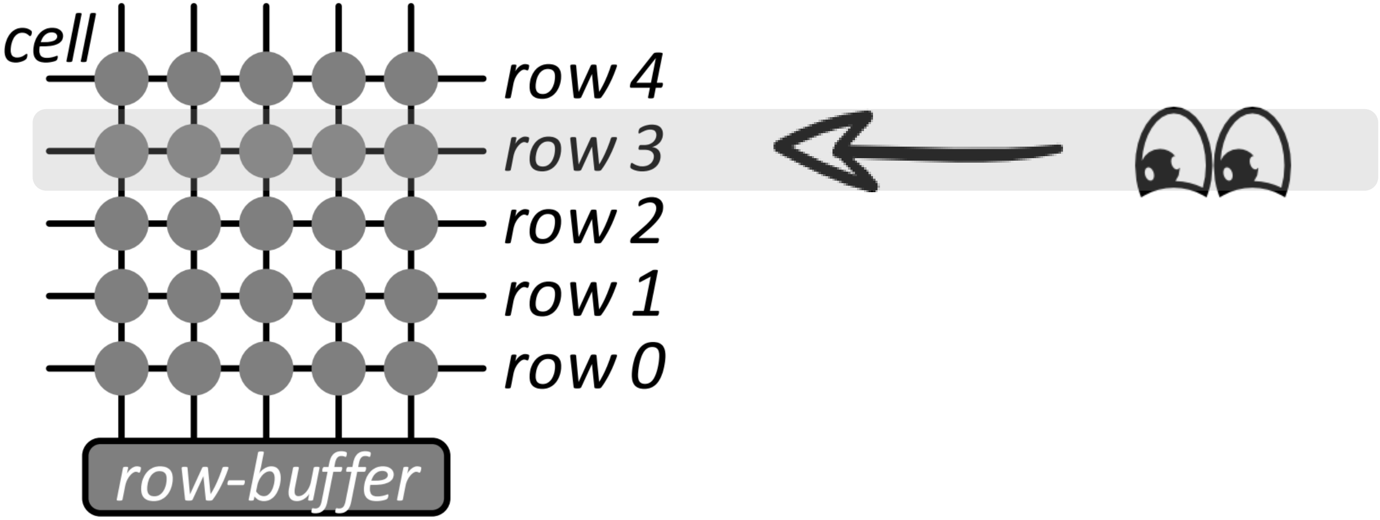
ex) 4GB memory = 2ranks * 8 banks *8K per row * 32768 rows

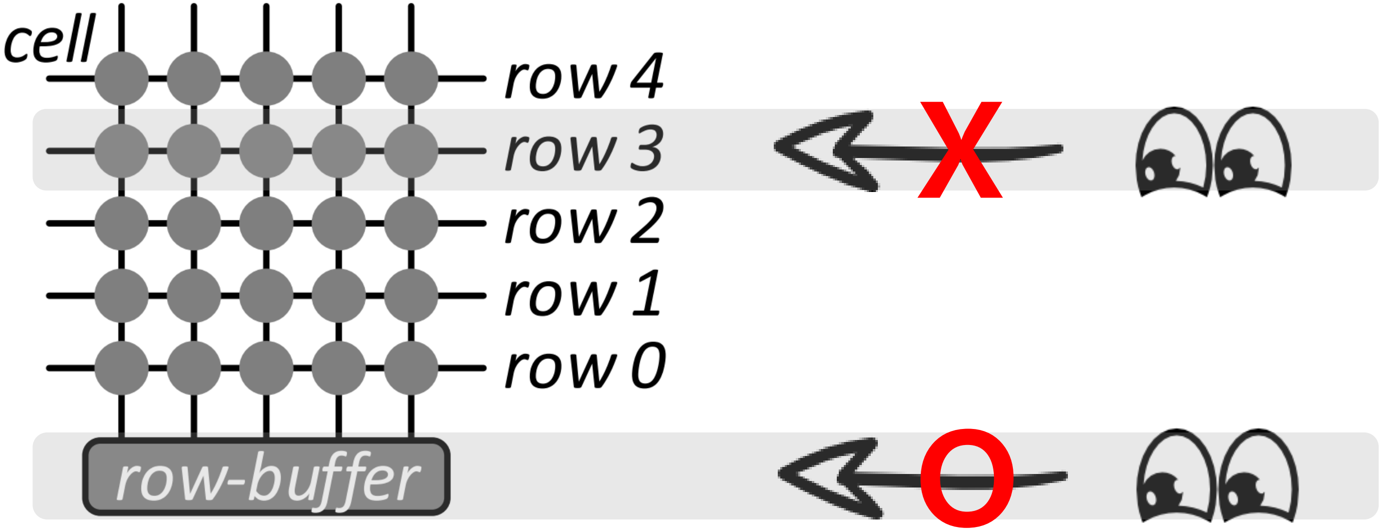
01

DRAM ?
Dynamic RAM !

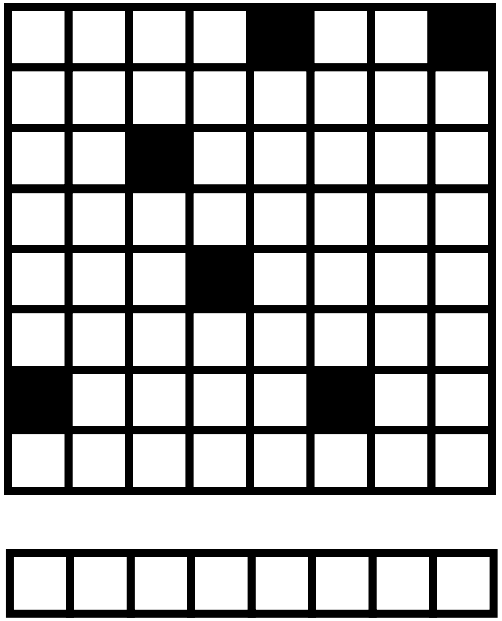
DRAM is really dynamic!



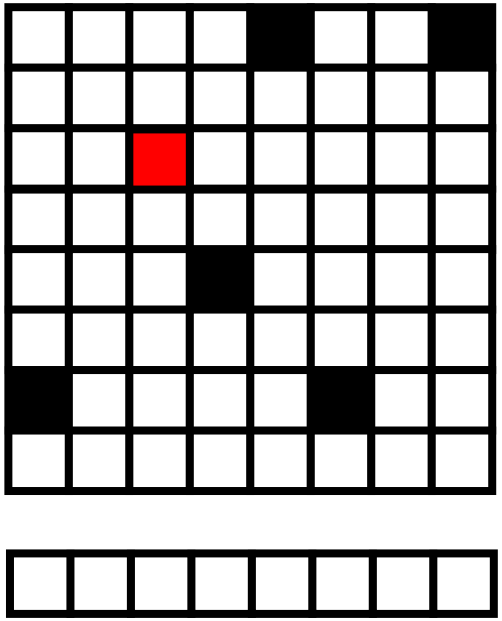

cell
row 4
row 3
row 2
row 1
row 0
row-buffer


cell
row 4
row 3
row 2
row 1
row 0
row-buffer
X
O

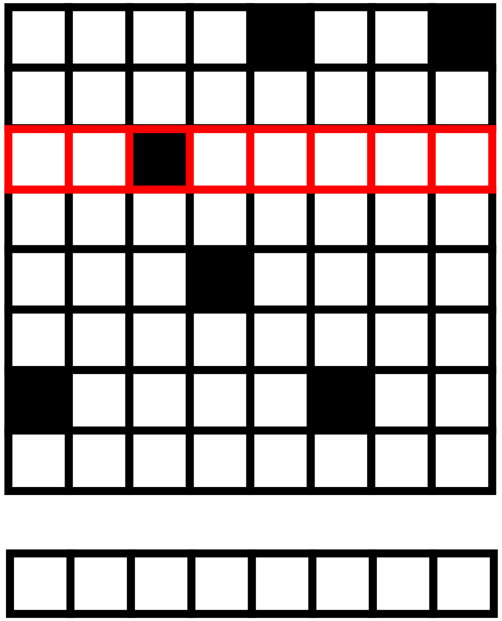
DRAM row buffer



DRAM row buffer



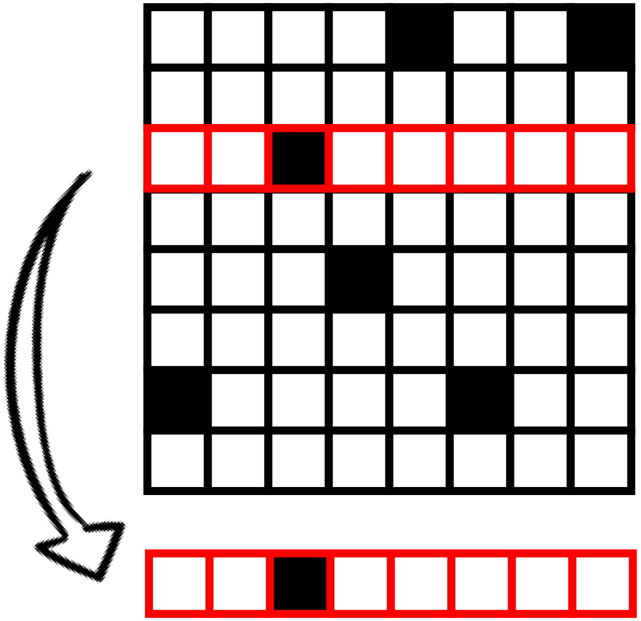
Row buffer



Open

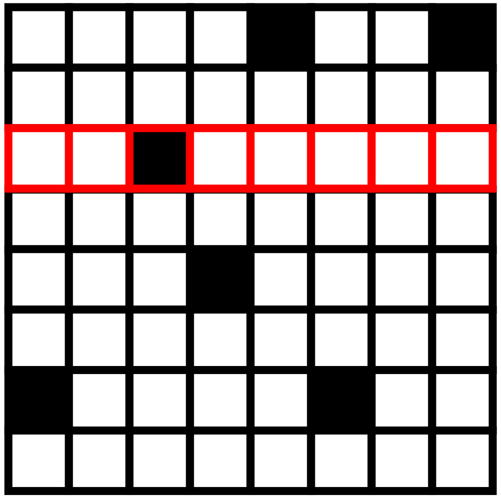
- raise **wordline** to high voltage

Row buffer



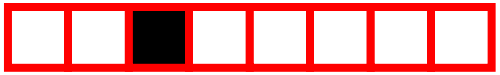
Open
- raise wordline to high voltage
- Connecting capacitor to **bitline**

Row buffer



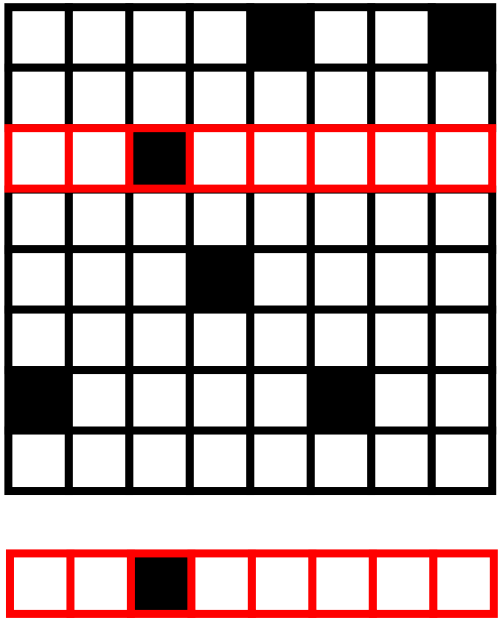
Open

- raise wordline to high voltage
- Connecting capacitor to bitline



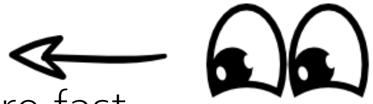
Row buffer

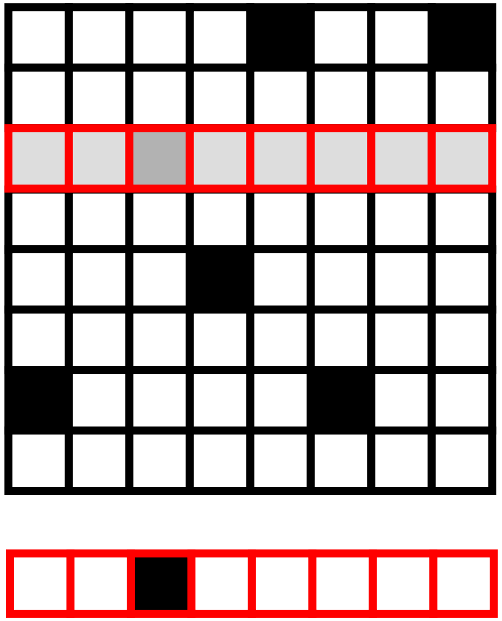
- Access to row buffer are fast



Open
- raise wordline to high voltage
- Connecting capacitor to bitline

Row buffer
- Access to row buffer are fast



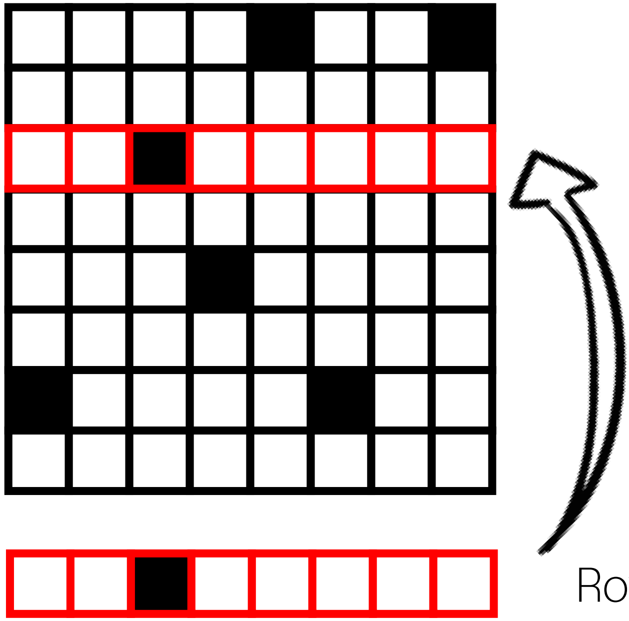


Open

- raise wordline to high voltage
- Connecting capacitor to bitline
- DRO (Destructive Read Out)

Row buffer

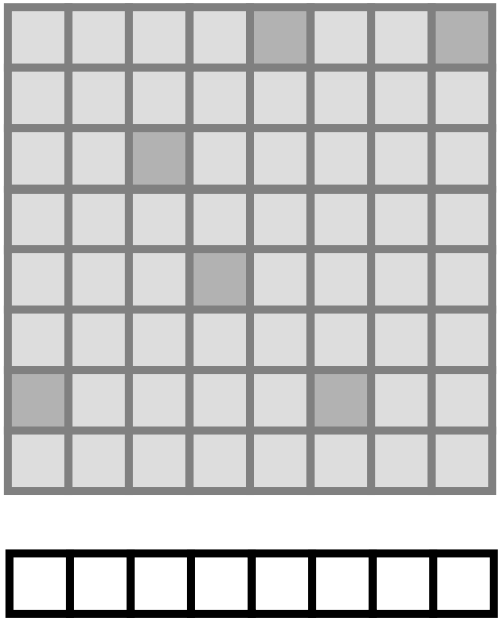
- Access to row buffer are fast

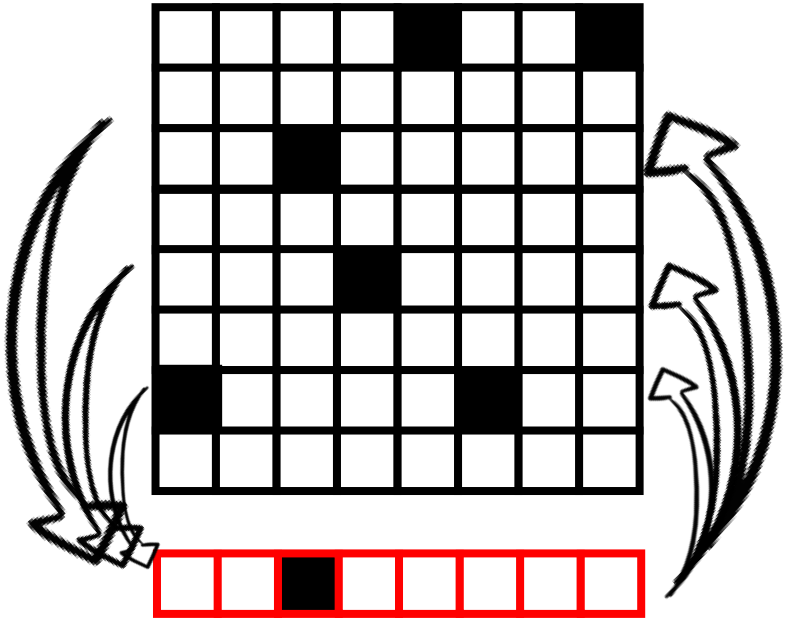


Recharge

- Copy the row back

Row buffer





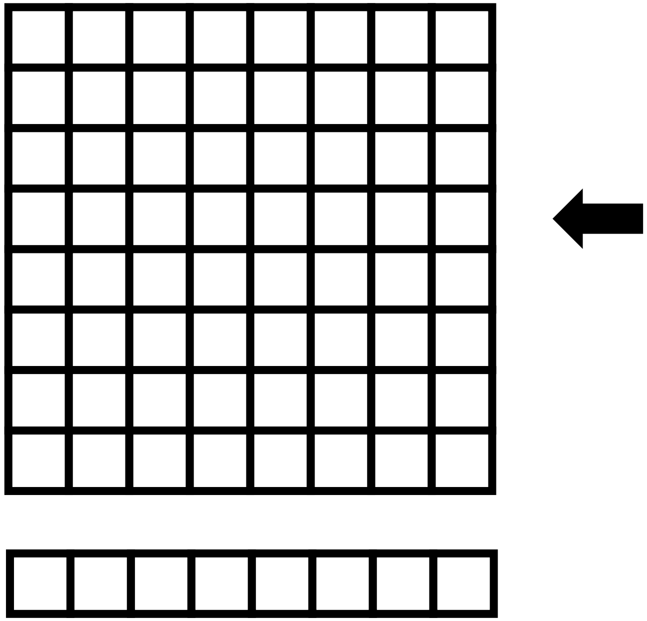
Cells are capacitor!

