black hat EUROPE 2022

DECEMBER 7-8, 2022

BRIEFINGS

Perfect Spray: A Journey From Finding a New Type of Logical Flaw at Linux Kernel To Developing a New Heap Exploitation Technique

Yoochan Lee[*], Byoungyoung Lee[*], Yuseok Jeon[†], Jinhan Kwak[*], and Junesoo Kang[†]

[*]Seoul National University

[†]Ulsan National Institute of Science and Technology (UNIST)

Short Bio

- Name : Yoochan Lee
- Current : PhD student @ Seoul National University (2019 ~)
- Affiliation : @CompSec Lab, @Team GYG
- Email : yoochan10@snu.ac.kr
- Interest : OS Security, Bug Hunting

Intro

- Exploit Reliability

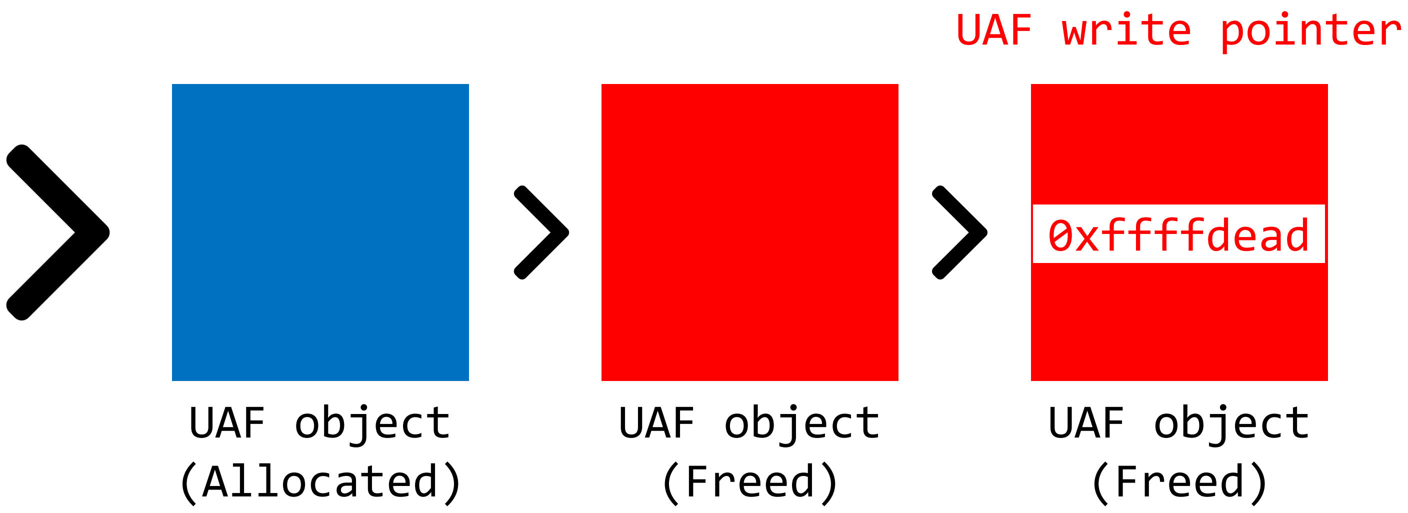
Exploit Failed

Alert is logged

Attack can be detected

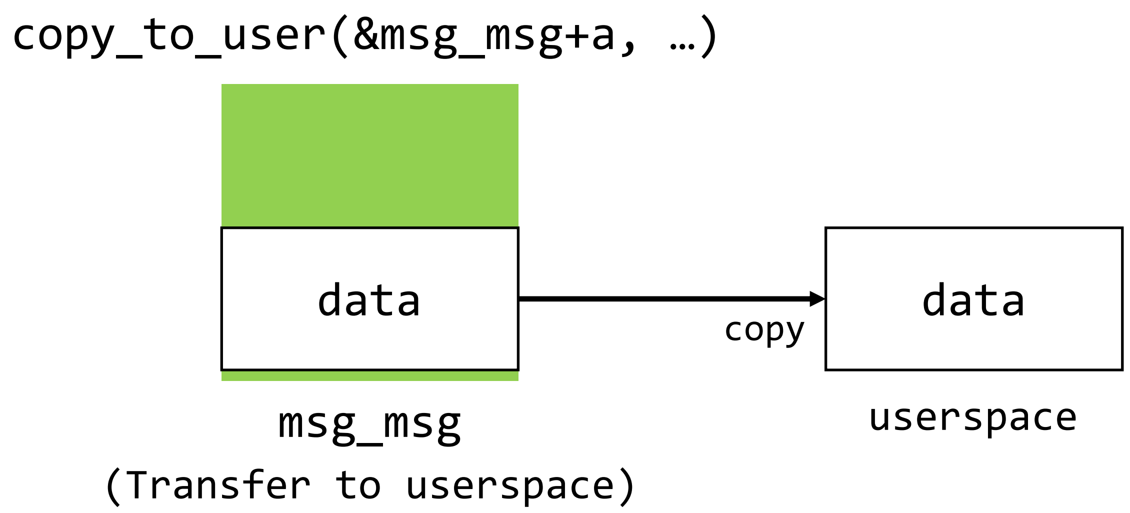
Motiv

83bec29088…



Motiv

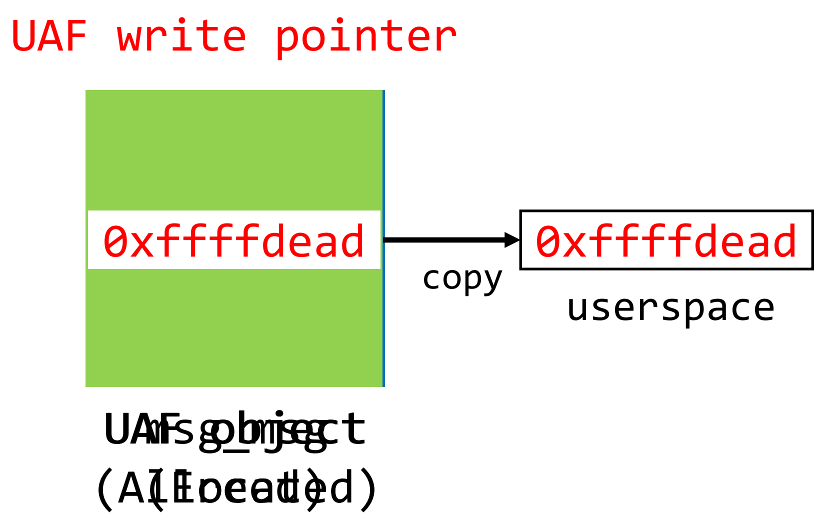
83bec29088…



copy_to_user(&msg_msg+a, …)

