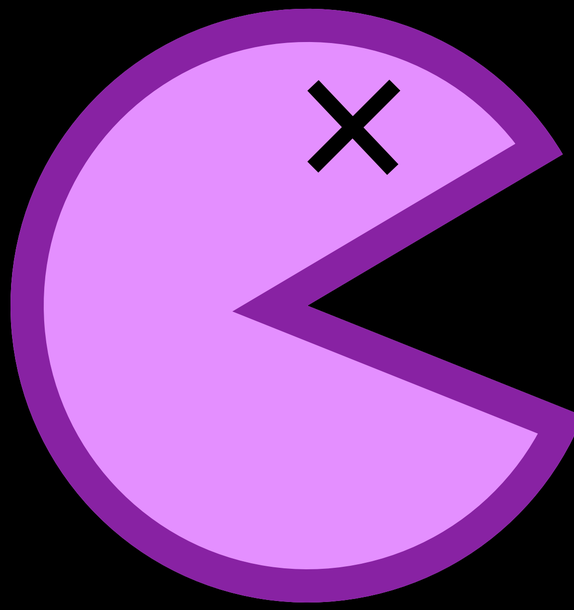
PA  MAN

ATTACKING ARM POINTER AUTHENTICATION WITH SPECULATIVE EXECUTION

Joseph Ravichandran*, Weon Taek Na*, Jay Lang, Mengjia Yan

*Both authors contributed equally to this work.