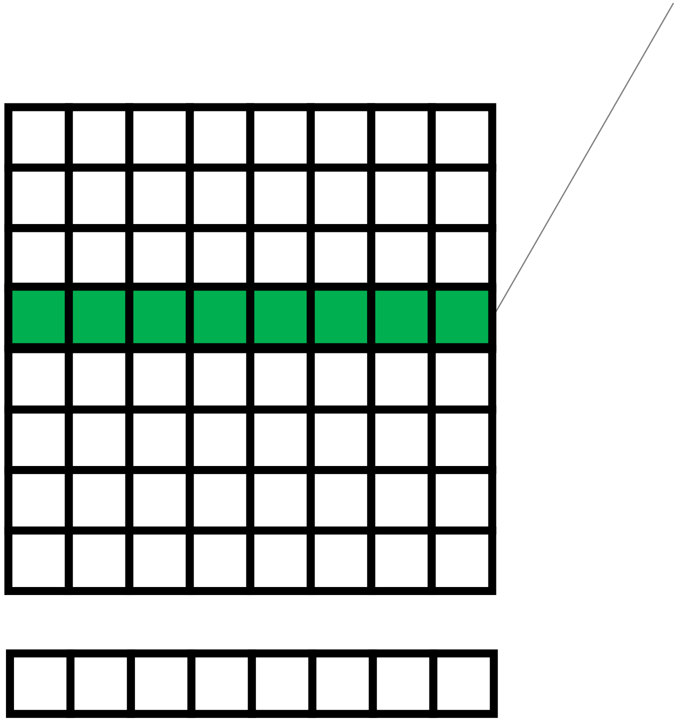
- They leak charge
- Cells should be periodically refreshed
- Refresh circuitry perform refresh cycle within the refresh time interval : 64ms

02

Introduction to rowhammer problems

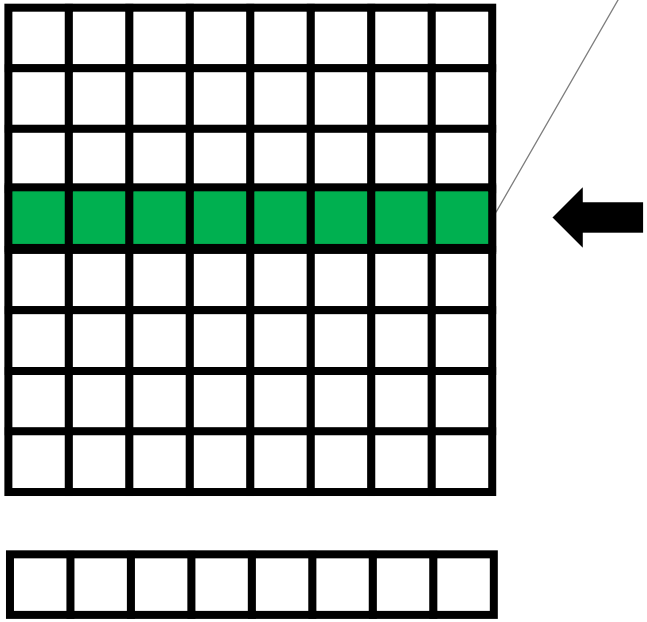
Introduction to rowhammer problems



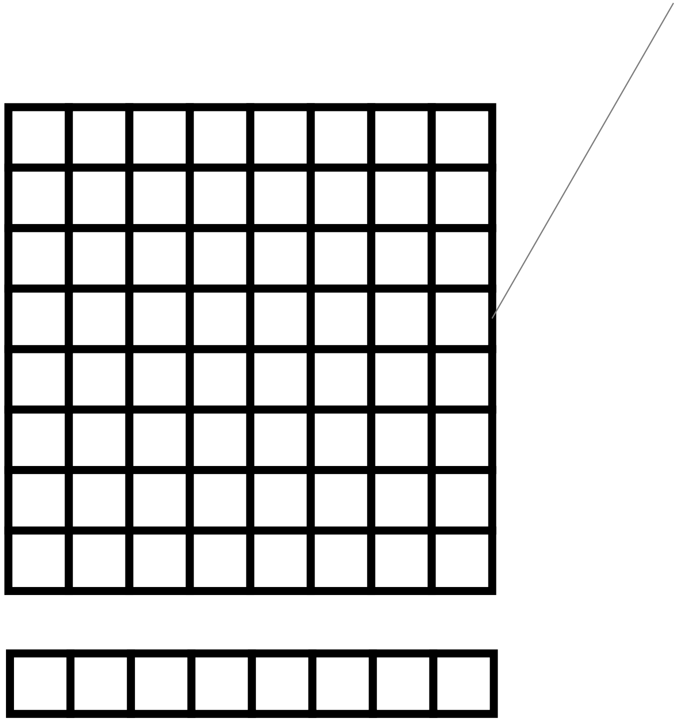


This “aggressor” row is repeatedly activated (hammered)

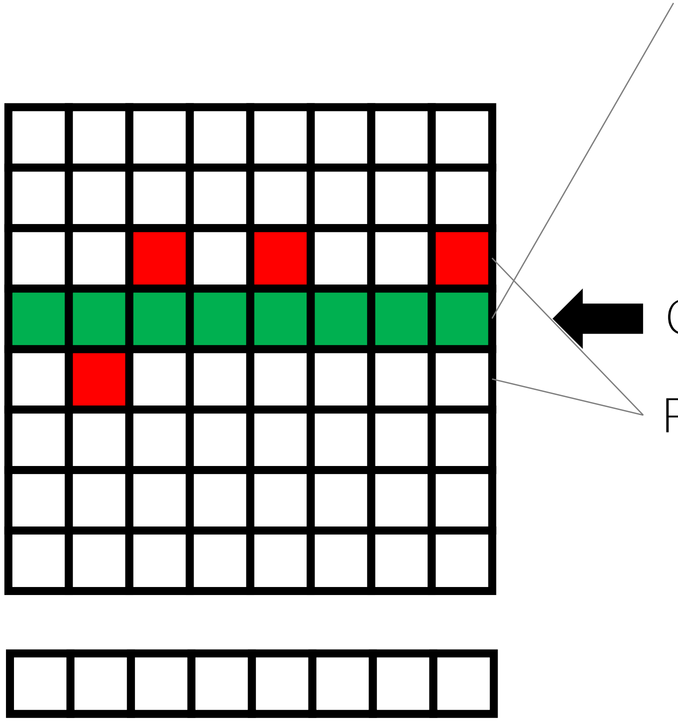
This “aggressor” row is repeatedly activated (hammered)



OPEN (voltage raise)



This “aggressor” row is repeatedly activated (hammered)

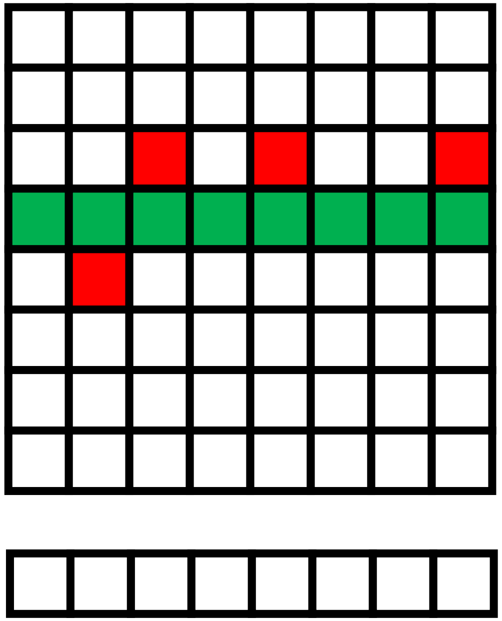


This “aggressor” row is repeatedly activated (hammered)

OPEN (voltage raise)

Result : These “victim” rows get bit flips

Bad Cells



- Randomly distributed
- Constantly flip when hammered
- varies by DRAM module
  - % of rows with bad cells : Varies from 30% to 99.9%

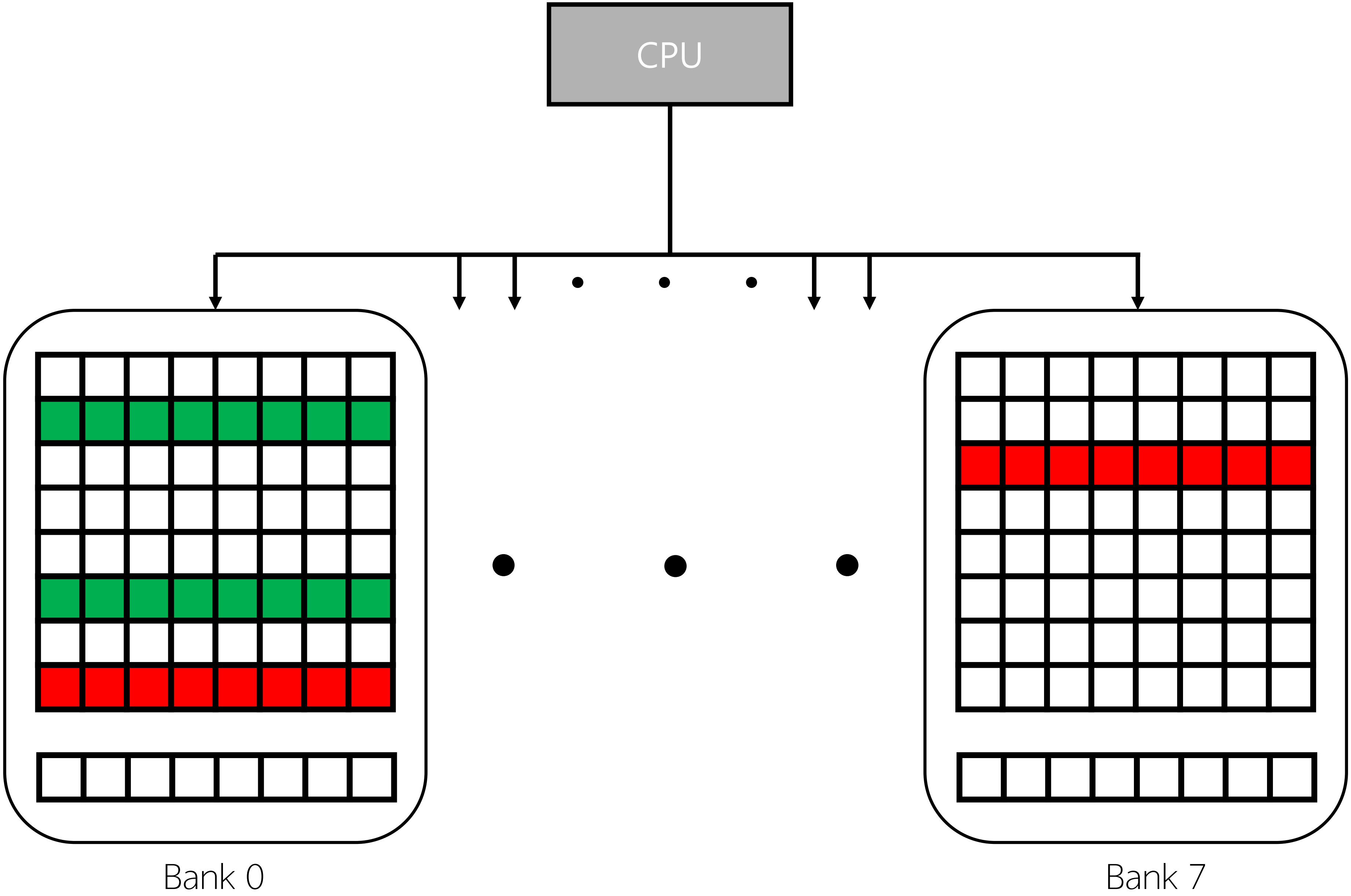
03

Understand bit flipping

by looking hammering code !

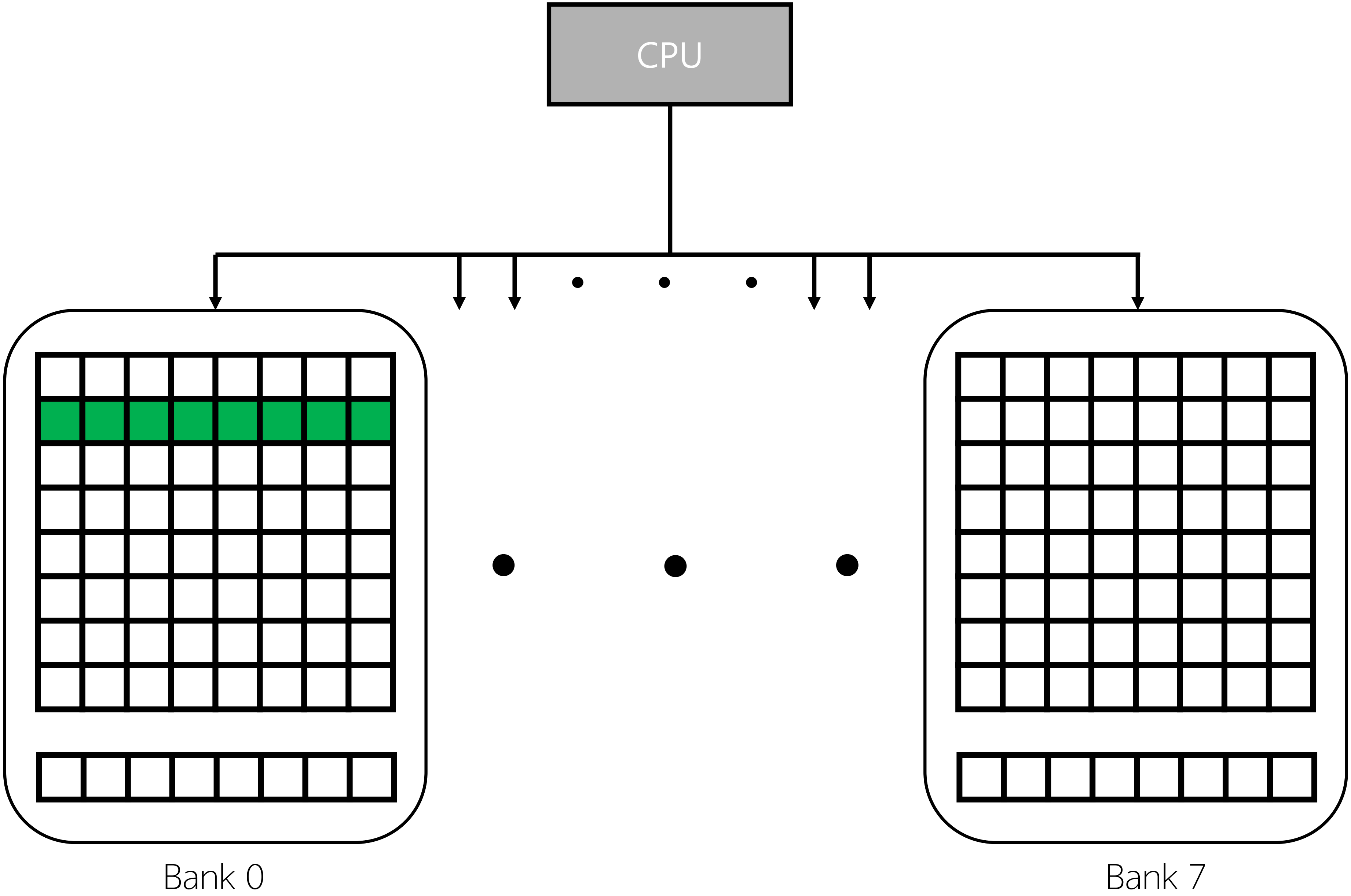
Challenge 1. Right way to flip bit. ①? ②?

Challenge 2. How to find pair of rows?



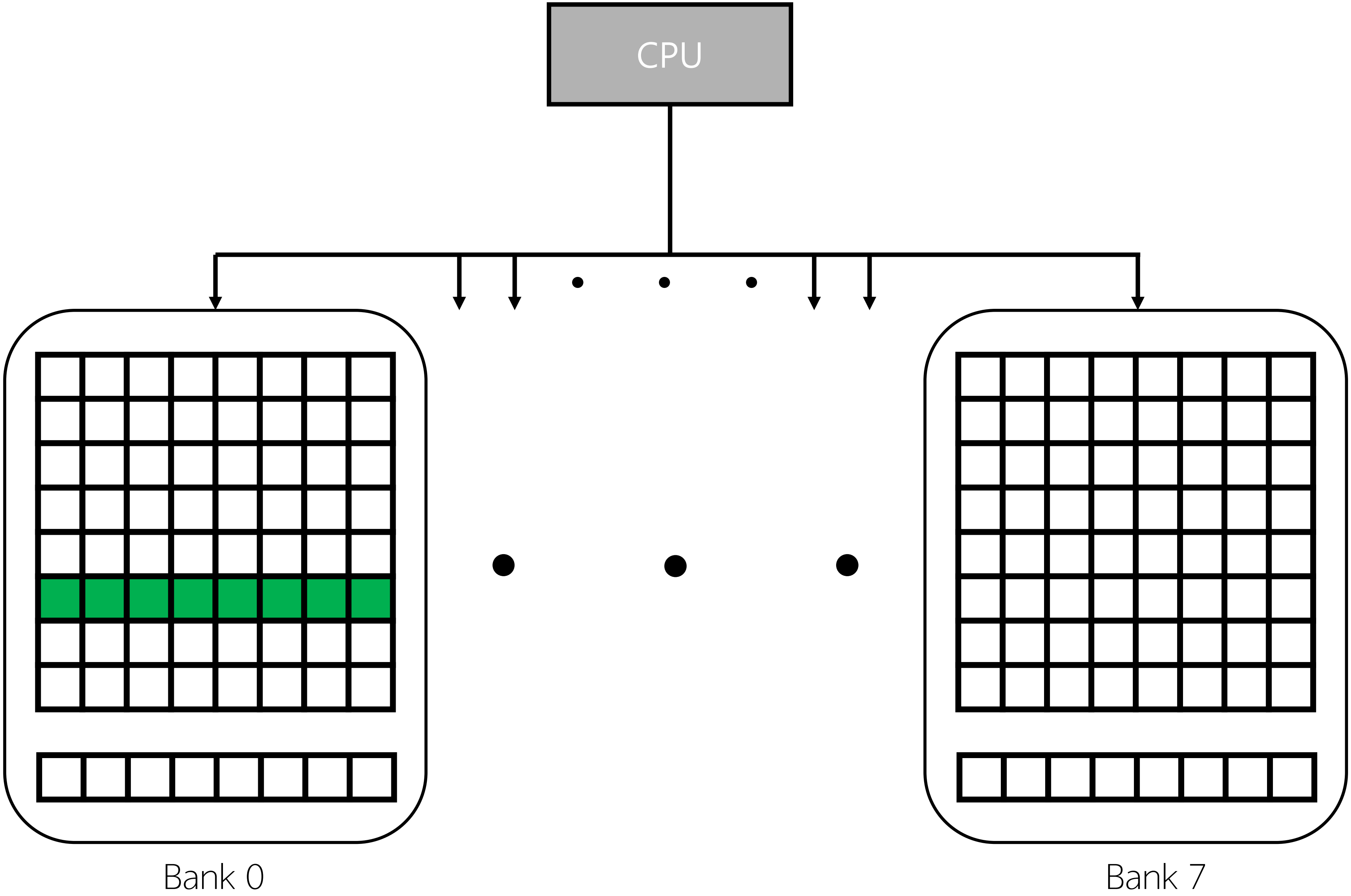
Challenge 1. Right way to flip bit. ①? ②?