Motiv

83bec29088…



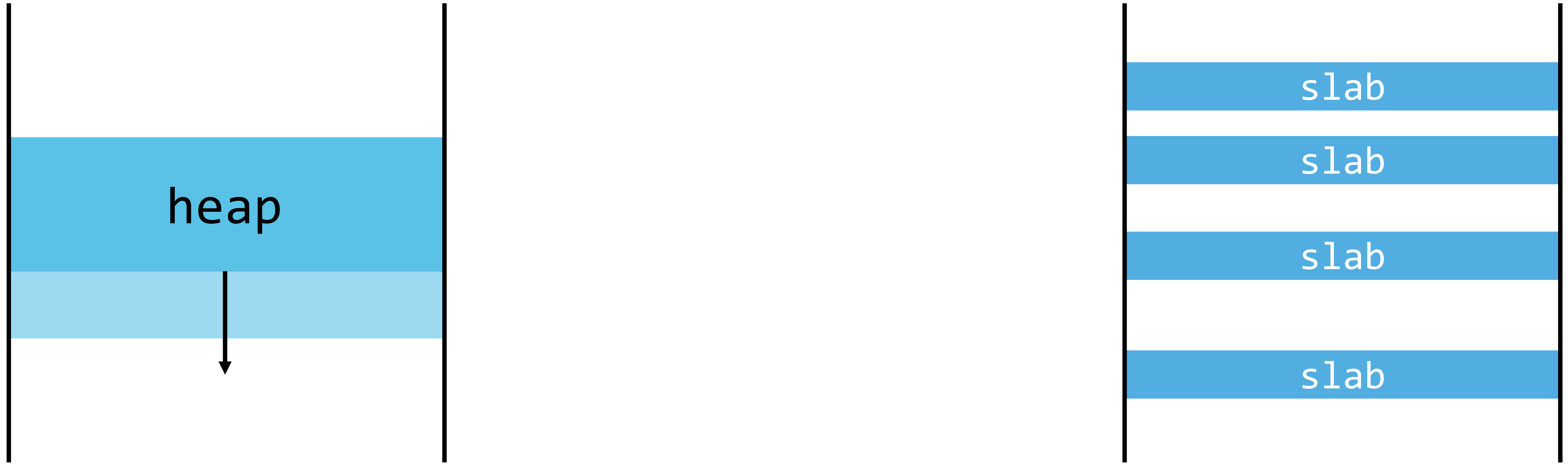
Motiv

83bec29088… › 13.70%

SLUB

SLAB **SLUB** SLOB

SLUB



User application

SLUB allocator

SLUB

Slab Cache

General Cache
> Specific size

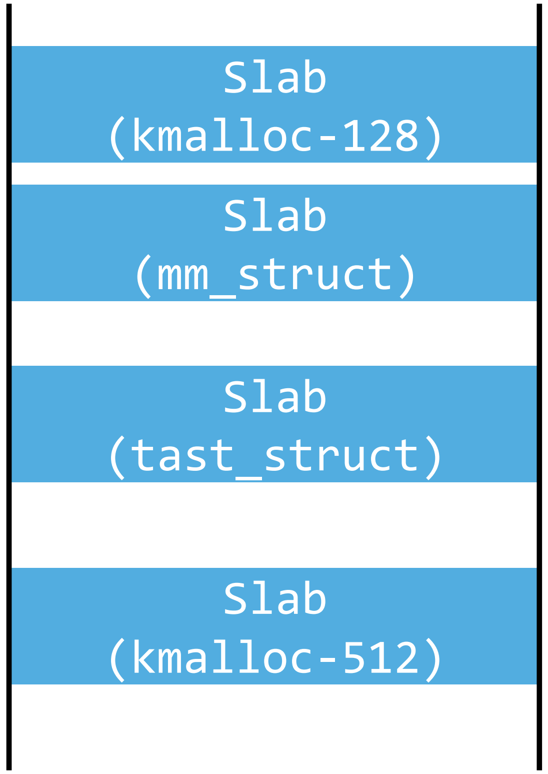
- kmalloc-32
- kmalloc-64
- Kmalloc-96

…

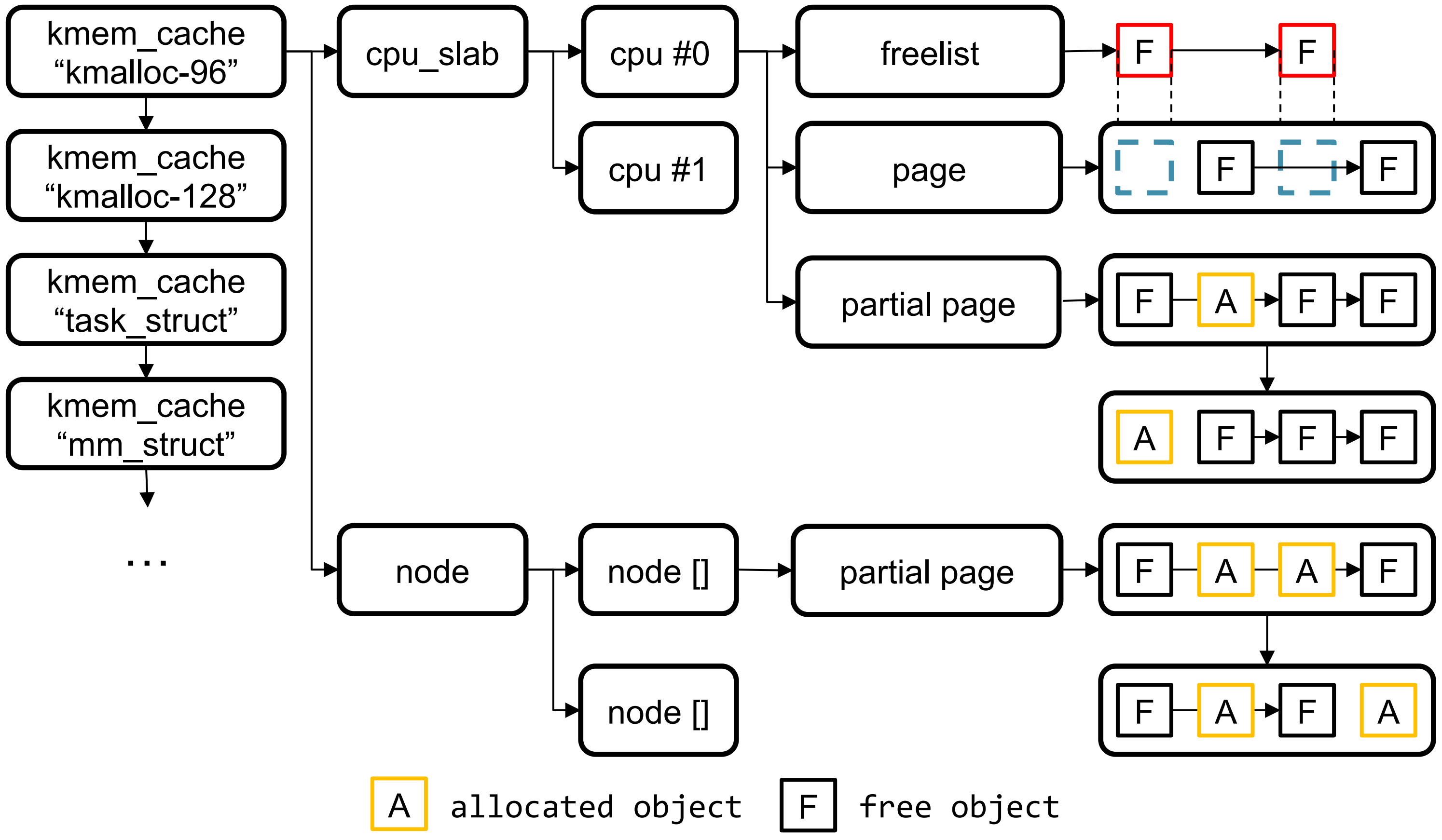
Special Cache
> Specific type

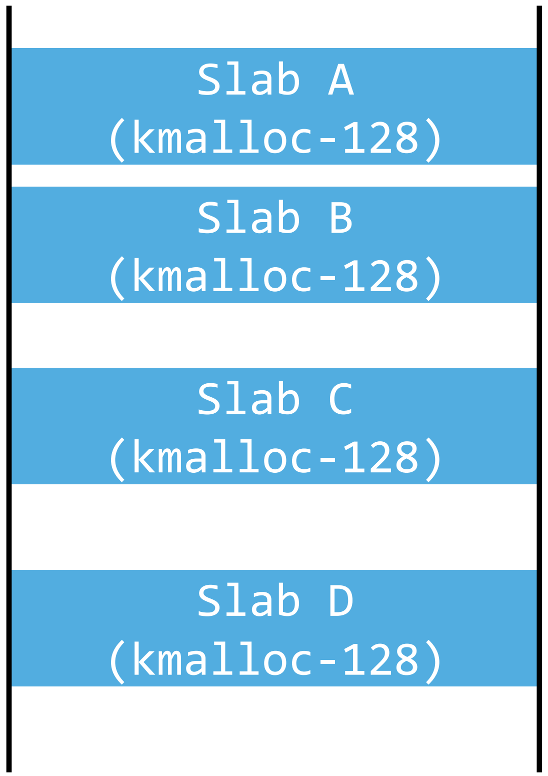
- task_struct
- mm_struct
- vm_area_struct

…

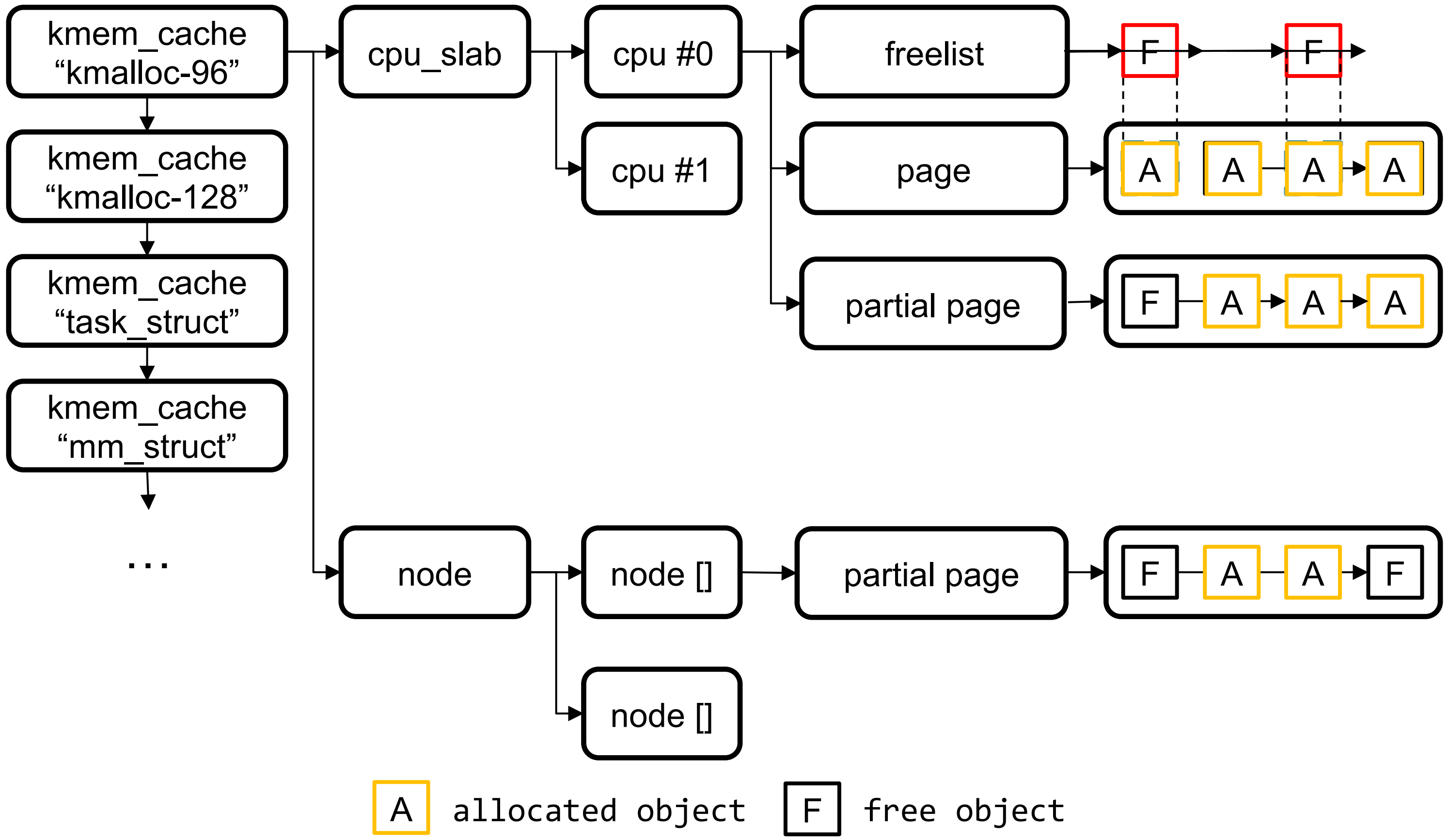


SLUB (Architecture)

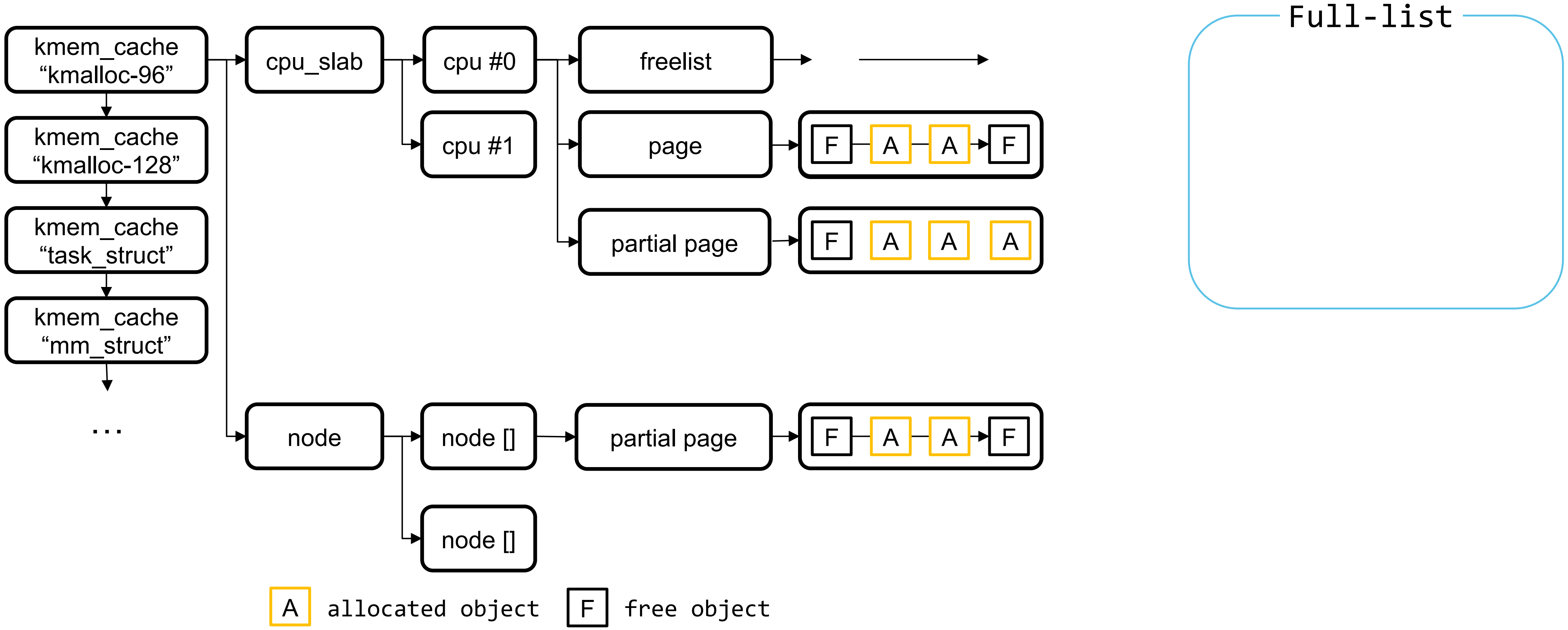




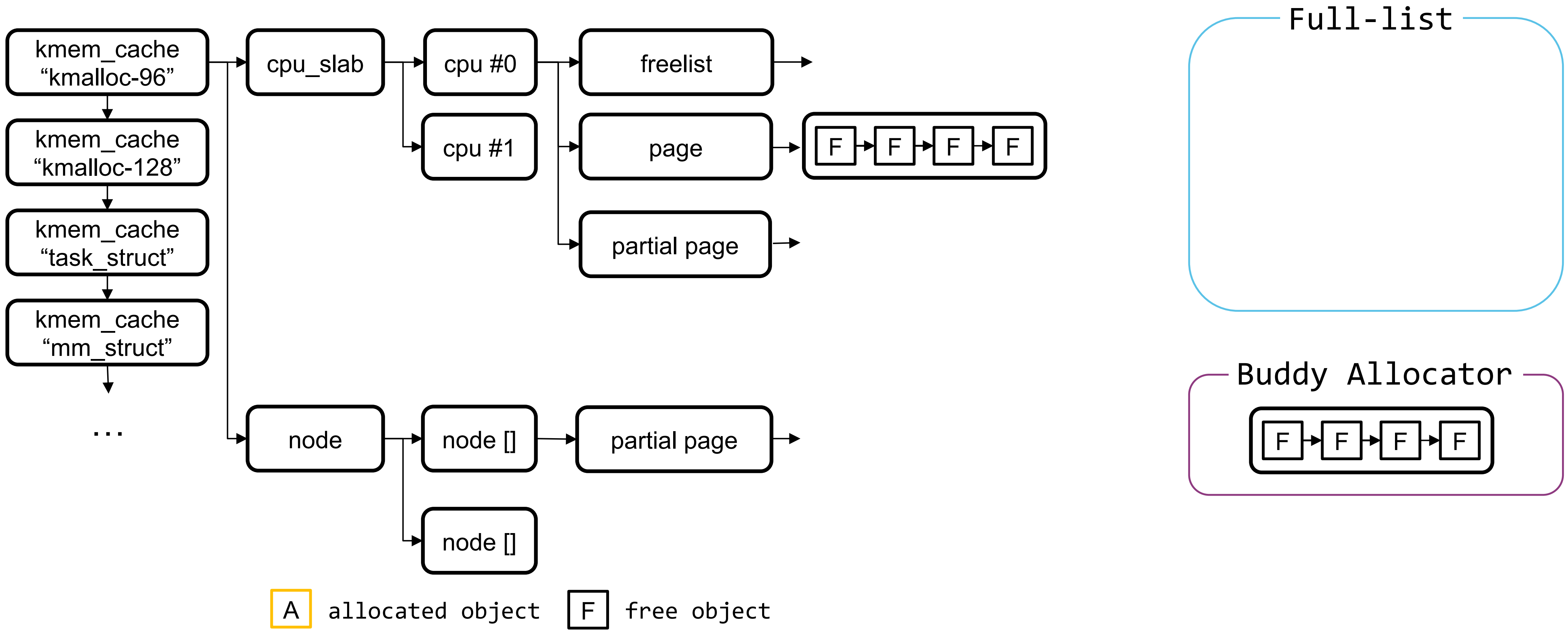
SLUB (Allocation mechanism)



SLUB (Allocation mechanism)



SLUB (Allocation mechanism)

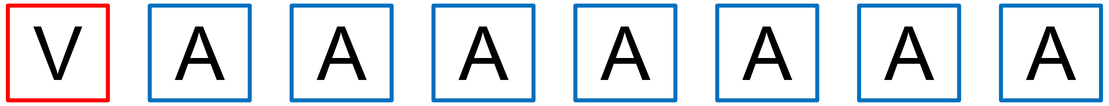


Back to Motiv

83bec29088…

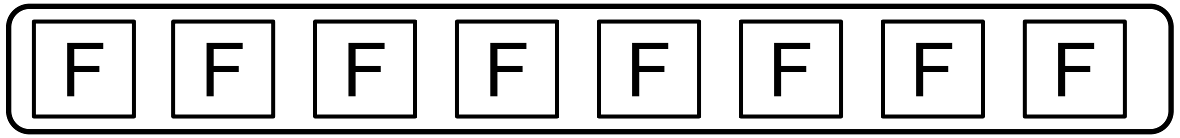
- kmalloc-4096

- Allocate 1 vuln, 7 add’l



V A A A A A A A

- 8 objects in slab



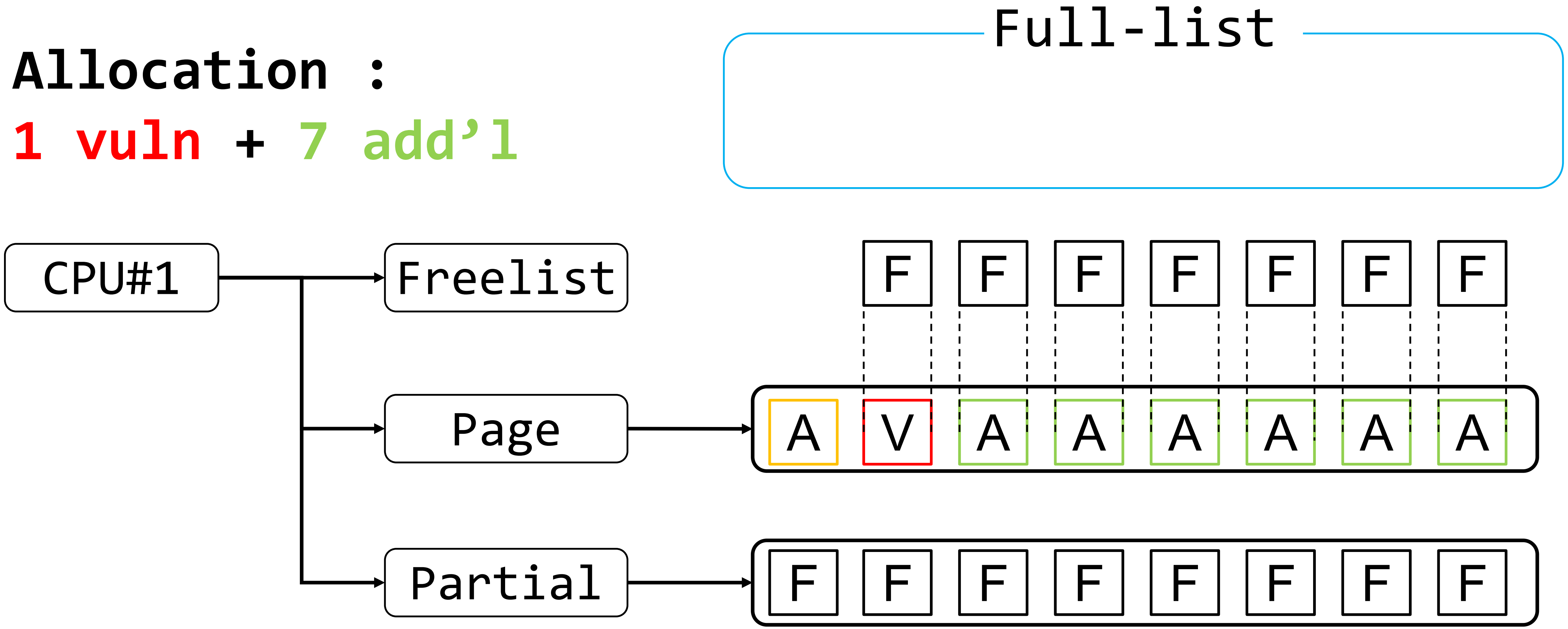
F F F F F F F F

kmalloc-4096 slab

Problem

Allocation :