Challenge 2. How to find pair of rows?



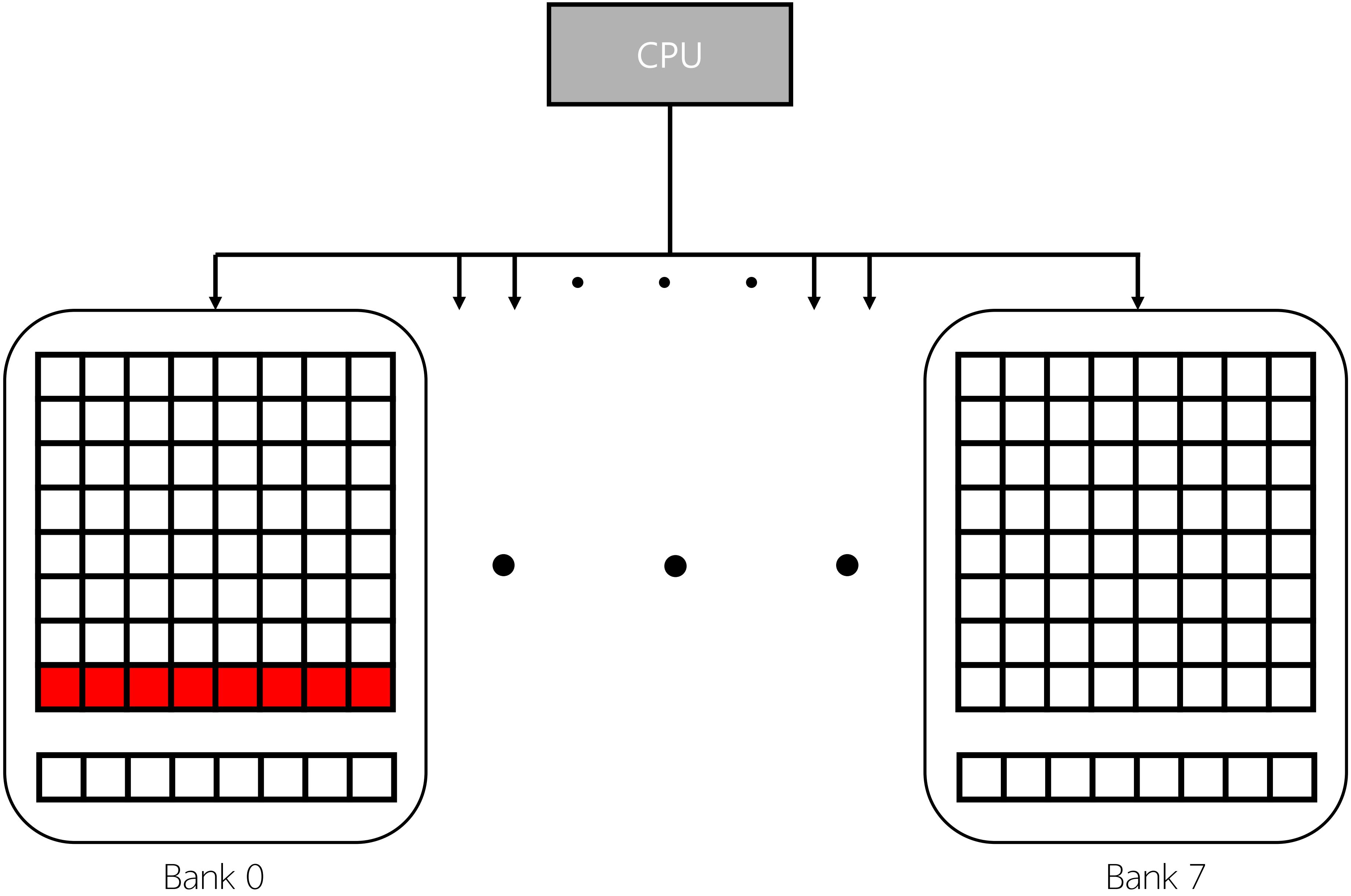
Challenge 1. Right way to flip bit. ①? ②?

Challenge 2. How to find pair of rows?



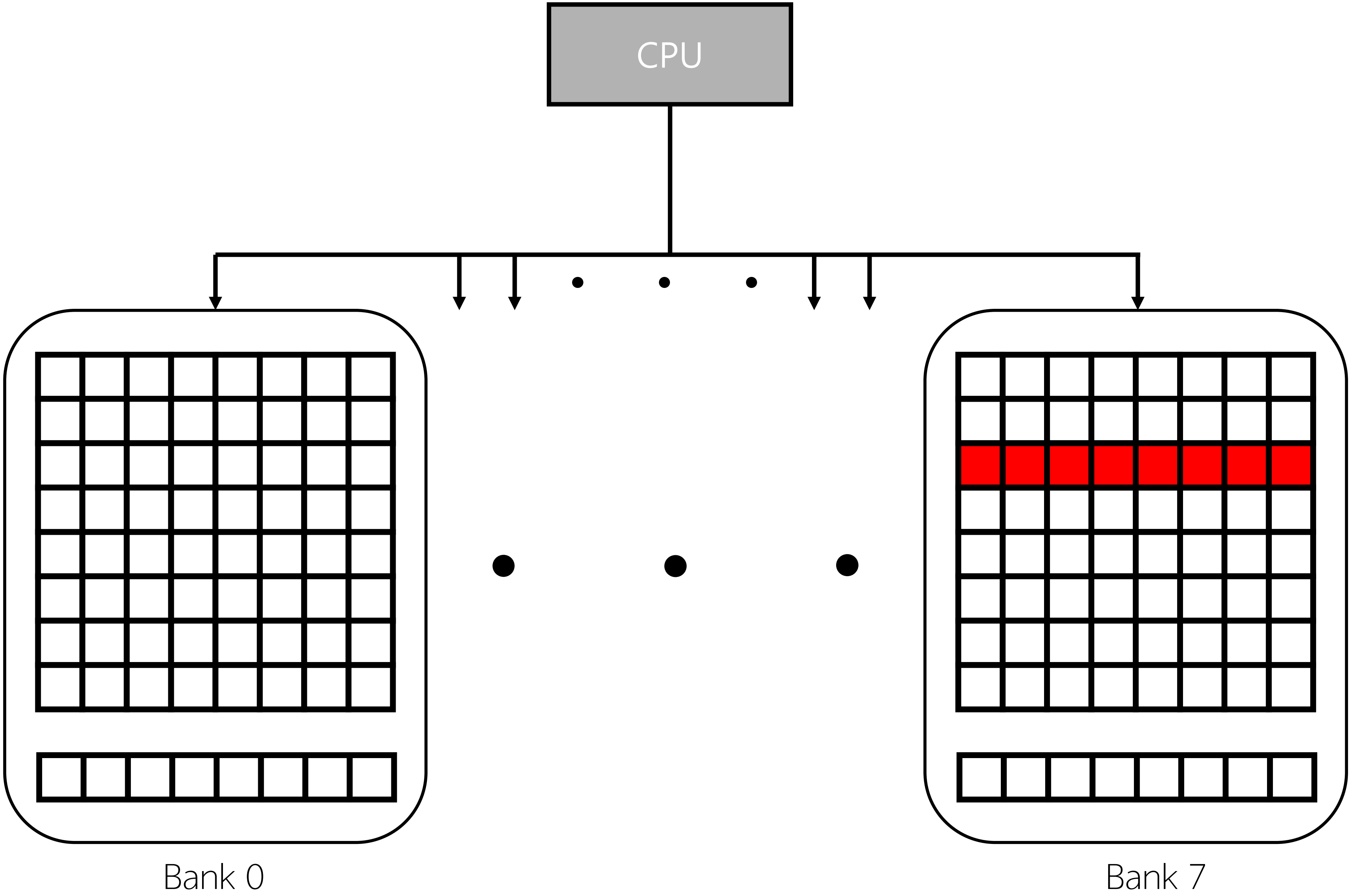
Challenge 1. Right way to flip bit. ①? ②?

Challenge 2. How to find pair of rows?



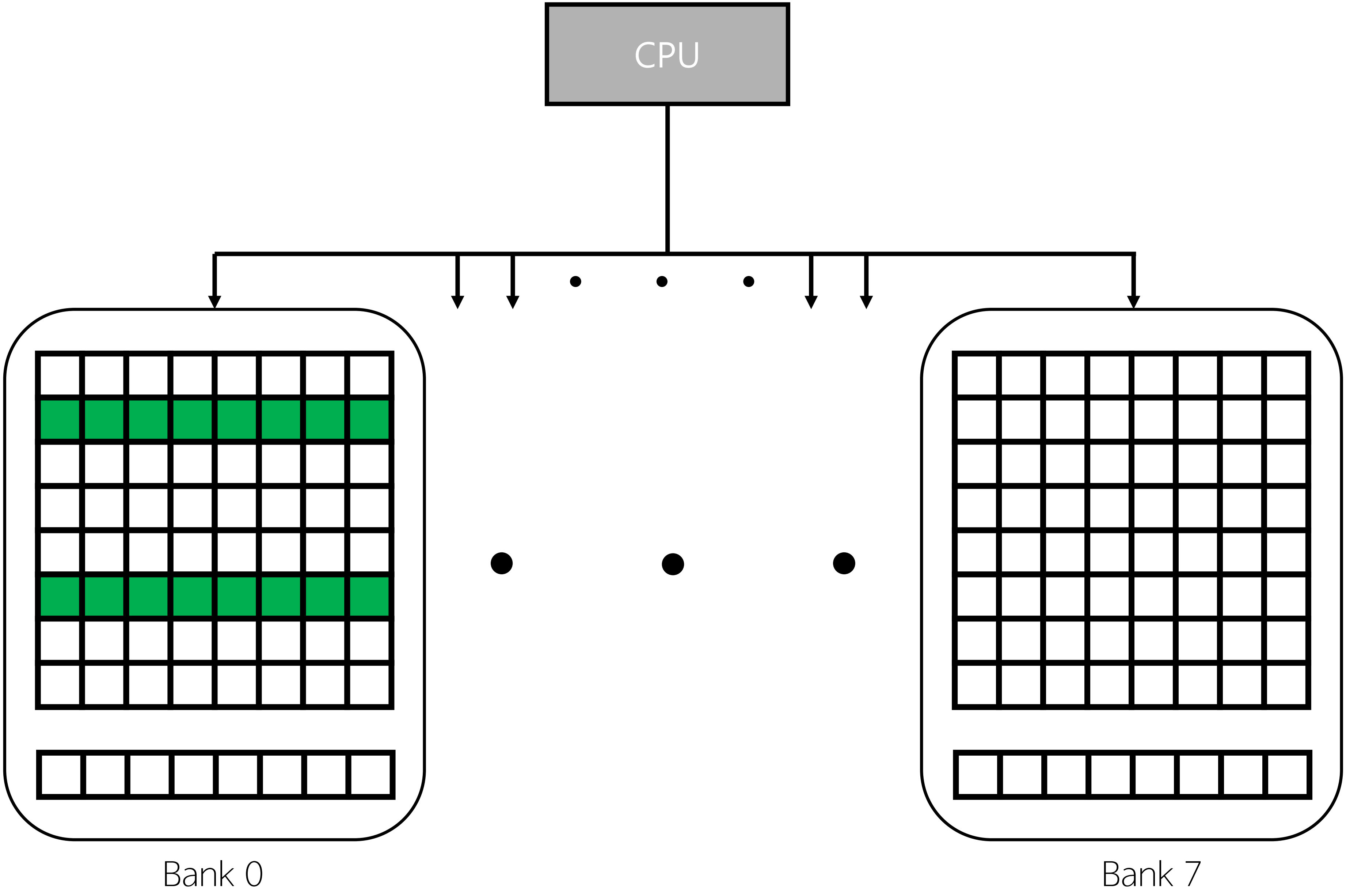
Challenge 1. Right way to flip bit. ①? ②?

Challenge 2. How to find pair of rows?



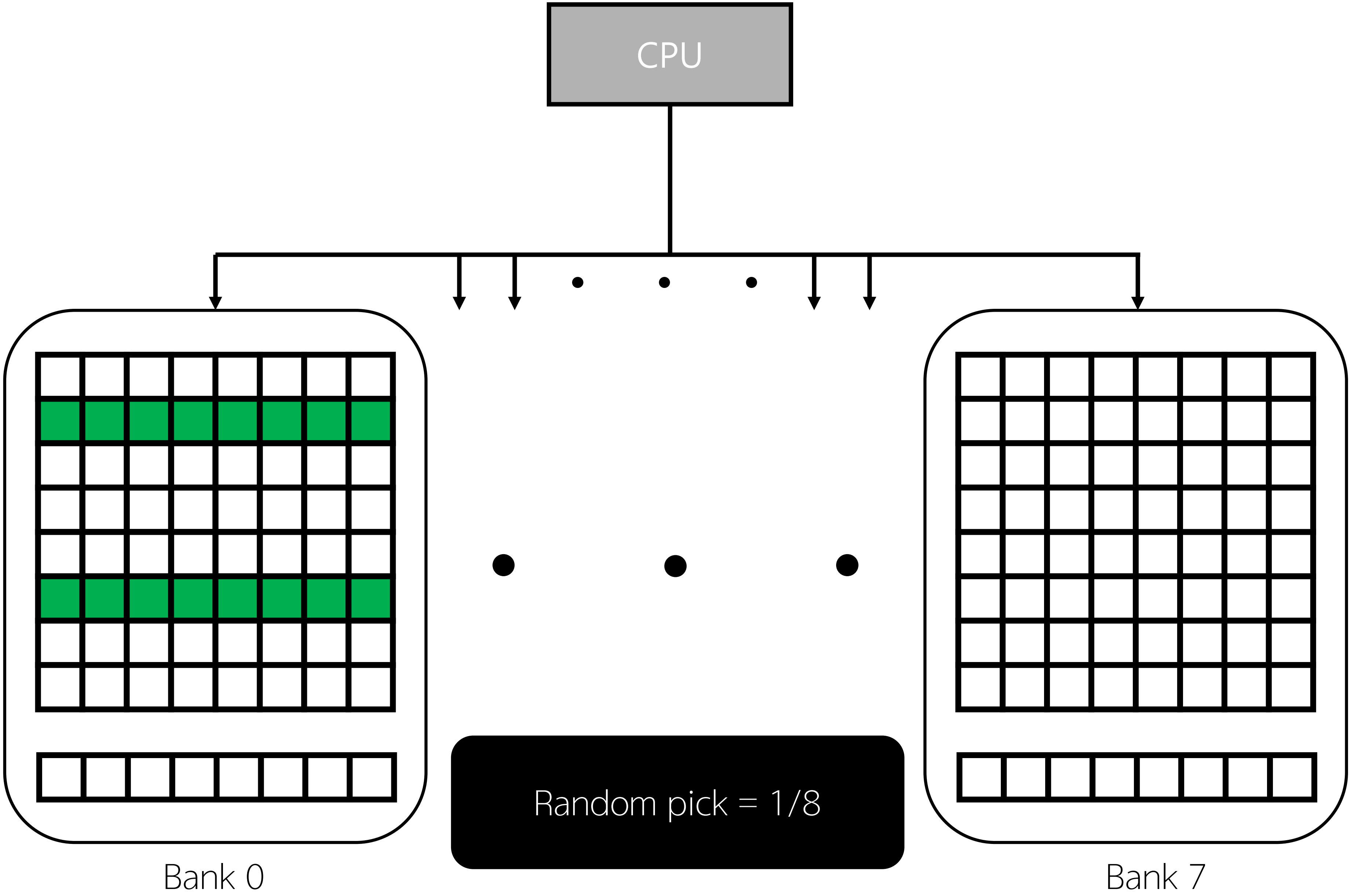
~~Challenge 1. Right way to flip bit. ①? ②?~~ ✓

Challenge 2. How to find pair of rows?



~~Challenge 1. Right way to flip bit. ①? ②?~~ ✓

~~Challenge 2. How to find pair of rows?~~ ✓



CPU

Random pick = 1/8

Bank 0

Bank 7

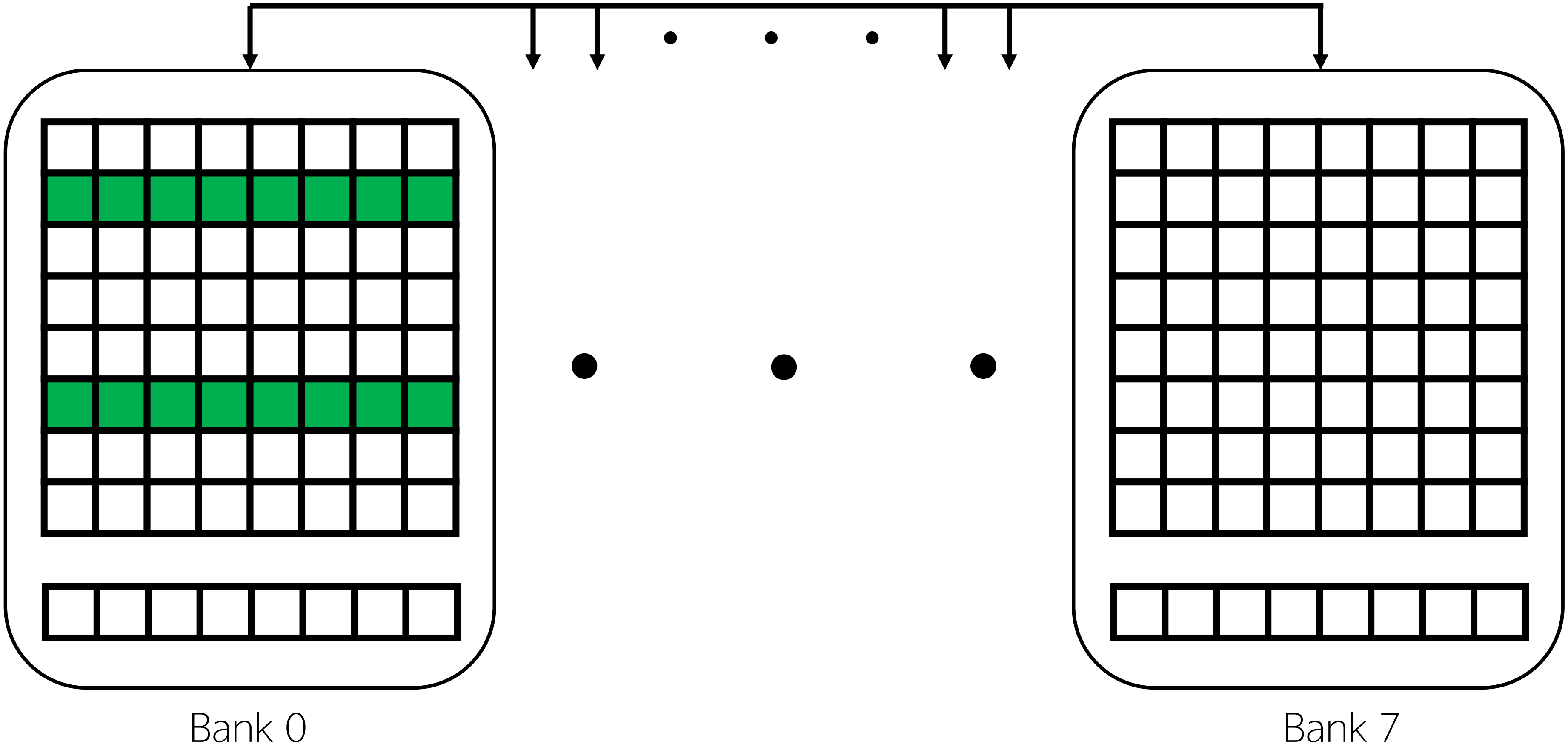
Bit flip code:

1. OPEN – CLOSE rows repeatedly

   pick 2 addresses : Same Bank Different Rows (SBDR)

2. CPU cache by clflush

```
code1a:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   jmp code1a
```



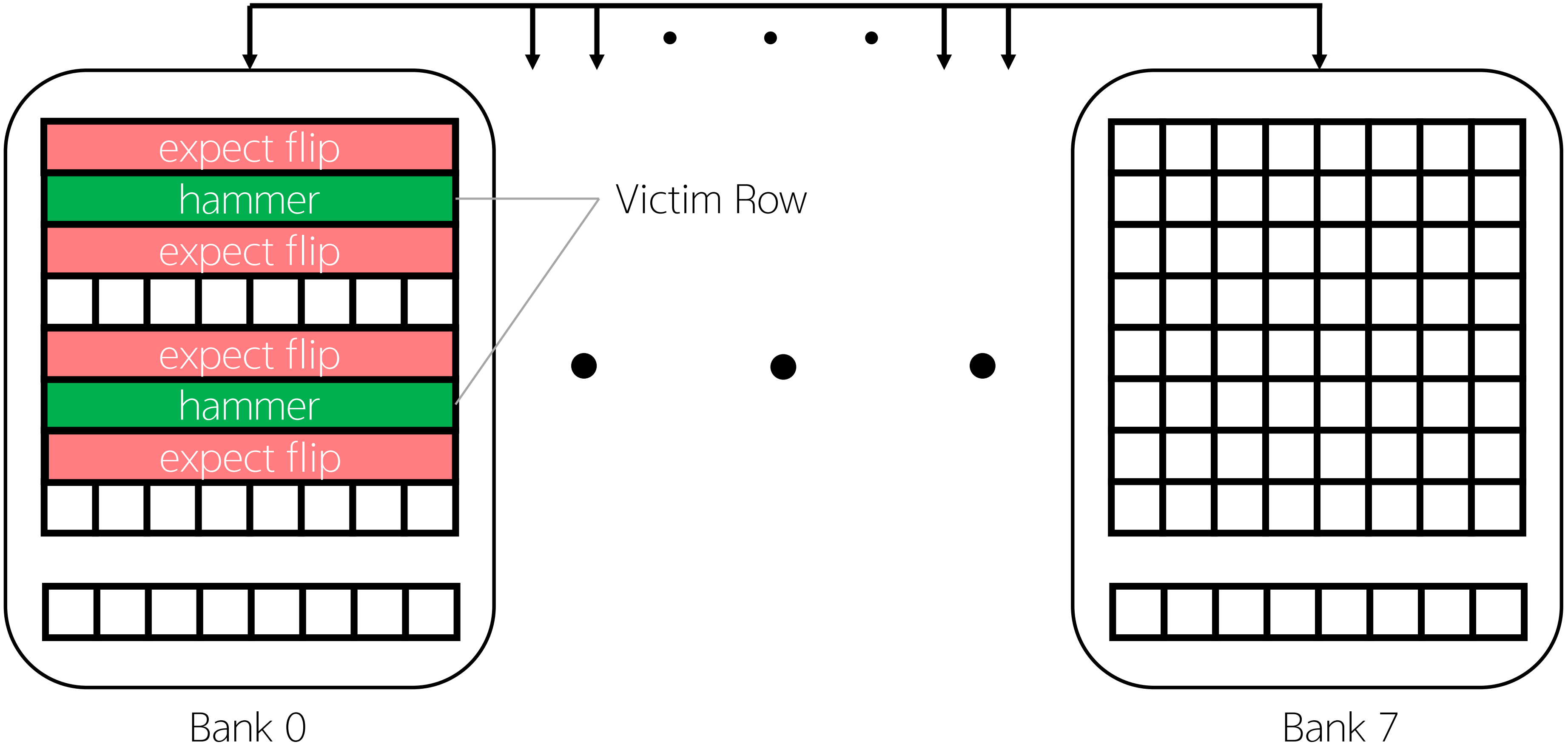
Bit flip code:

1. OPEN – CLOSE rows repeatedly

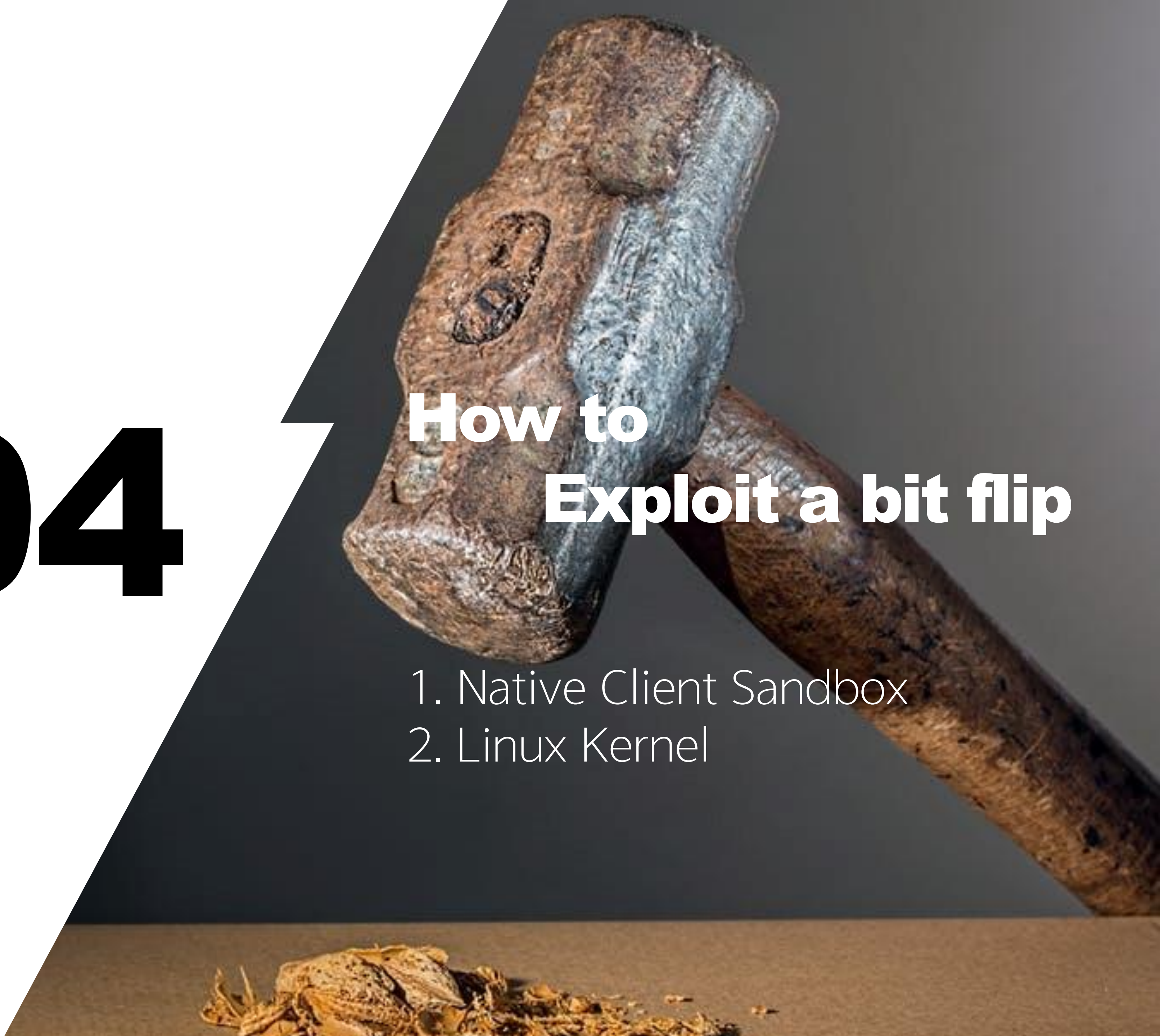
   pick 2 addresses : Same Bank Different Rows (SBDR)

2. CPU cache by clflush

```
code1a:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   jmp code1a
```



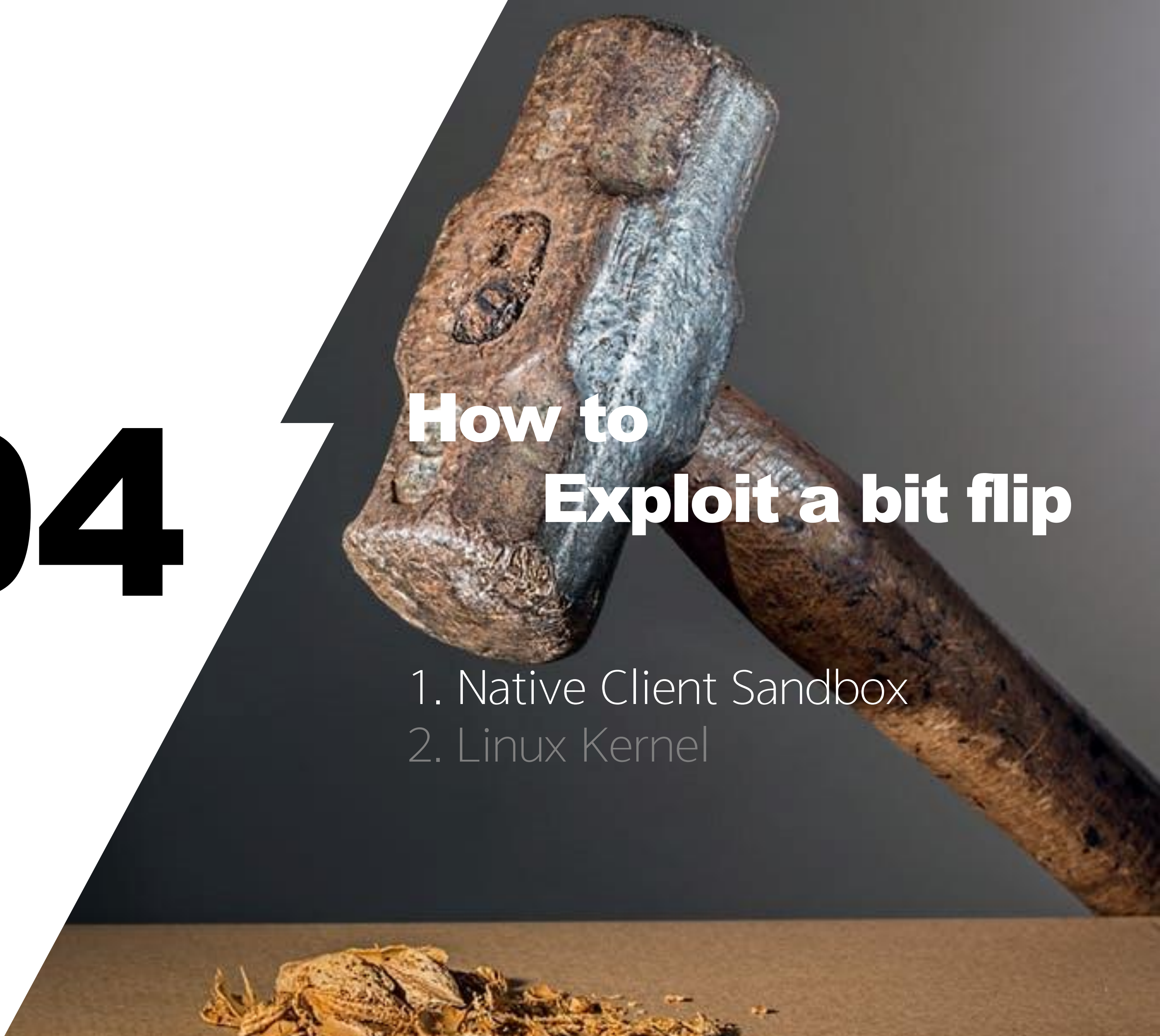
04



How to Exploit a bit flip

1. Native Client Sandbox
2. Linux Kernel

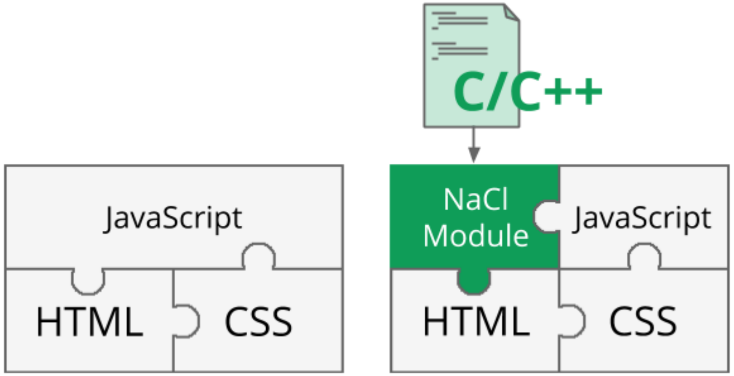
04



How to Exploit a bit flip

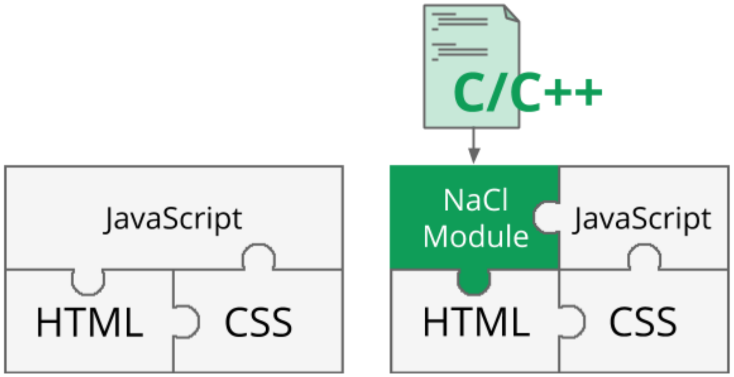
1. Native Client Sandbox
2. Linux Kernel

Native Client Sandbox



- ✓ Sandbox for running C/C++ “native code” on the web
- ✓ Used in chrome
- ✓ Goal : make C/C++ code as safe as javascript
- ✓ In-process sandbox
  - Can’t call host OS’s syscalls

Native Client Sandbox



- ✓ Sandbox for running C/C++ “native code” on the web
- ✓ Used in chrome
- ✓ Go
- ✓ In-

Challenges

1. Mark shellcode as executable
2. Jump to shellcode

Challenges

1. Mark shellcode as executable
2. Jump to shellcode

```
20ea0: 48 b8 0f 05 eb 0c f4 f4 f4 f4
             movabs $0xf4f4f4f40ceb050f, %rax
```

} Allowed by NaCl's validater

This conceals:

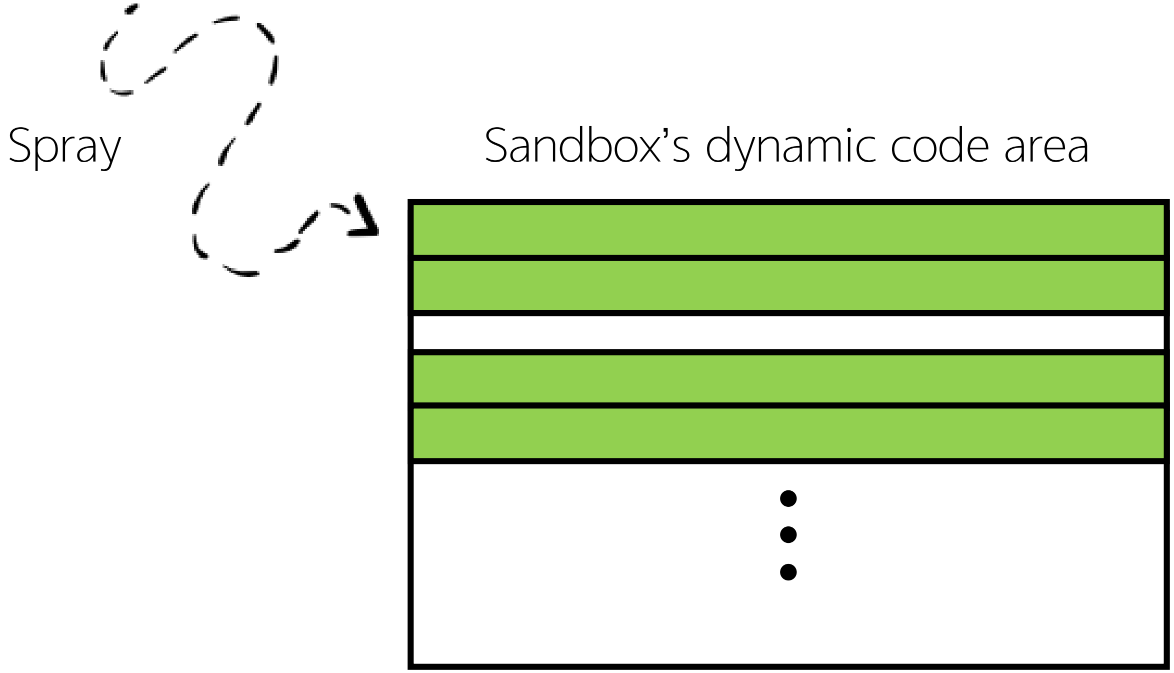
```
20ea2: 0f 05    syscall
20ea4: eb 0c    jmp ...   // Jump to next hidden instr
20ea6: f4       hlt       // Padding
```