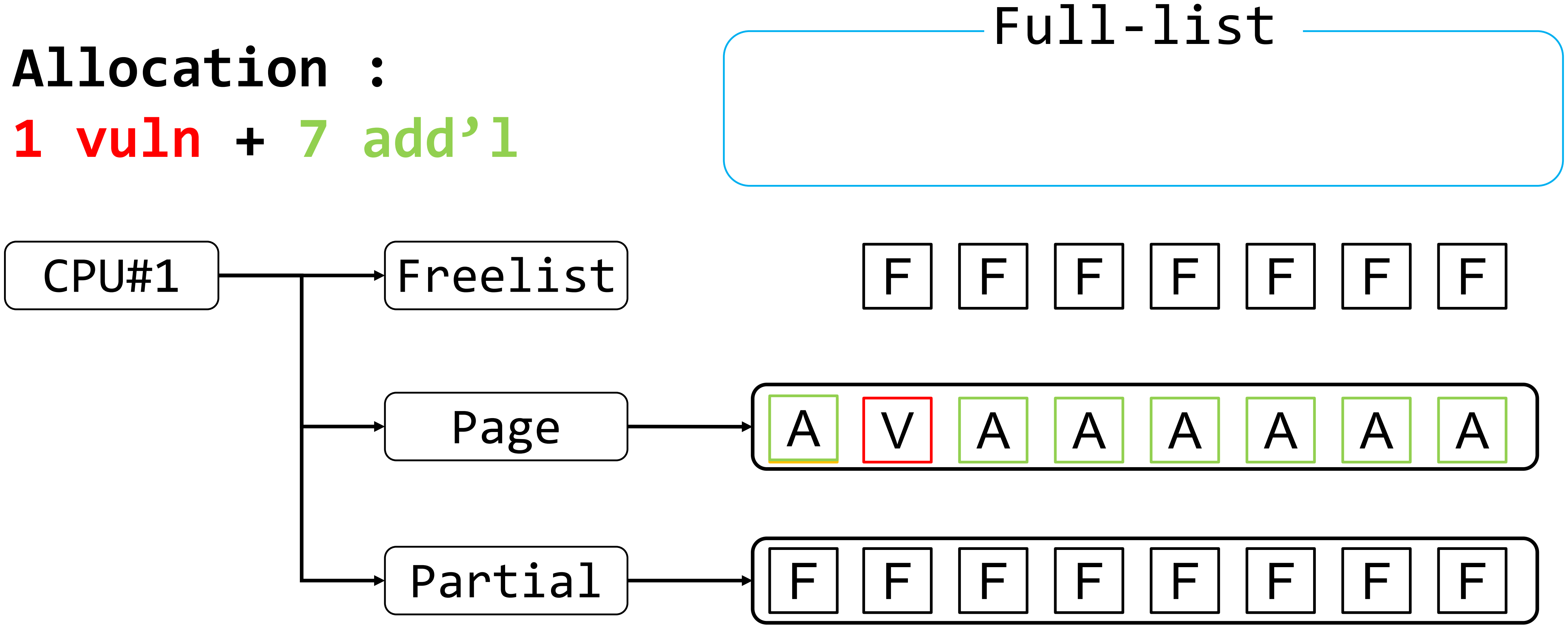
1 vuln + 7 add’l



Problem

Allocation :

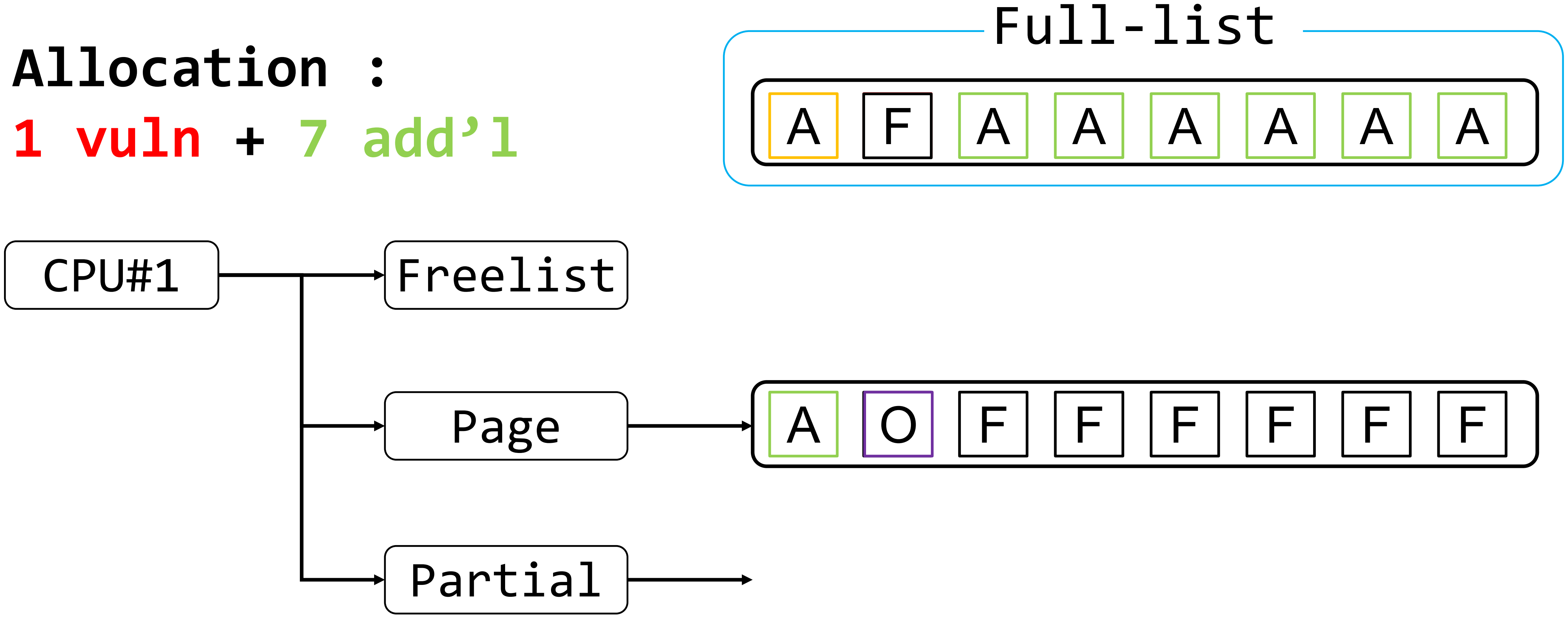
1 vuln + 7 add'l



Problem

Allocation :

1 vuln + 7 add’l



Summarize the problem

- Allocation : 1 vuln + 7 add’l
- # object in slab : 8 objects

⇒ CPU’s page is **changed** if slab contains at least one allocated object.

⇒ Overlapping object is allocated to wrong address.

To exploit

⇒ The CPU’s page should not use any object.

Research Question

Question

⇒ How to know the current slab is not used?

Idea

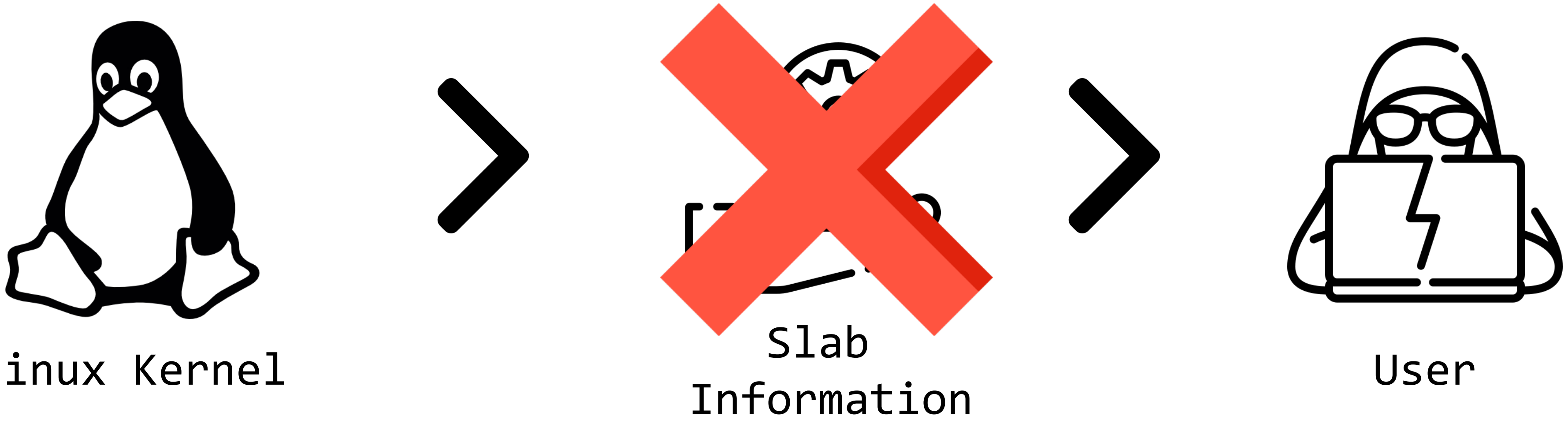
⇒ If the attacker know the **allocation status** of slab, the attacker can know current slab is not used

Another Question

⇒How to know the allocation status of slab?