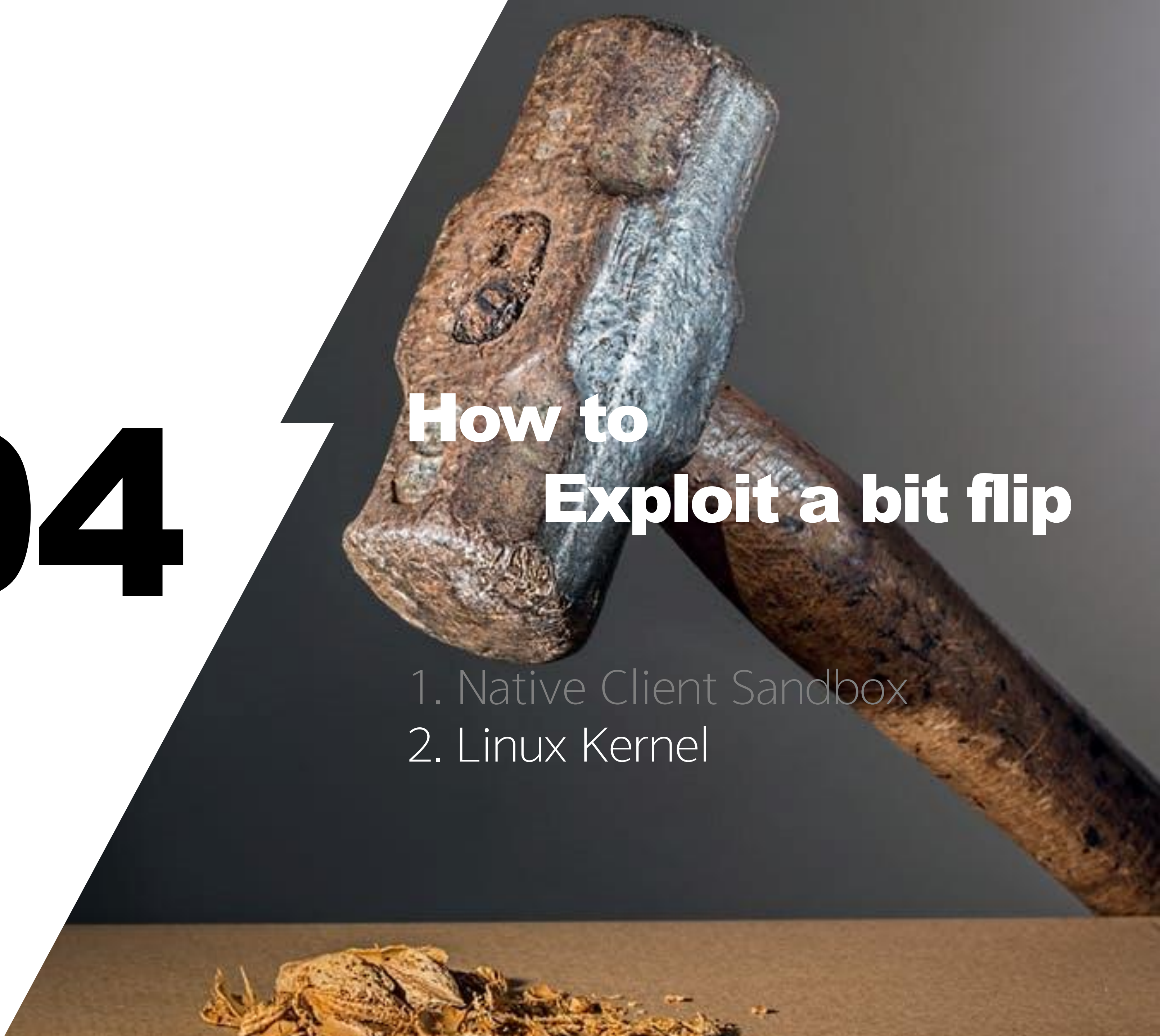
Challenges

1. Mark shellcode as executable
2. Jump to shellcode

Only allows “jmp *%rax” as part of this safe indirect jump sequence:

```
4c 01 f8   addq %r15, %rax   // Add %r15, the sandbox base address.
ff e0      jmp *%rax         // Indirect jump.
```

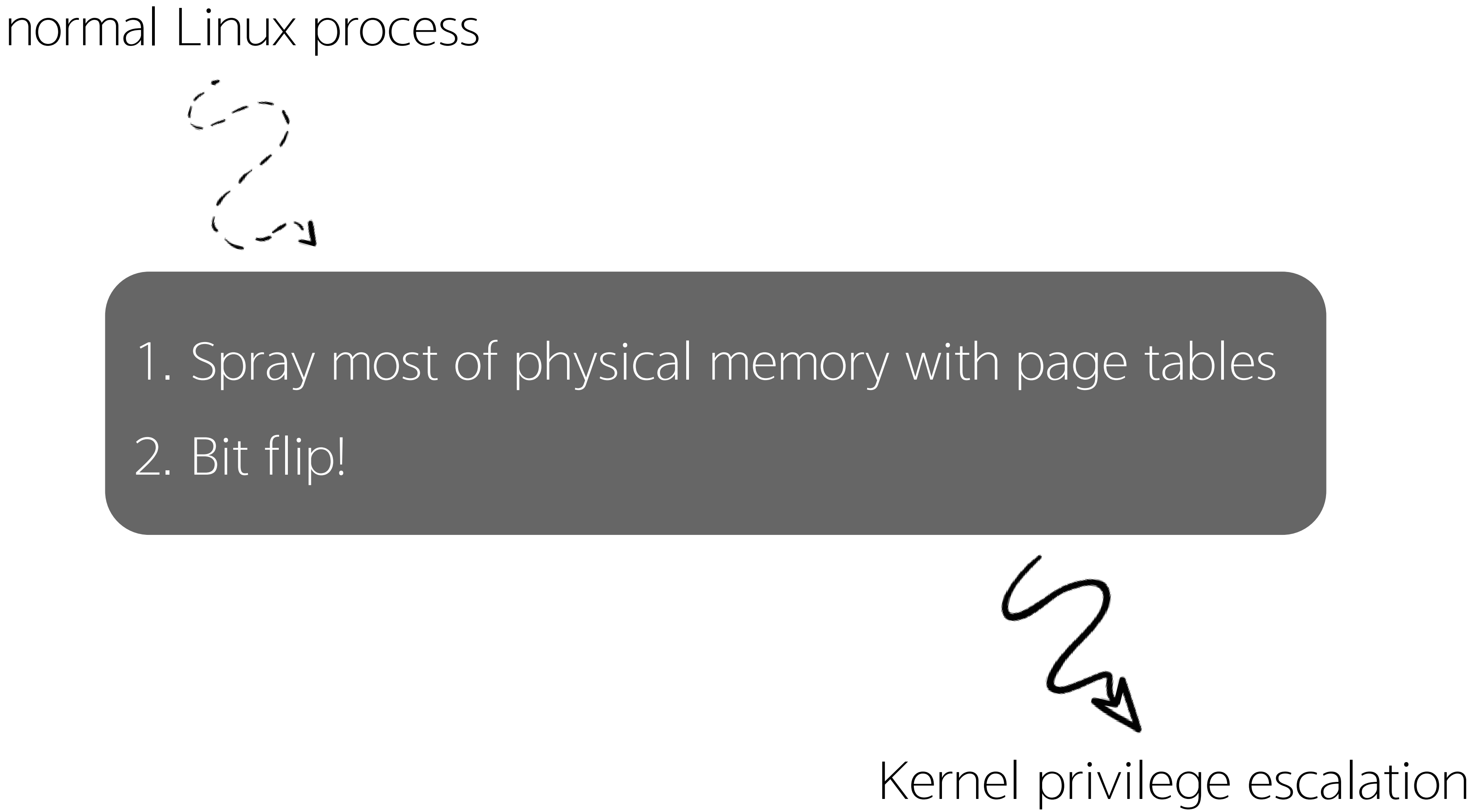




04

How to Exploit a bit flip

1. Native Client Sandbox
2. Linux Kernel



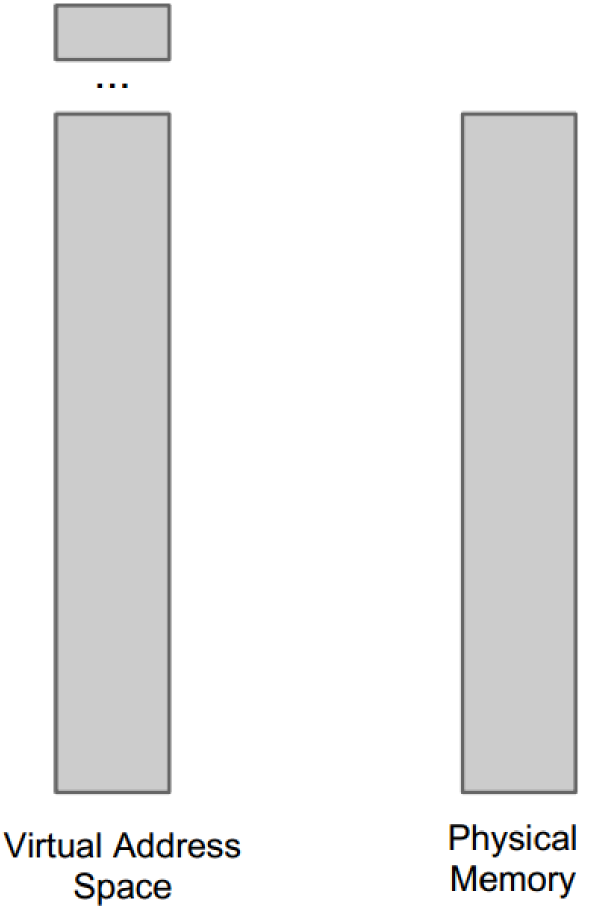
normal Linux process

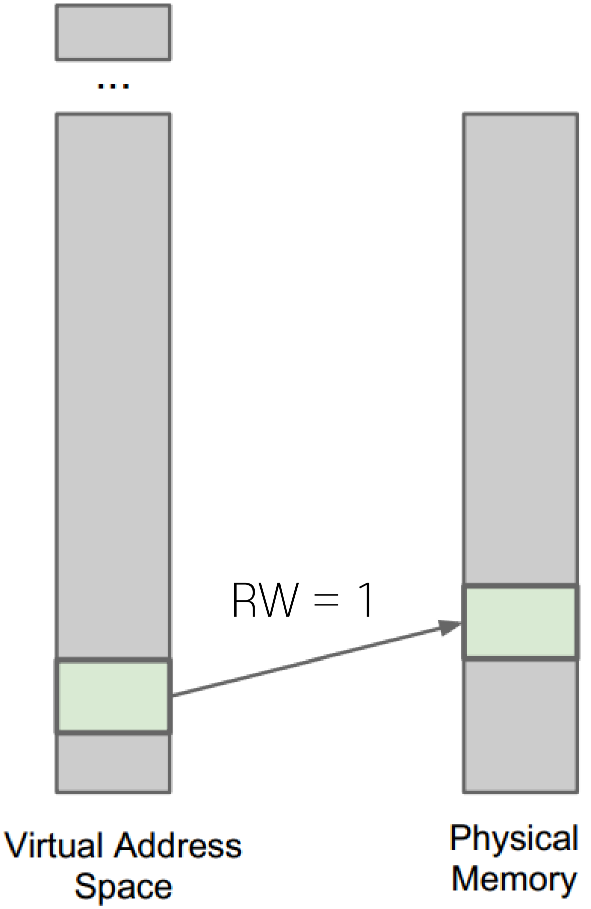
1. Spray most of physical memory with page tables

2. Bit flip!

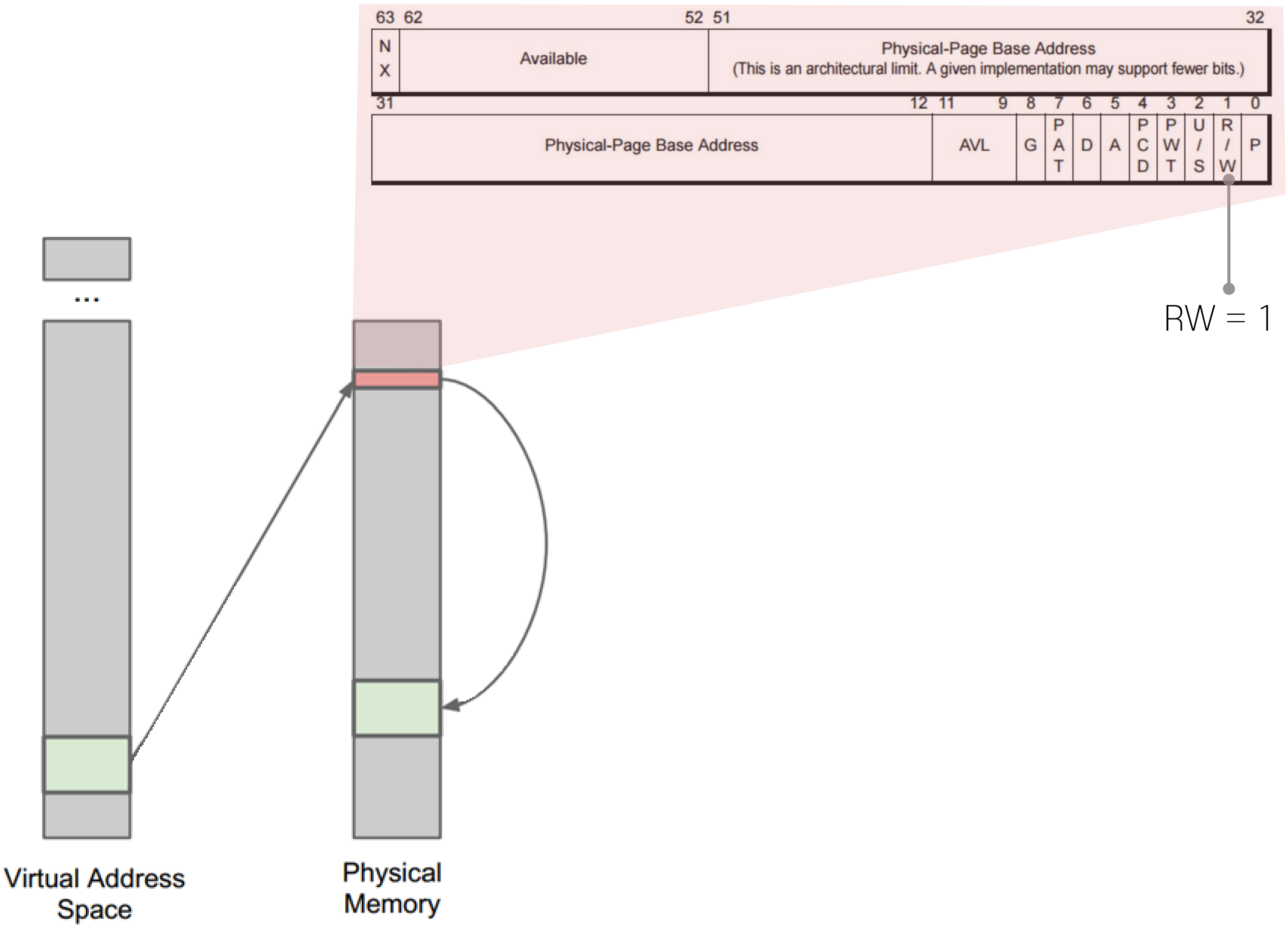
Kernel privilege escalation

Linux kernel exploit



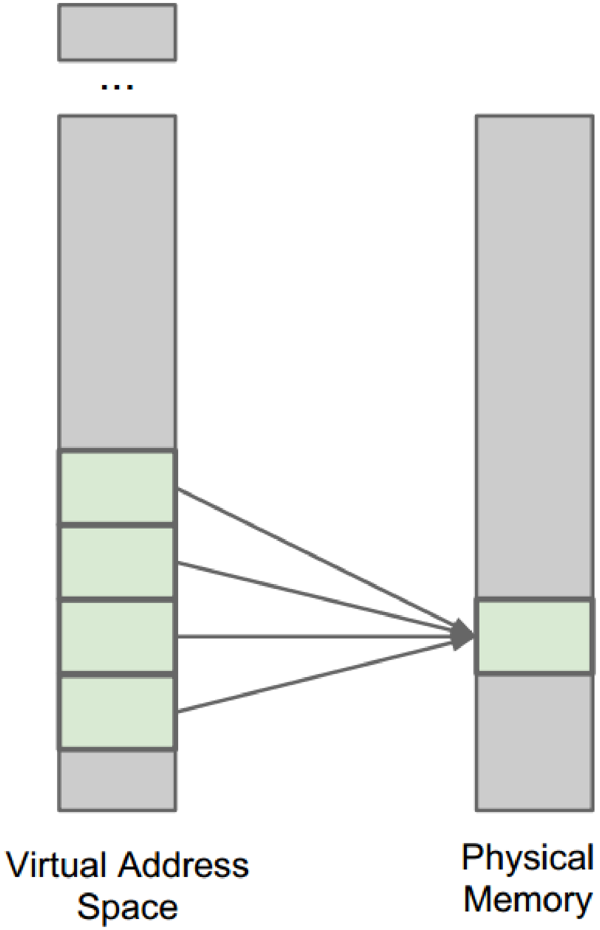


Create shared memory


63 62 52 51 32
N X
Available
Physical-Page Base Address
(This is an architectural limit. A given implementation may support fewer bits.)
31 12 11 9 8 7 6 5 4 3 2 1 0
Physical-Page Base Address
AVL
G
PAT
D
A
PCD
PWT
U/S
R/W
P
...
Virtual Address Space
Physical Memory

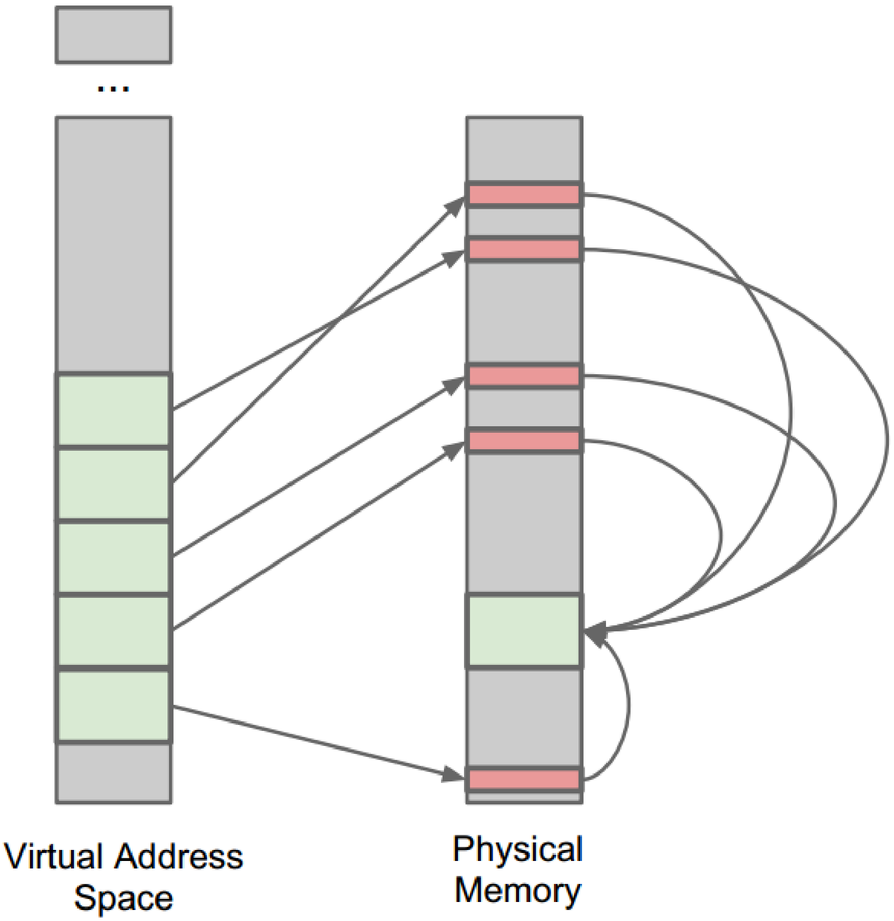
RW = 1

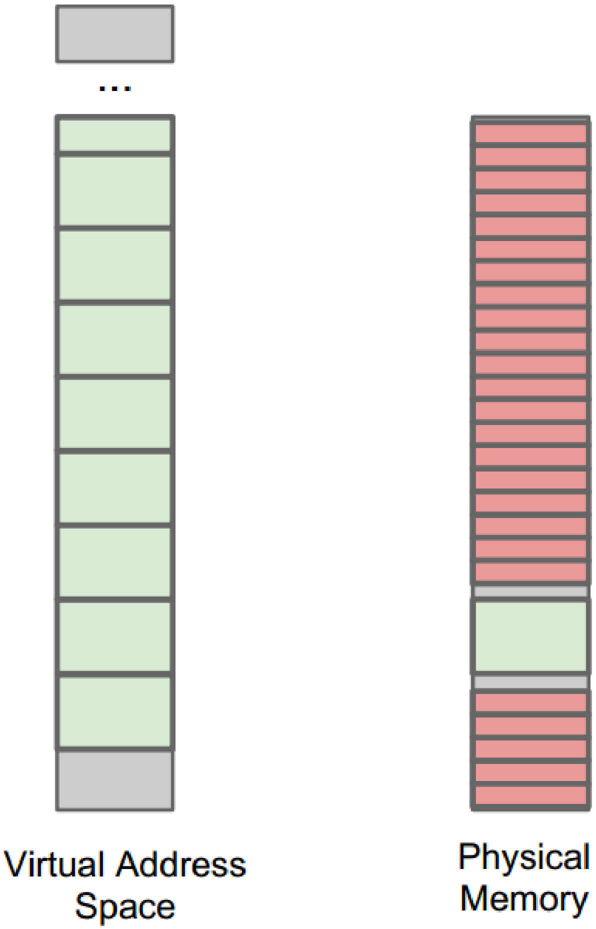
1. mmap() data file repeatedly
2. Spray memory page table

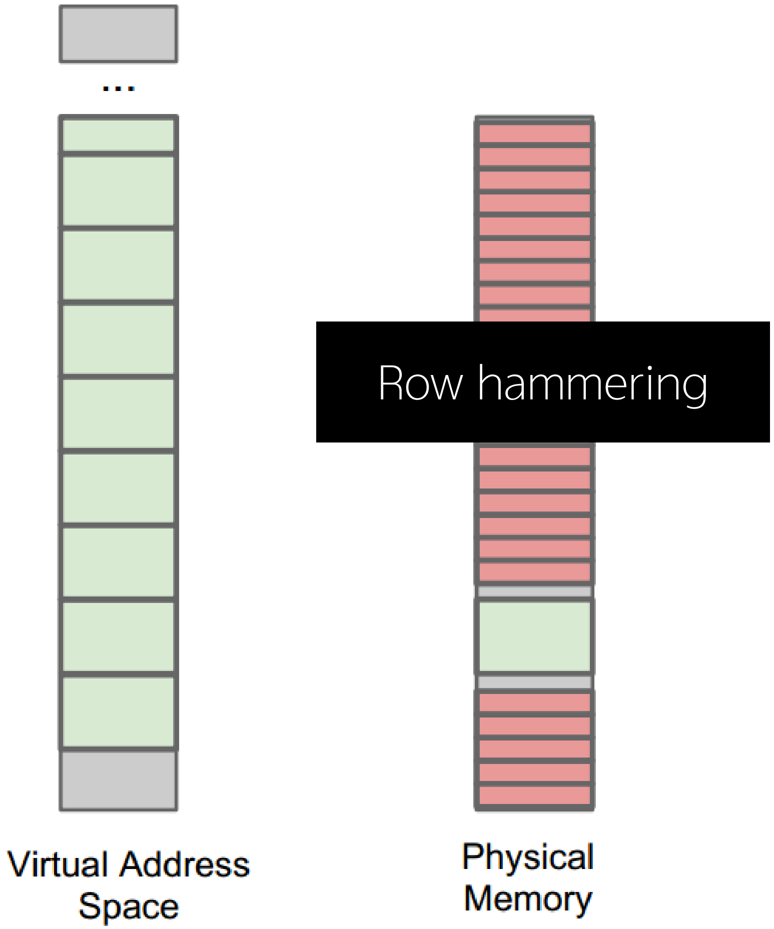


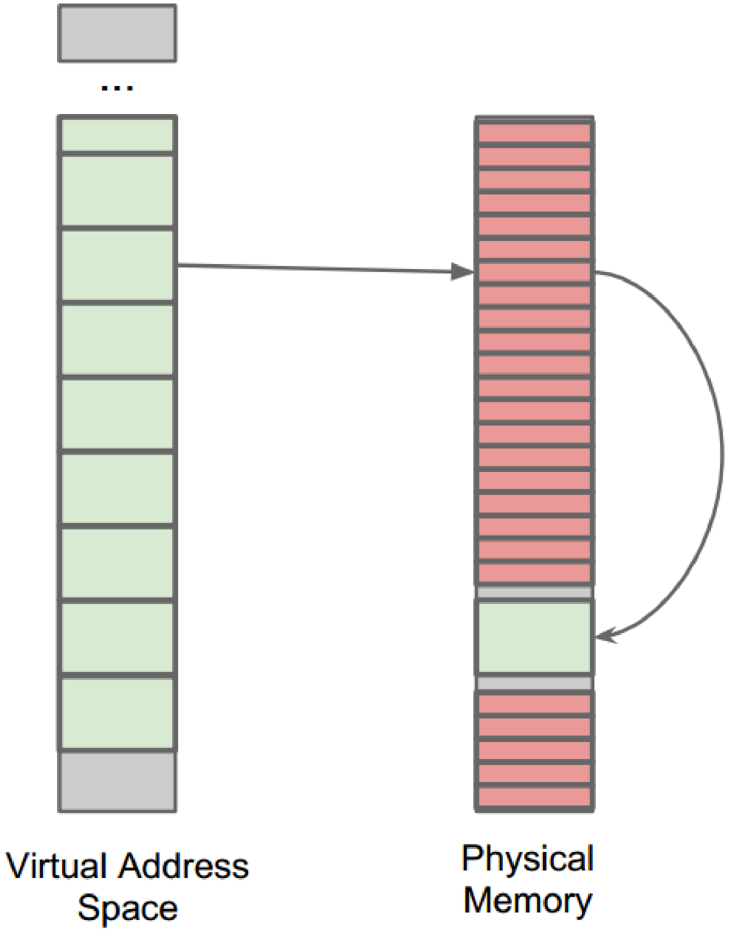
Map it multiple times

1. mmap() data file repeatedly
2. Spray memory page table

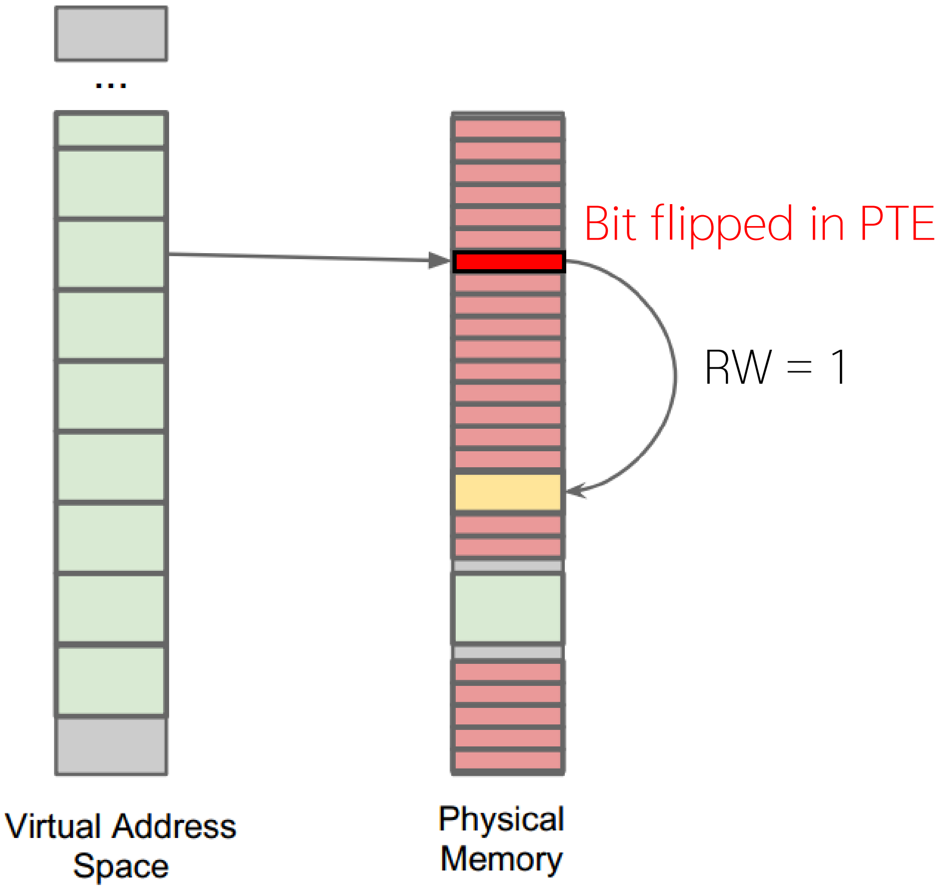



...
Virtual Address Space
Physical Memory

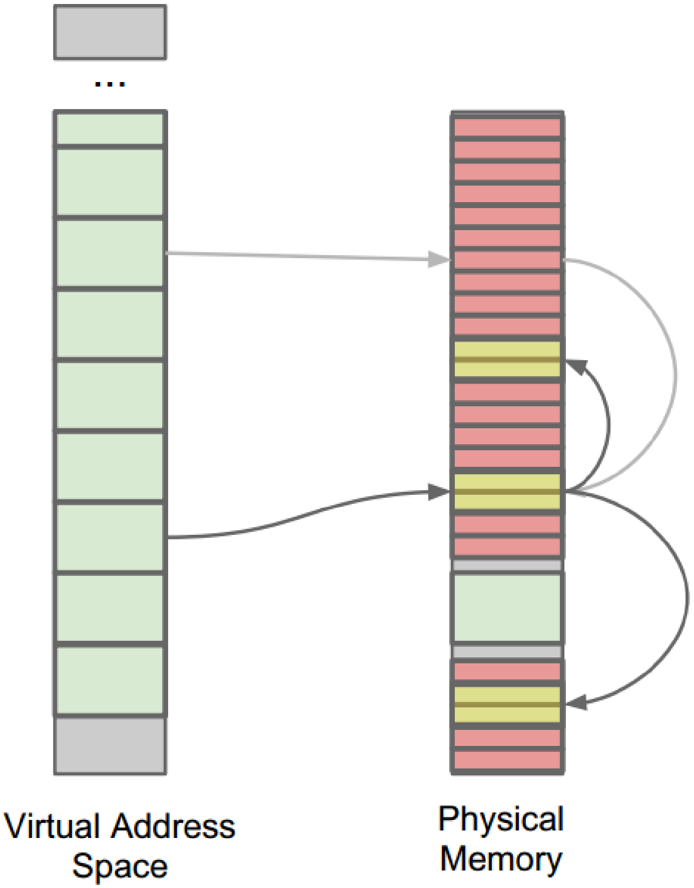

...
Row hammering
Virtual Address Space
Physical Memory


...
Virtual Address Space
Physical Memory

Got write access to page table!



Got write access to page table!



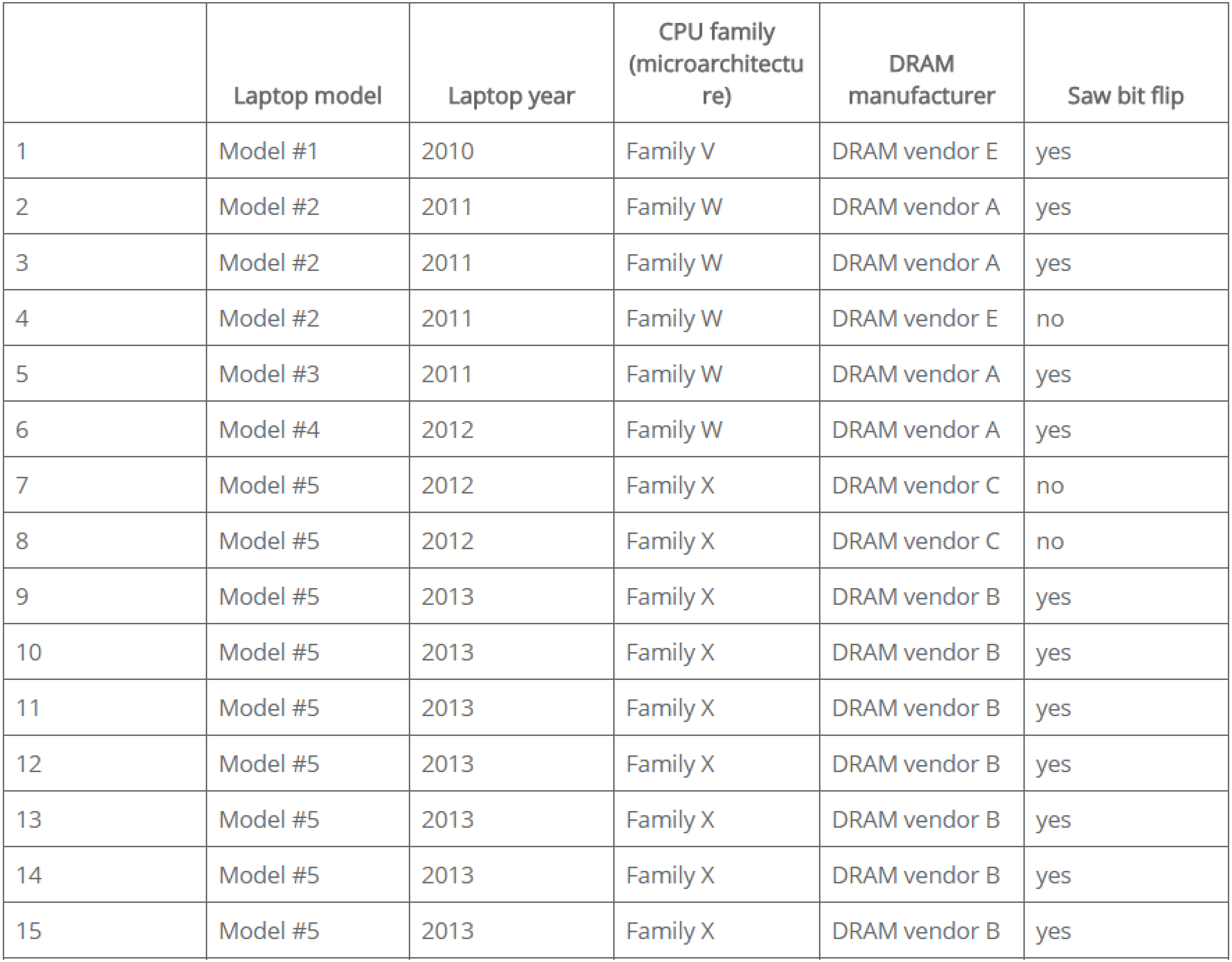
Overwrite entry point of
SUID-root executable (e.g. /bin/ping) to shell code

↓

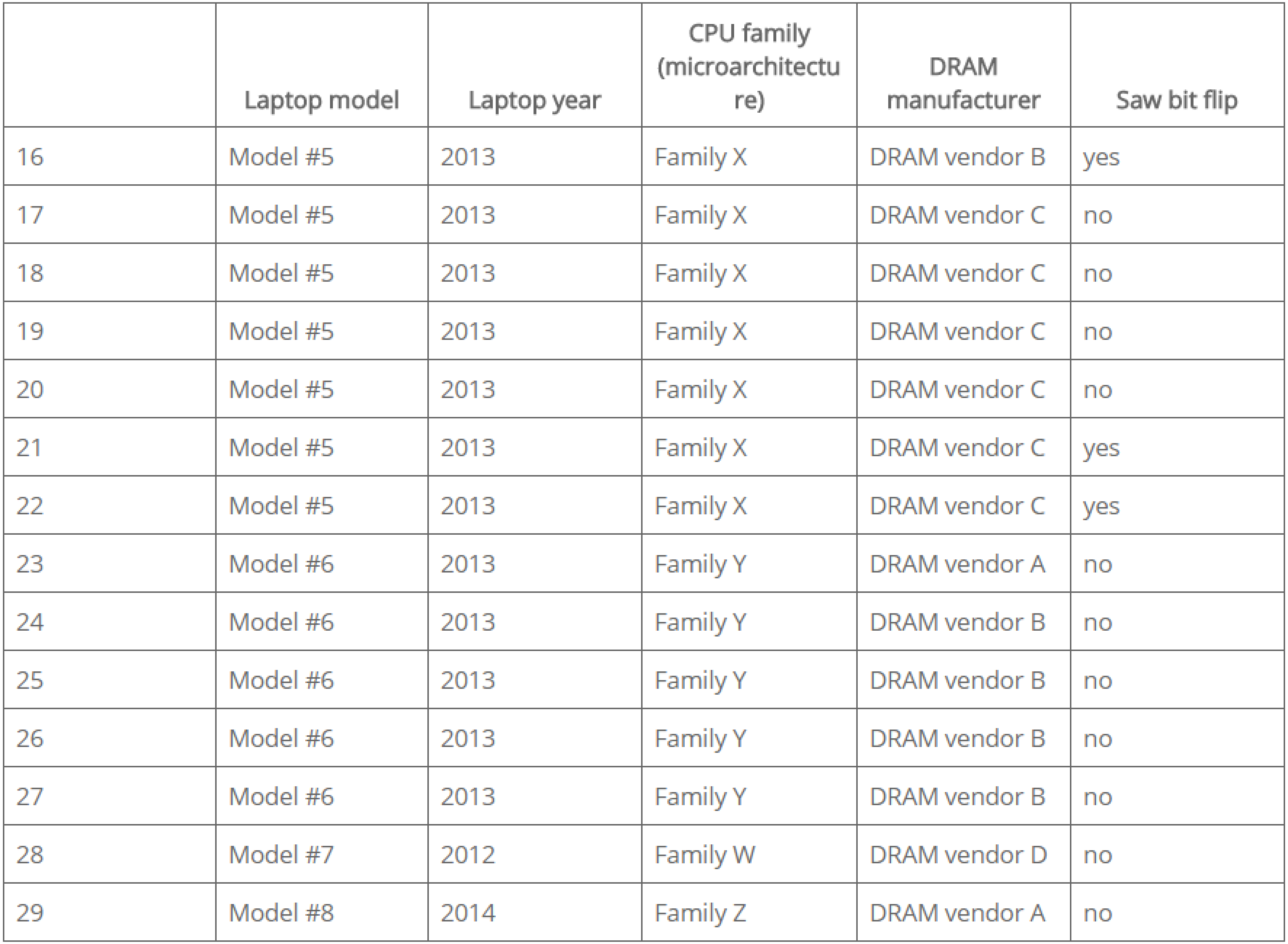
Privilege escalation !