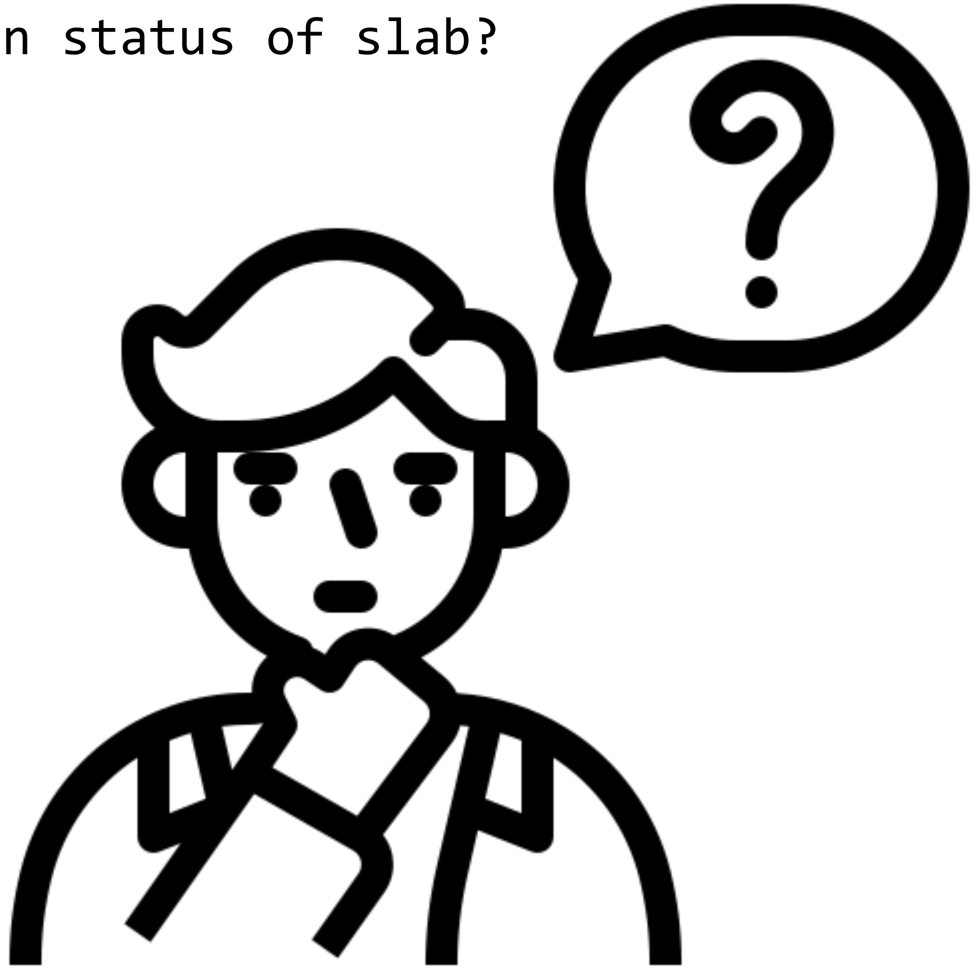
Research Question

⇒ How to know the allocation status of slab?



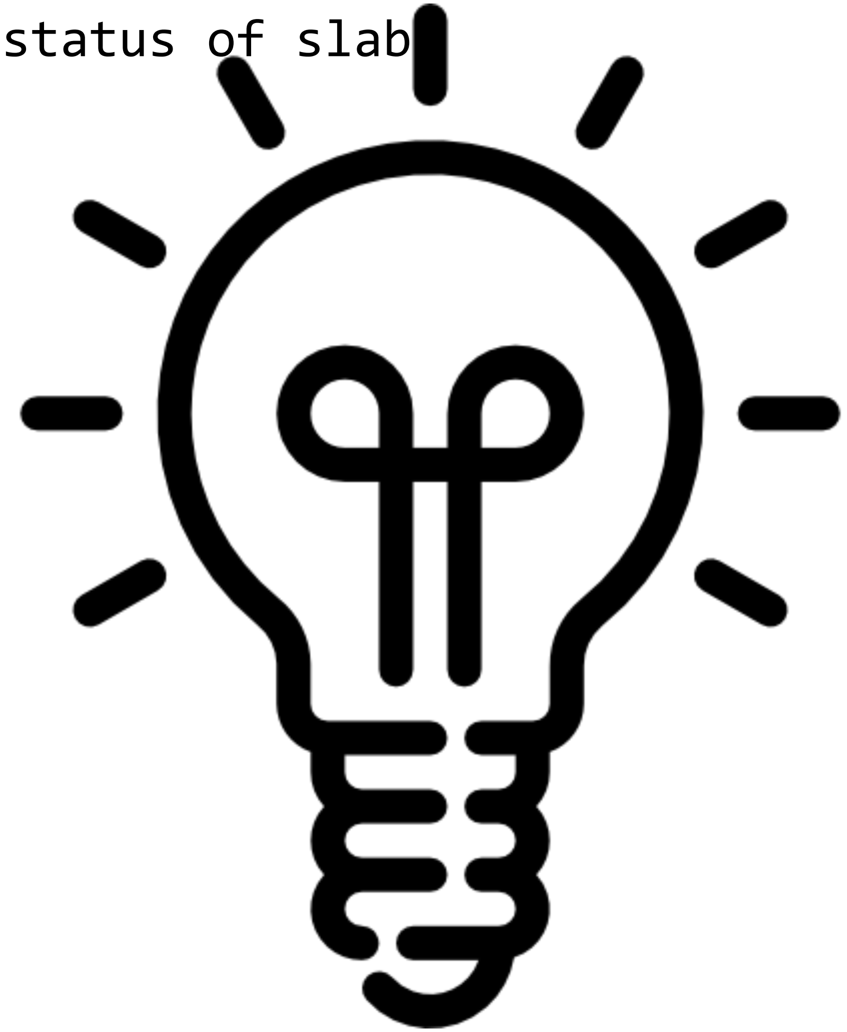
Research Question

⇒ How to know the allocation status of slab?