05

Experimental results



	Laptop model	Laptop year	CPU family (microarchitecture)	DRAM manufacturer	Saw bit flip
1	Model #1	2010	Family V	DRAM vendor E	yes
2	Model #2	2011	Family W	DRAM vendor A	yes
3	Model #2	2011	Family W	DRAM vendor A	yes
4	Model #2	2011	Family W	DRAM vendor E	no
5	Model #3	2011	Family W	DRAM vendor A	yes
6	Model #4	2012	Family W	DRAM vendor A	yes
7	Model #5	2012	Family X	DRAM vendor C	no
8	Model #5	2012	Family X	DRAM vendor C	no
9	Model #5	2013	Family X	DRAM vendor B	yes
10	Model #5	2013	Family X	DRAM vendor B	yes
11	Model #5	2013	Family X	DRAM vendor B	yes
12	Model #5	2013	Family X	DRAM vendor B	yes
13	Model #5	2013	Family X	DRAM vendor B	yes
14	Model #5	2013	Family X	DRAM vendor B	yes
15	Model #5	2013	Family X	DRAM vendor B	yes



	Laptop model	Laptop year	CPU family (microarchitecture)	DRAM manufacturer	Saw bit flip
16	Model #5	2013	Family X	DRAM vendor B	yes
17	Model #5	2013	Family X	DRAM vendor C	no
18	Model #5	2013	Family X	DRAM vendor C	no
19	Model #5	2013	Family X	DRAM vendor C	no
20	Model #5	2013	Family X	DRAM vendor C	no
21	Model #5	2013	Family X	DRAM vendor C	yes
22	Model #5	2013	Family X	DRAM vendor C	yes
23	Model #6	2013	Family Y	DRAM vendor A	no
24	Model #6	2013	Family Y	DRAM vendor B	no
25	Model #6	2013	Family Y	DRAM vendor B	no
26	Model #6	2013	Family Y	DRAM vendor B	no
27	Model #6	2013	Family Y	DRAM vendor B	no
28	Model #7	2012	Family W	DRAM vendor D	no
29	Model #8	2014	Family Z	DRAM vendor A	no

15/29 Machines were vulnerable…

06

Rowhammer defenses

Rowhammer detection

- Software binary analysis

Rowhammer detection

- Software binary analysis

Rowhammer neutralization

- *G-CATT
  - ✓ Isolate user space / kernel space in physical memory
  - ✓ attacker cannot exploit bit flips in kernel memory

* "CAn't Touch This: Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory", F.Brasser et al. (2017.08)

Rowhammer detection

- Software binary analysis

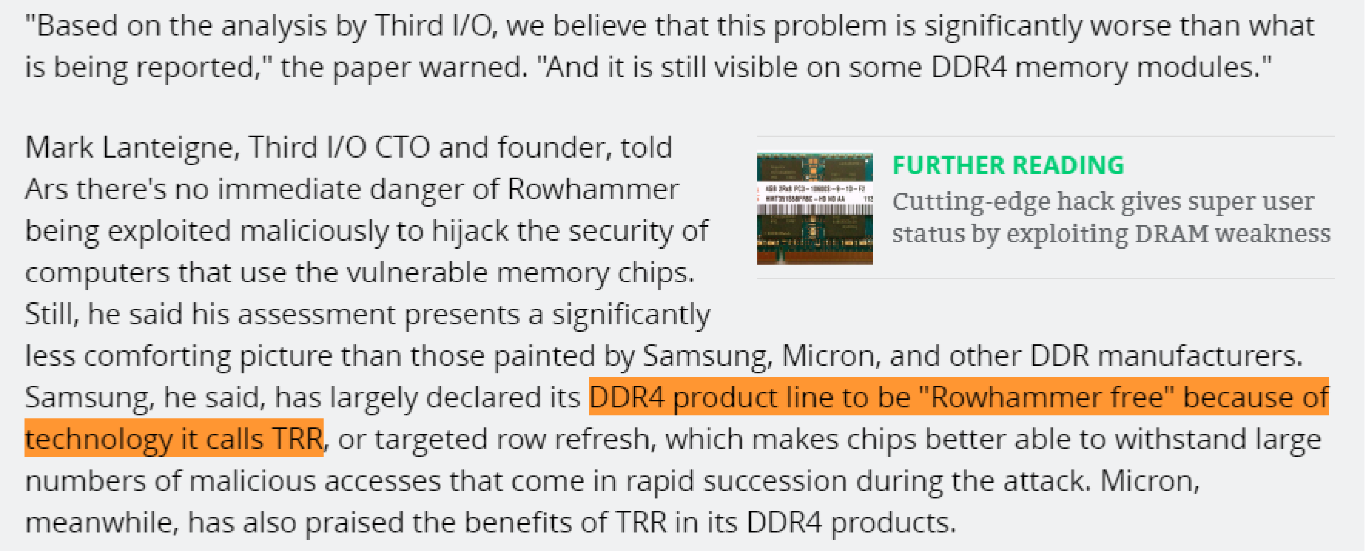
Rowhammer neutralization

- *G-CATT
  - ✓ Isolate user space / kernel space in physical memory
  - ✓ attacker cannot exploit bit flips in kernel memory

Rowhammer elimination

- TRR (Target Row Refresh) : Identify frequently accessed DRAM addresses
- tREFI (time of REfresh Interval) → e.g. Intel Skylake, Kaby lake
- ECC memory (Error Correcting Code)

* "CAn't Touch This: Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory", F.Brasser et al. (2017.08)



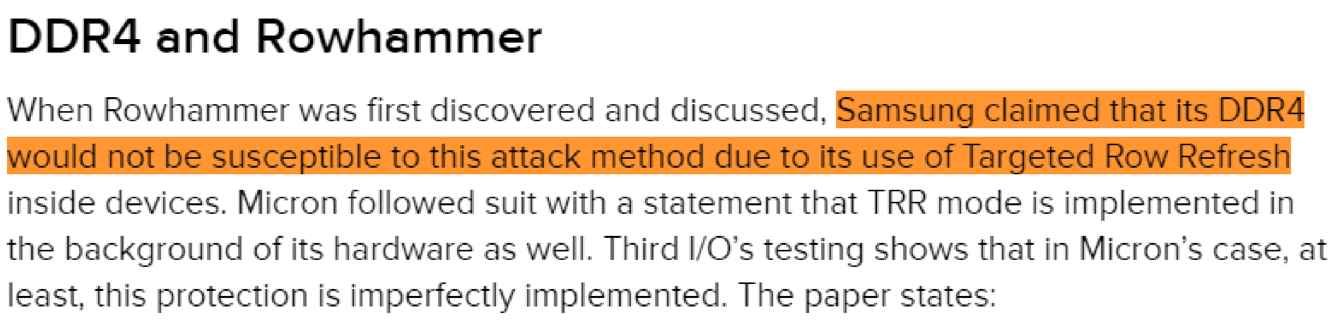
"Based on the analysis by Third I/O, we believe that this problem is significantly worse than what is being reported," the paper warned. "And it is still visible on some DDR4 memory modules."

Mark Lanteigne, Third I/O CTO and founder, told Ars there's no immediate danger of Rowhammer being exploited maliciously to hijack the security of computers that use the vulnerable memory chips. Still, he said his assessment presents a significantly less comforting picture than those painted by Samsung, Micron, and other DDR manufacturers. Samsung, he said, has largely declared its DDR4 product line to be "Rowhammer free" because of technology it calls TRR, or targeted row refresh, which makes chips better able to withstand large numbers of malicious accesses that come in rapid succession during the attack. Micron, meanwhile, has also praised the benefits of TRR in its DDR4 products.

FURTHER READING
Cutting-edge hack gives super user status by exploiting DRAM weakness

✓ Isolate

✓ attacke



DDR4 and Rowhammer

When Rowhammer was first discovered and discussed, Samsung claimed that its DDR4 would not be susceptible to this attack method due to its use of Targeted Row Refresh inside devices. Micron followed suit with a statement that TRR mode is implemented in the background of its hardware as well. Third I/O's testing shows that in Micron's case, at least, this protection is imperfectly implemented. The paper states:

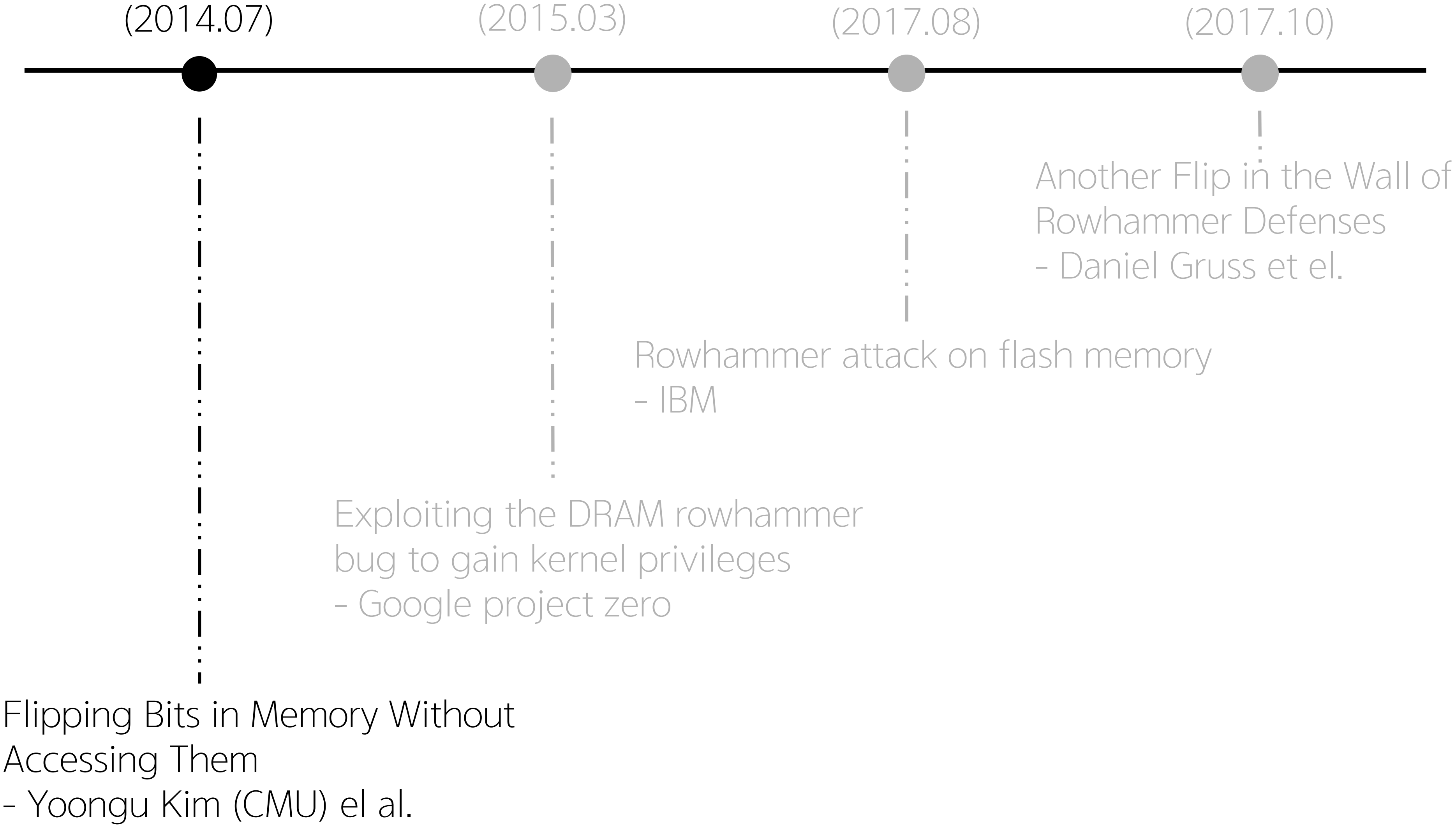
Rowhamm

- TRR (Target Row Refresh) : Identify frequently accessed DRAM addresses
- tREFI (time of REfresh Interval) → e.g. Intel Skylake, Kaby lake
- ECC memory (Error Correcting Code)

* https://www.extremetech.com/extreme/224860-new-paper-alleges-servers-some-ddr4-dram-still-vulnerable-to-critical-rowhammer-attack
** https://arstechnica.com/information-technology/2016/03/once-thought-safe-ddr4-memory-shown-to-be-vulnerable-to-rowhammer/

07

Conclusion & Recent study



(2014.07)

Flipping Bits in Memory Without Accessing Them
- Yoongu Kim (CMU) el al.

(2015.03)

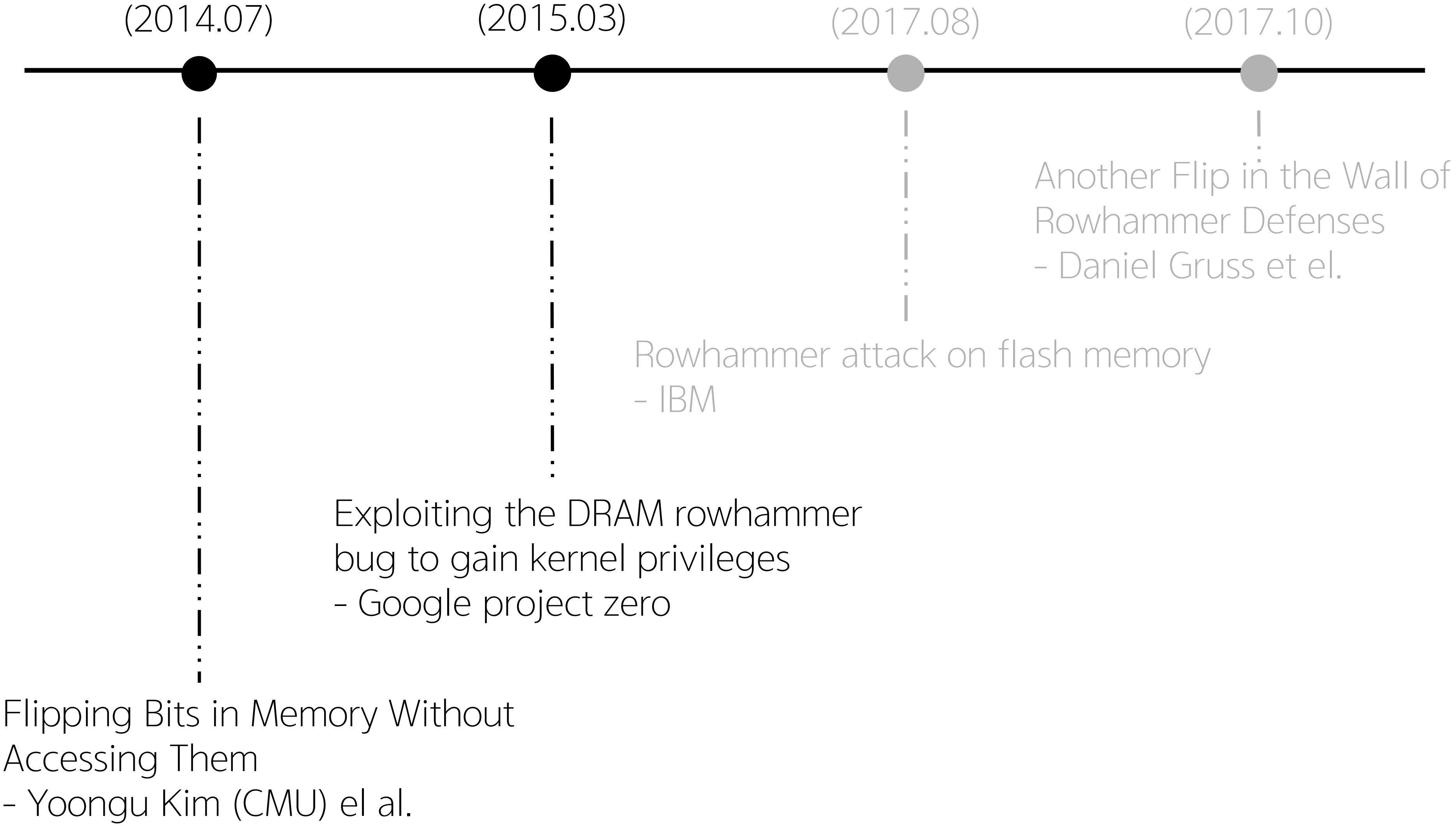
Exploiting the DRAM rowhammer bug to gain kernel privileges
- Google project zero

(2017.08)

Rowhammer attack on flash memory
- IBM

(2017.10)

Another Flip in the Wall of Rowhammer Defenses
- Daniel Gruss et el.



(2014.07)

Flipping Bits in Memory Without Accessing Them
- Yoongu Kim (CMU) el al.

(2015.03)

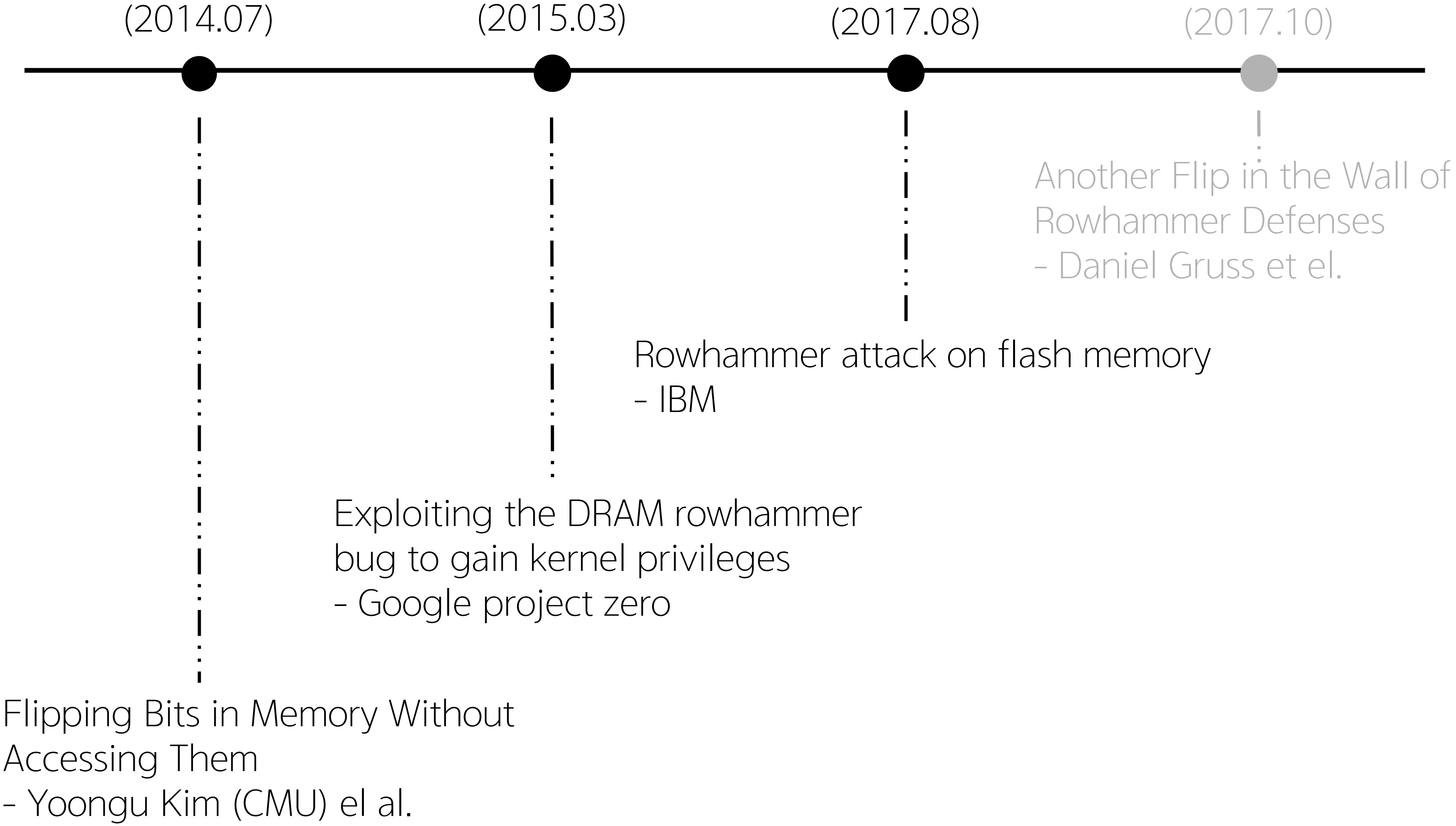
Exploiting the DRAM rowhammer bug to gain kernel privileges
- Google project zero

(2017.08)

Rowhammer attack on flash memory
- IBM

(2017.10)

Another Flip in the Wall of Rowhammer Defenses
- Daniel Gruss et el.



(2014.07)

Flipping Bits in Memory Without Accessing Them
- Yoongu Kim (CMU) el al.

(2015.03)

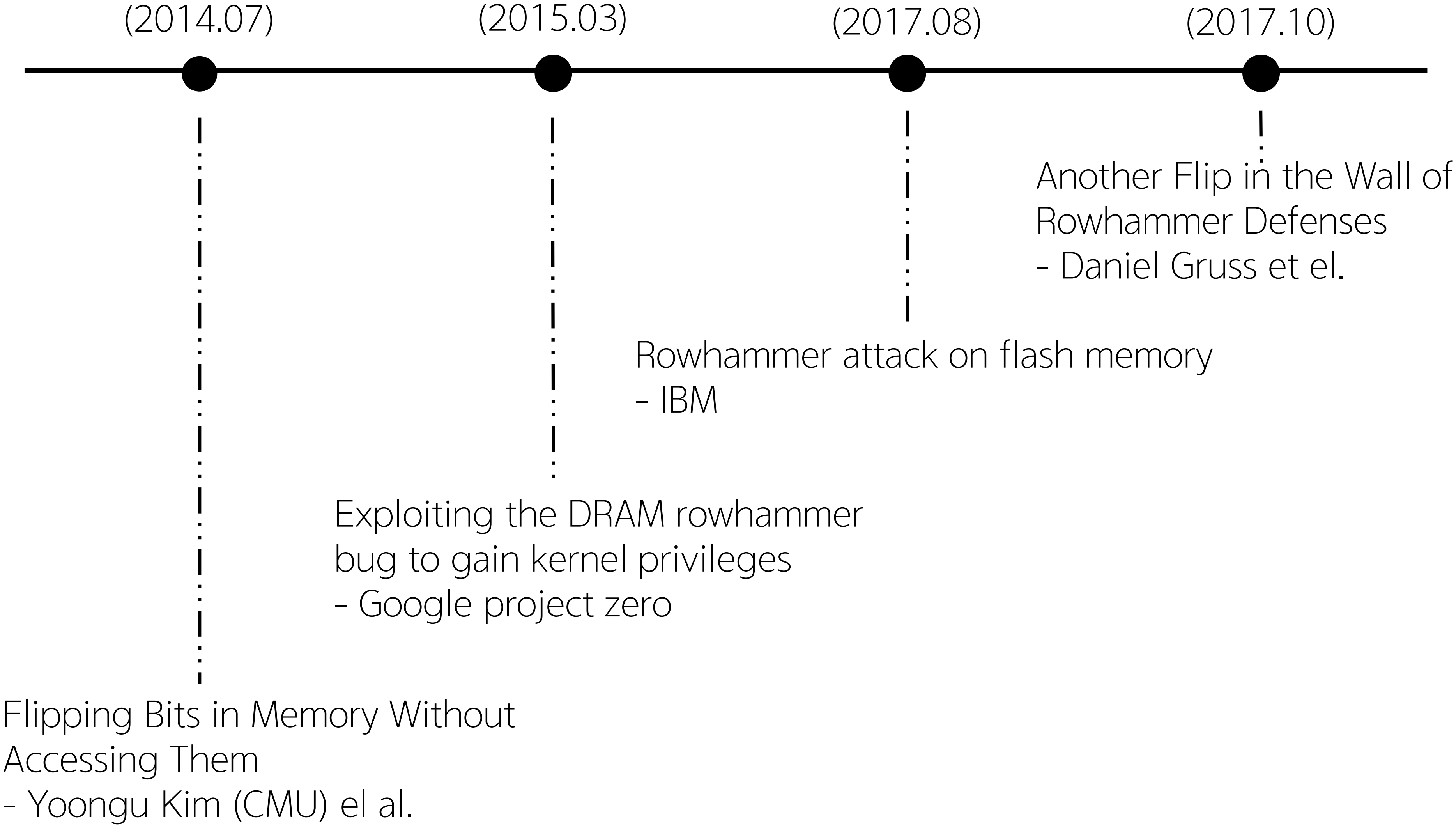
Exploiting the DRAM rowhammer bug to gain kernel privileges
- Google project zero

(2017.08)

Rowhammer attack on flash memory
- IBM

(2017.10)

Another Flip in the Wall of Rowhammer Defenses
- Daniel Gruss et el.



(2014.07)

Flipping Bits in Memory Without Accessing Them
- Yoongu Kim (CMU) el al.

(2015.03)

Exploiting the DRAM rowhammer bug to gain kernel privileges
- Google project zero

(2017.08)

Rowhammer attack on flash memory
- IBM

(2017.10)

Another Flip in the Wall of Rowhammer Defenses
- Daniel Gruss et el.

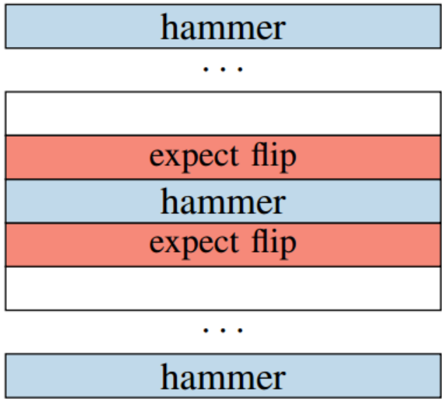
(2014.07)

(2015.03)

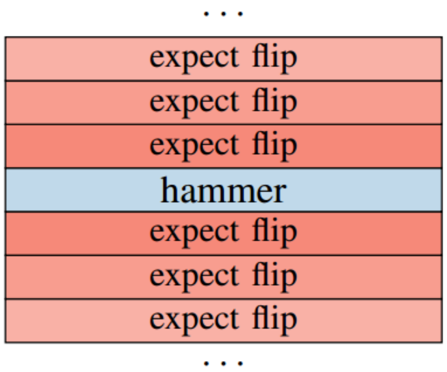
(2017.08)

(2017.10)

Another Flip in the Wall of Rowhammer Defenses
- Daniel Gruss et el.



▲ Ordinary rowhammer



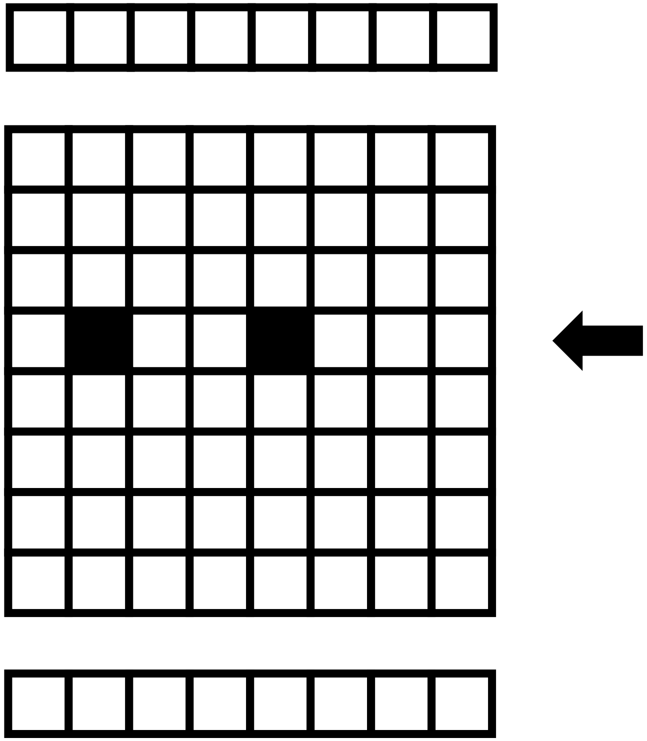
▲ One-location hammering

08

Future work

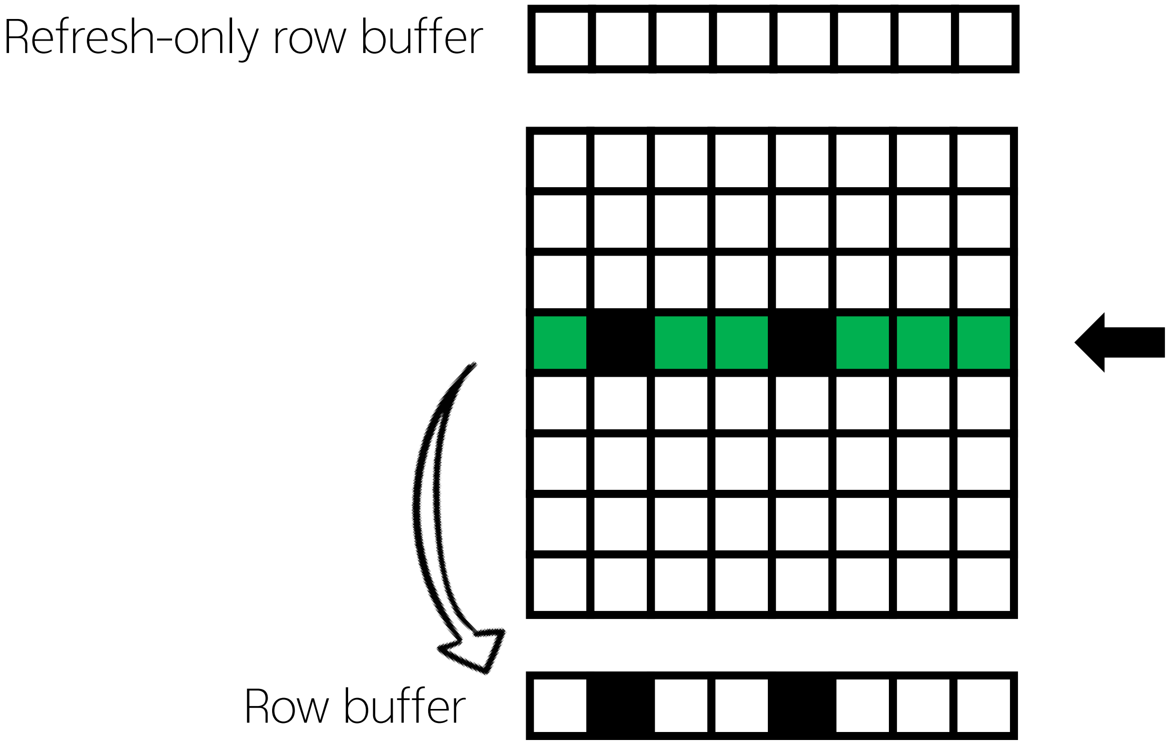
It might be a good mitigation…

- Arrange Refresh-only row buffer



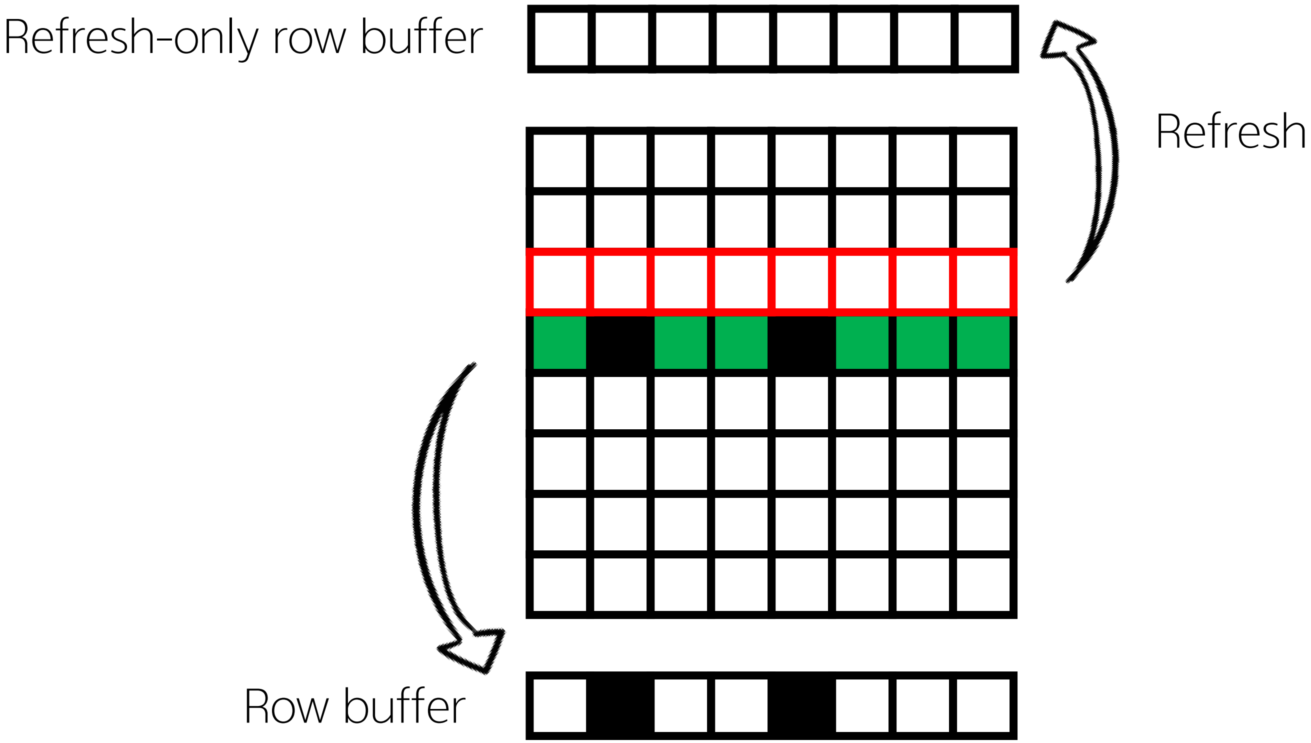
It might be a good mitigation…

- Arrange Refresh-only row buffer



It might be a good mitigation…

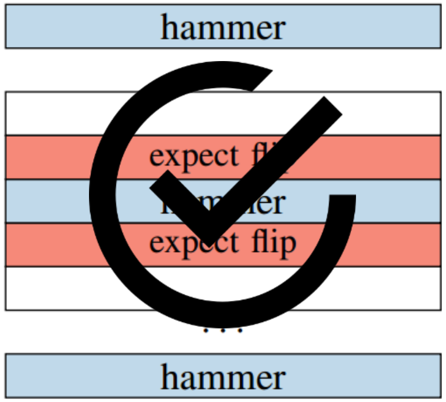
- Arrange Refresh-only row buffer



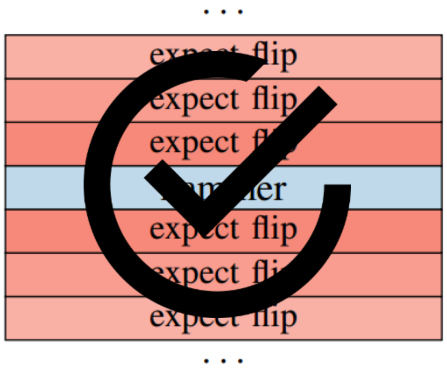
It might be a good mitigation…

- Arrange Refresh-only row buffer

Refresh-only row buffer



▲ Ordinary rowhammer



▲ One-location hammering

Q/A

THANK
YOU