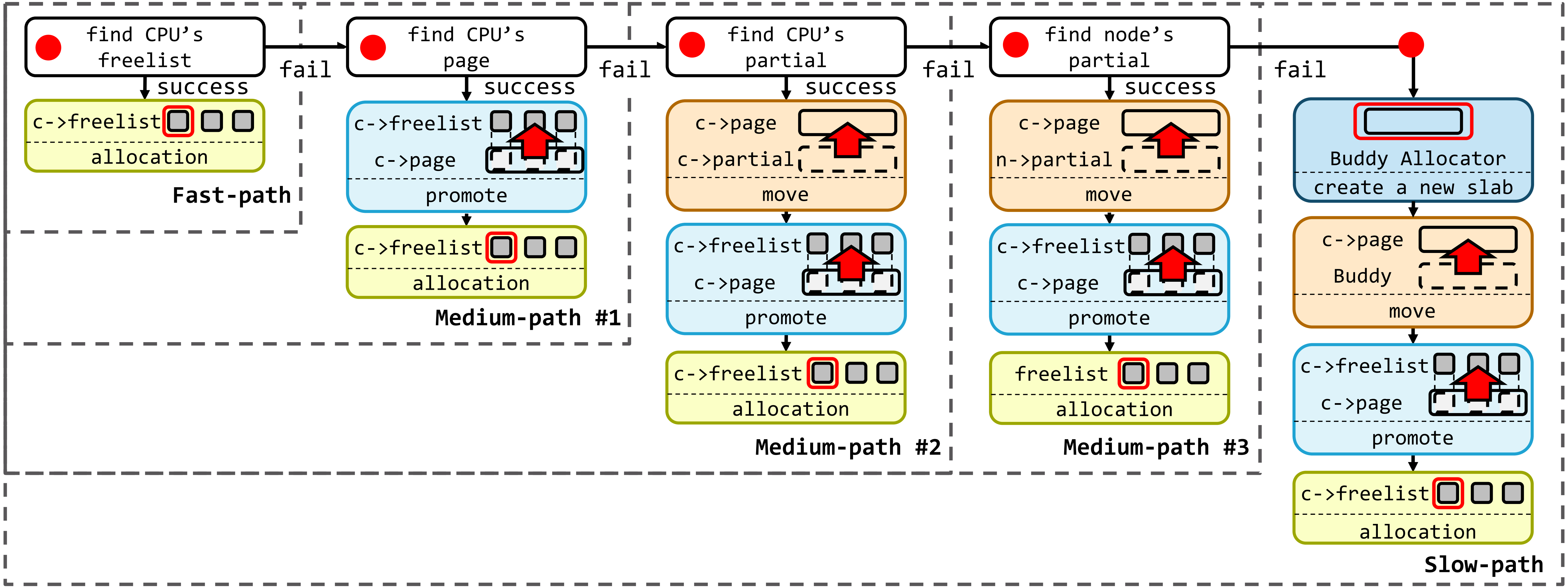
Research Question

⇒ How to know the allocation status of slab

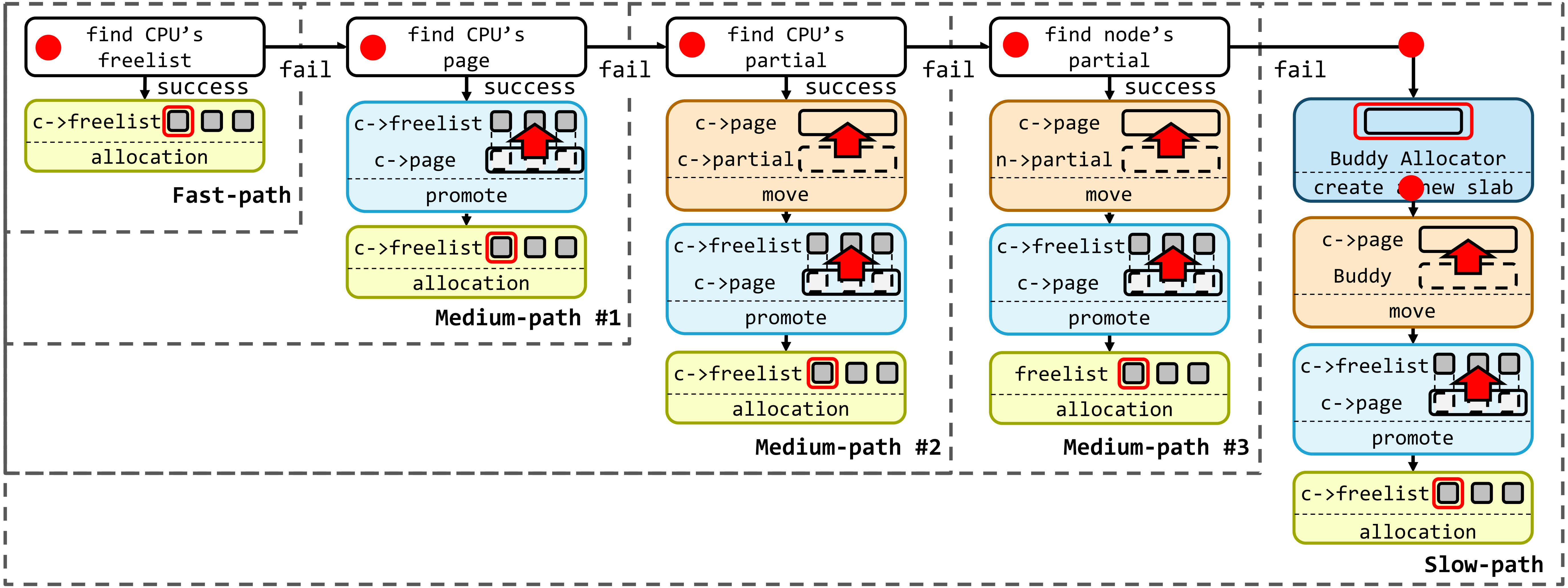


Back to SLUB

● ← Track down

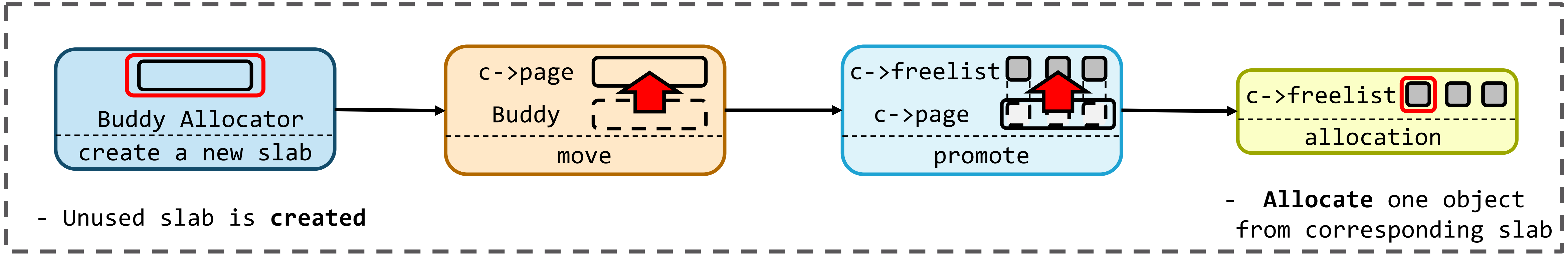


Back to SLUB


find CPU's freelist
success
c->freelist
allocation
Fast-path
fail
find CPU's page
success
c->freelist
c->page
promote
c->freelist
allocation
Medium-path #1
fail
find CPU's partial
success
c->page
c->partial
move
c->freelist
c->page
promote
c->freelist
allocation
Medium-path #2
fail
find node's partial
success
c->page
n->partial
move
c->freelist
c->page
promote
freelist
allocation
Medium-path #3
fail
Buddy Allocator
create new slab
c->page
Buddy
move
c->freelist
c->page
promote
c->freelist
allocation
Slow-path

Back to SLUB

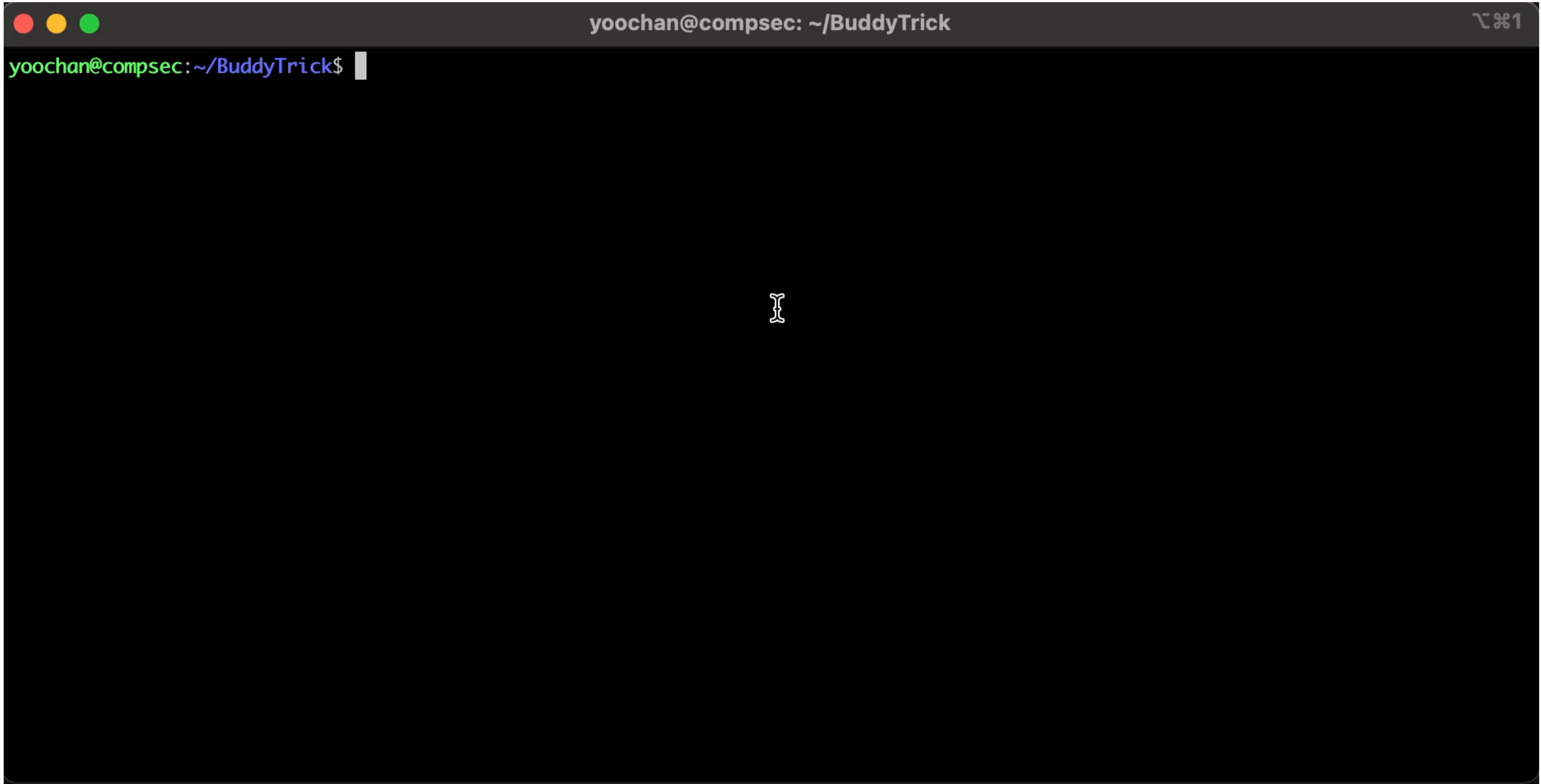
Slow-path



- Unused slab is **created**

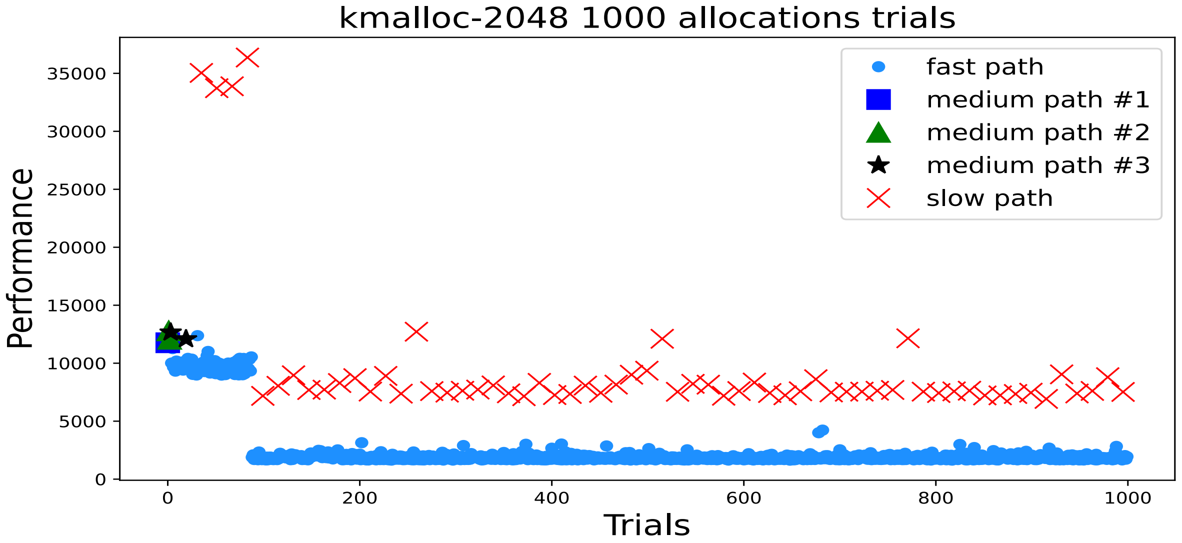
- **Allocate** one object from corresponding slab

Proof-Of-Concept

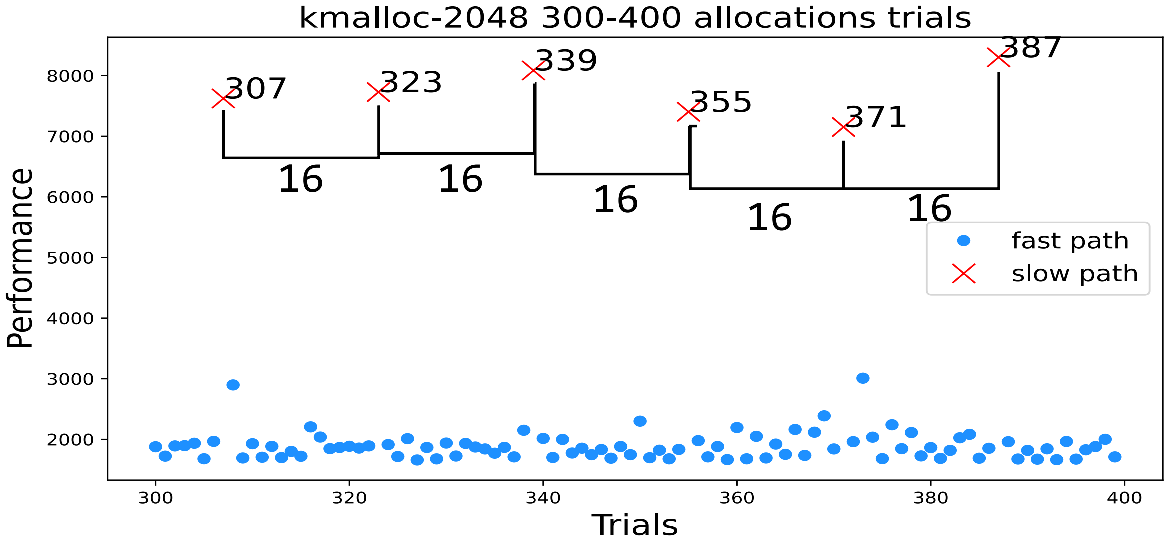


```
yoochan@compsec:~/BuddyTrick$ 
```

Proof-Of-Concept



(1) 0-1000 allocations



(2) 300-400 allocations

Pspray

Timing Side-Channel based Linux Kernel Heap Exploitation Technique

Pspray against UAF

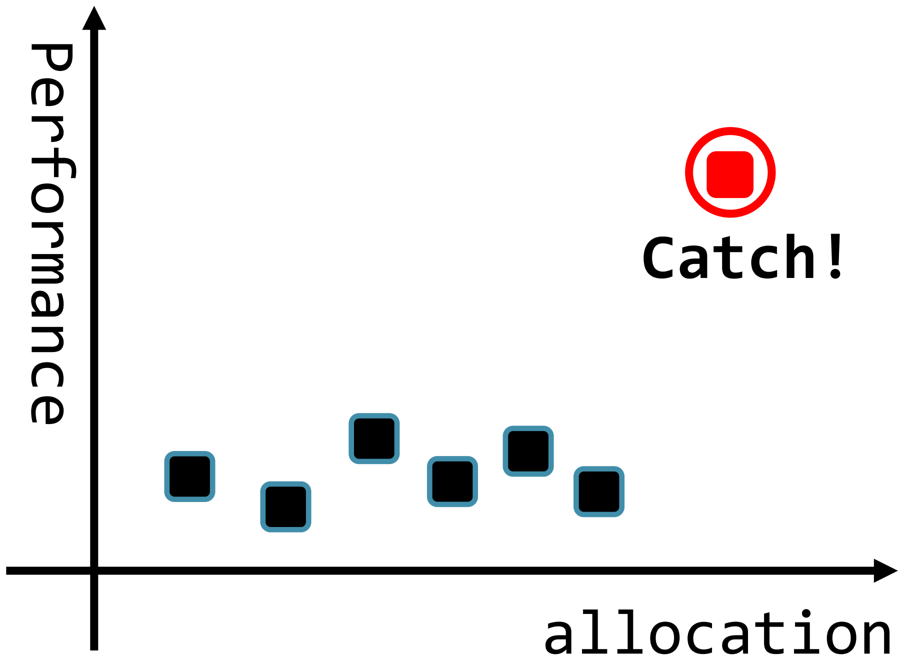
83bec29088…

- Allocate 1 vuln, 7 add’l
  - 8 objects in one slab

=> It needs unused slab

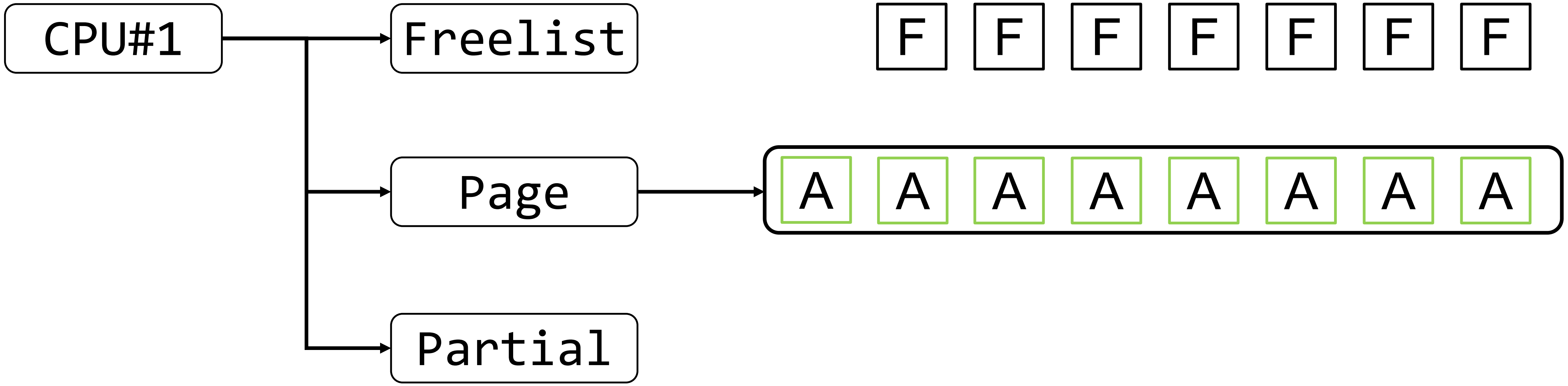
Pspray against UAF

1) do Pspray



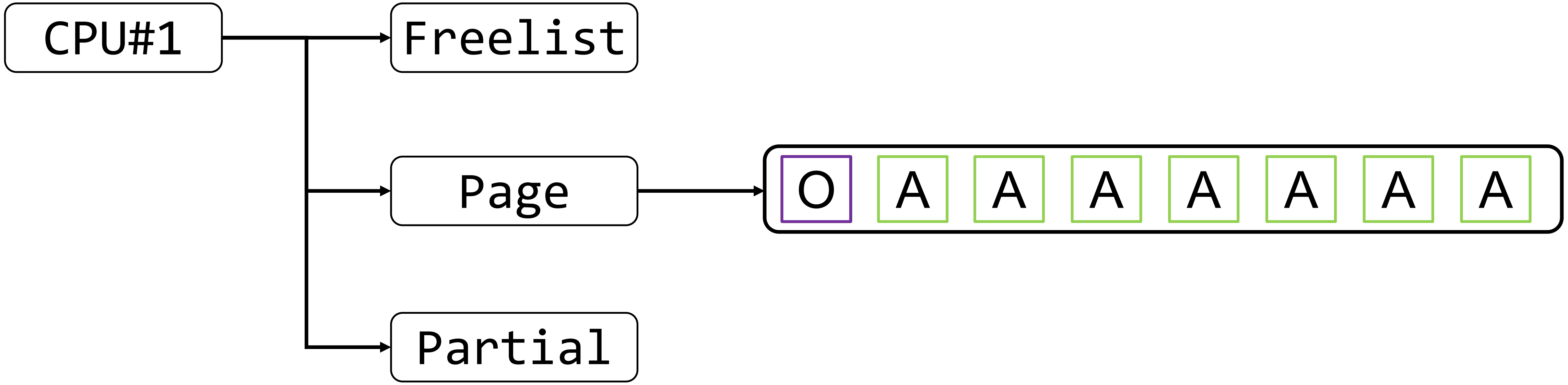
Pspray against UAF

2) Allocates 7 objects



Pspray against UAF

3) Allocates 1 vuln and 7 add’l



Pspray against UAF

4) Conclusion

13.70% > 98.16%

Pspray against UAF

4) Conclusion

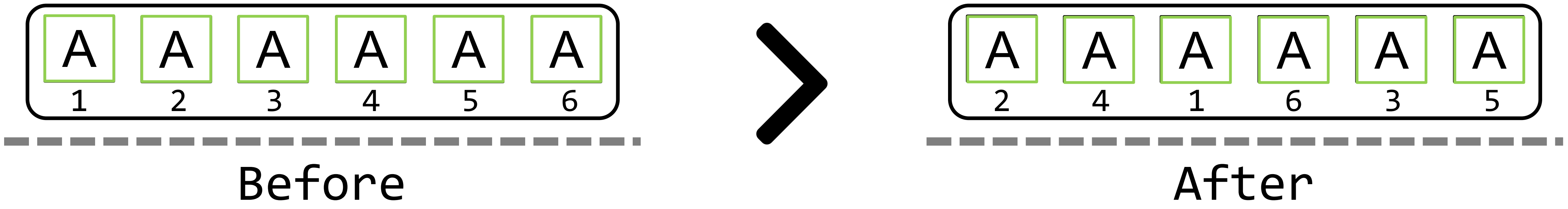
Vulns	Type	# of alloc	Baseline		**Pspray**
CVE-2019-2215	UAF	2	93.28%		100%⬆️
CVE-2018-6555	UAF	13	63.50%	>	99.94%⬆️
83bec2...	UAF	8	13.70%		98.16%⬆️
77e2cf...	UAF	1	95.74%		100%⬆️
CVE-2017-6074	DF	4	80.64%		100%⬆️
6b8d6b...	DF	1	96.28%		99.98%⬆️

Out-Of-Bounds

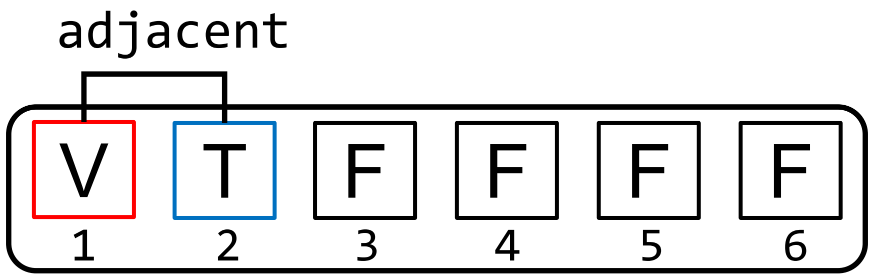
CVE-2017-7533	33.78%
CVE-2017-7184	21.18%
CVE-2016-6187	23.38%
CVE-2010-2959	39.60%

Out-Of-Bounds

SLAB Freelist Random

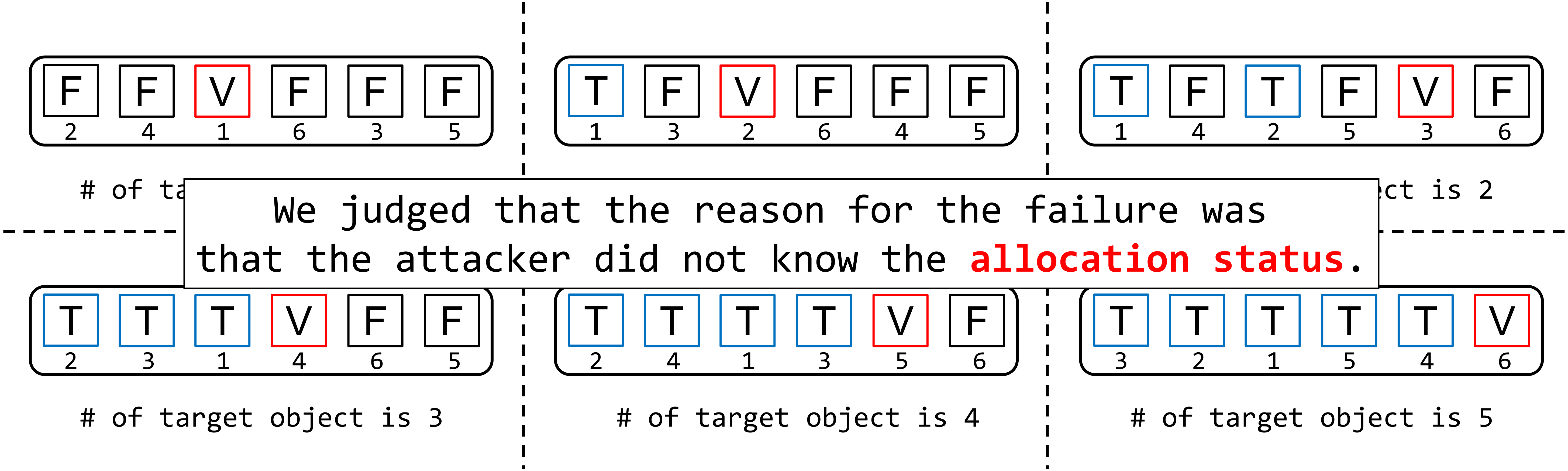


Out-Of-Bounds



- The vulnerable object and the target object must be adjacent

Out-Of-Bounds



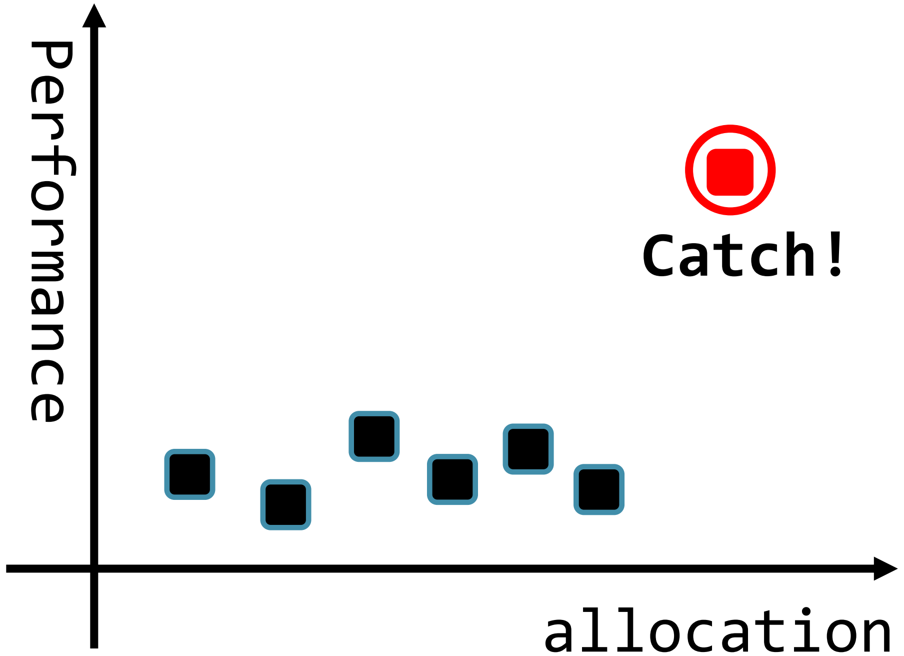
We judged that the reason for the failure was that the attacker did not know the **allocation status**.

Failed case of Out-Of-Bounds exploitation

- The attacker cannot know how many target object is allocated.

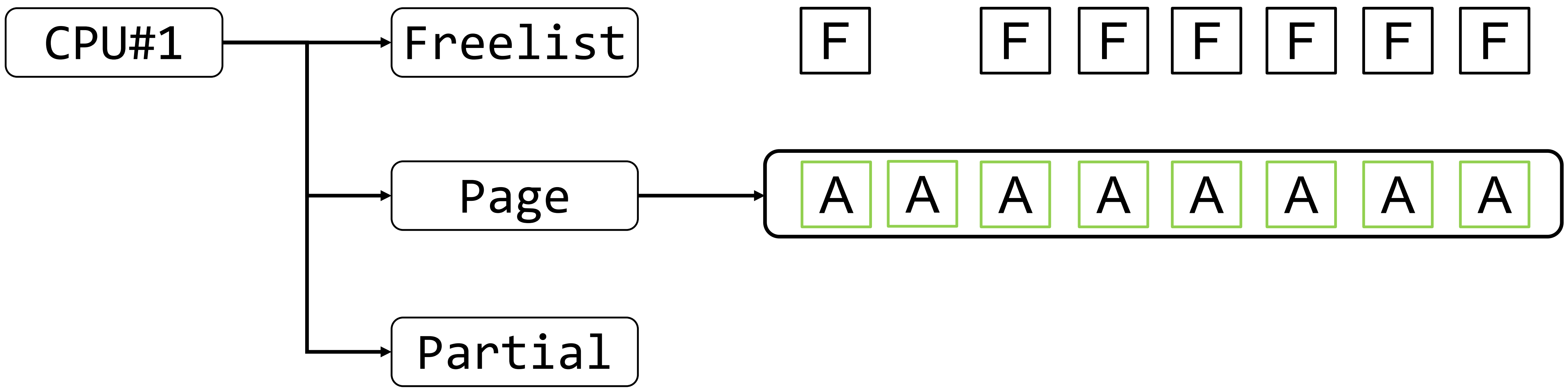
Pspray against OOB

1) do Pspray



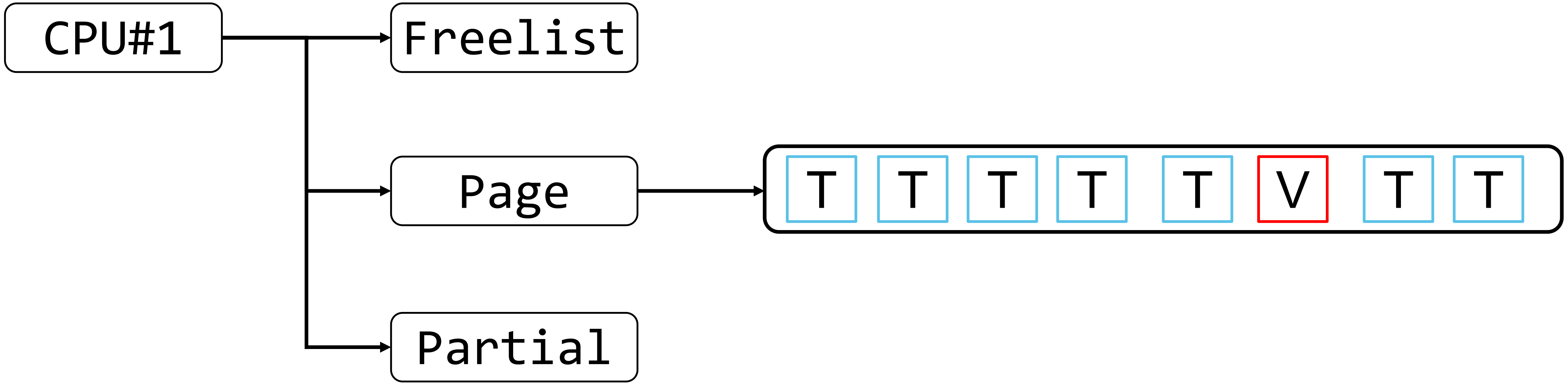
Pspray against OOB

2) Allocates 7 objects



Pspray against OOB

3) Allocates 7 target and 1 vuln



Pspray against OOB

4) Conclusion

Vulns	Type	Baseline		**Pspray**
CVE-2017-7533	OOB	33.78%	>	94.26%⬆
CVE-2017-7184	OOB	21.18%		96.52%⬆
CVE-2016-6187	OOB	23.38%		95.58%⬆
CVE-2010-2959	OOB	39.60%		94.80%⬆

Pspray against UAF

Problem:

- The change of CPU’s page during allocating vulnerable object

Solution:

- Using **Pspray**, we can circumvent the situation that CPU’s page is changed.

Pspray against OOB

Problem:

- Slab Freelist Random

Solution:

- Using **Pspray**, we can make the vulnerable object and target object adjacent.

Conclusion

- We introduce **Pspray**, which combines previous exploit techniques with timing side-channel attack.
- The utilization of **Pspray** is endless.
- There might be another logical flaws like **Pspray** in Linux kernel or else.

Q&A

yoochan10@snu.ac.kr