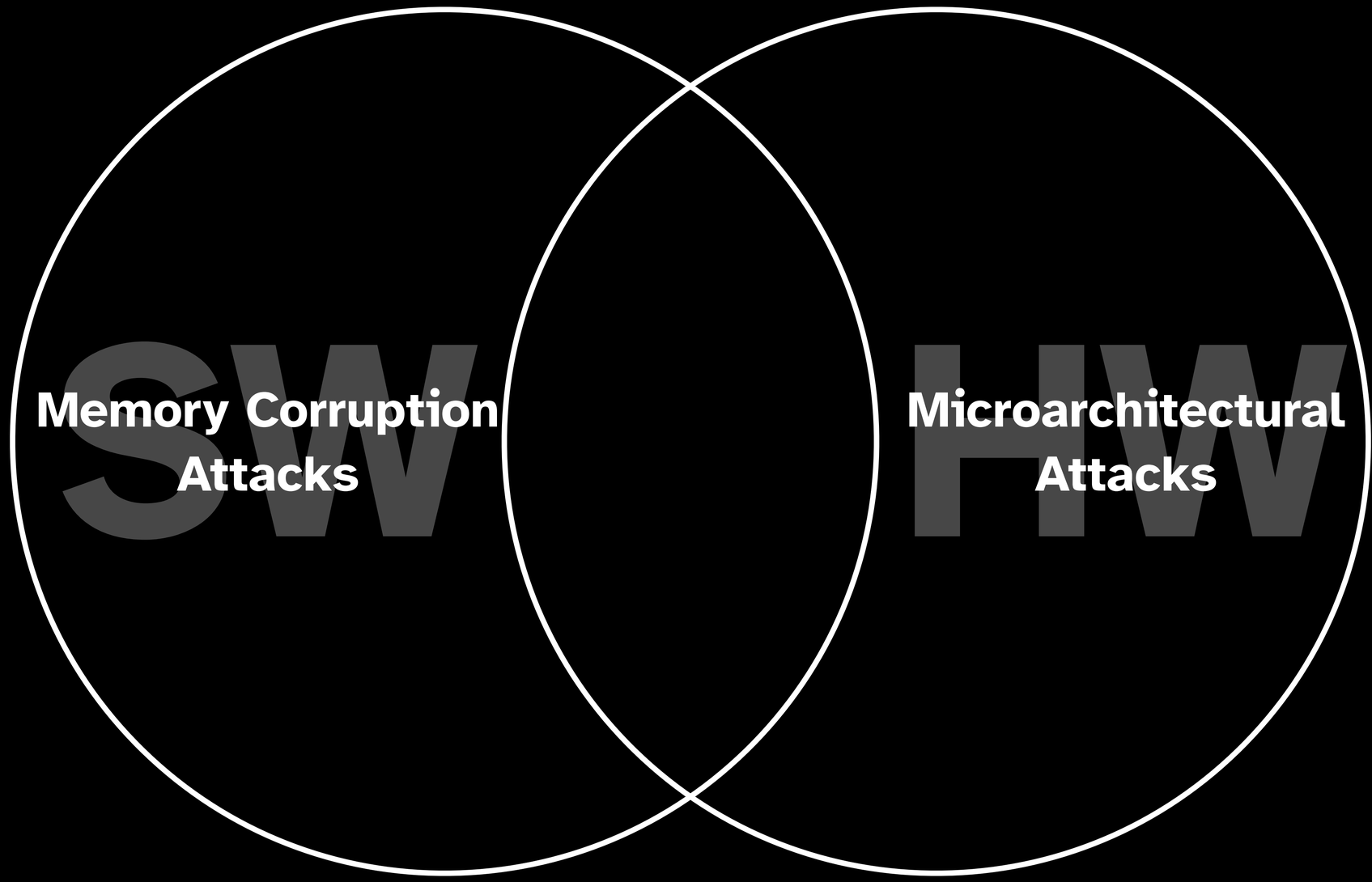
MIT CSAIL

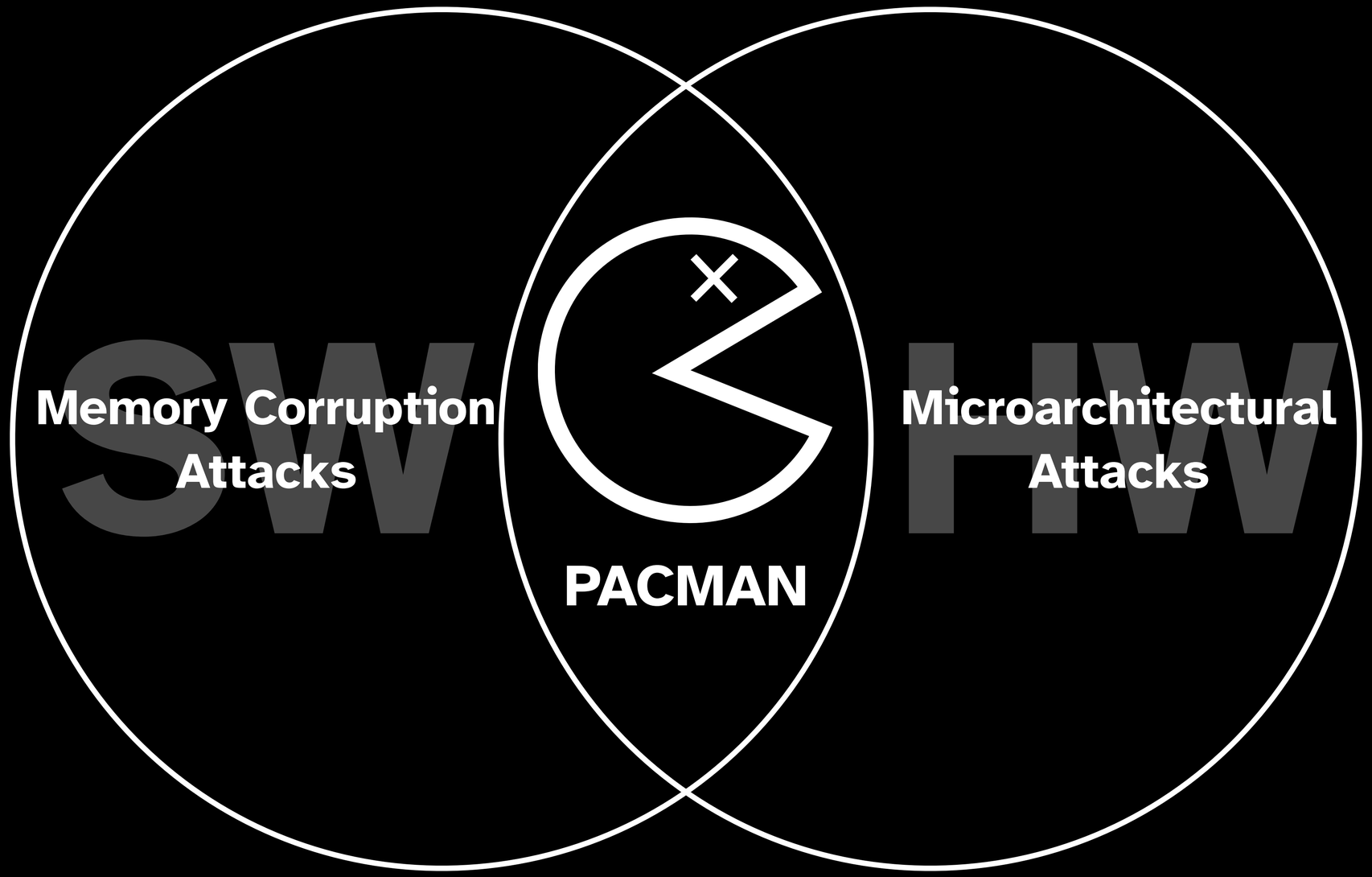
compute. collaborate. create.

$whoami

Joseph Ravichandran
1st Year PhD Student, MIT

Weon Taek Na
1st Year PhD Student, MIT


SW
HW
Memory Corruption Attacks
Microarchitectural Attacks


SW
HW
Memory Corruption Attacks
Microarchitectural Attacks
PACMAN

Contributions

1

New way of thinking about compounding threat models.

2

Hardware bypass for ARM Pointer Authentication.

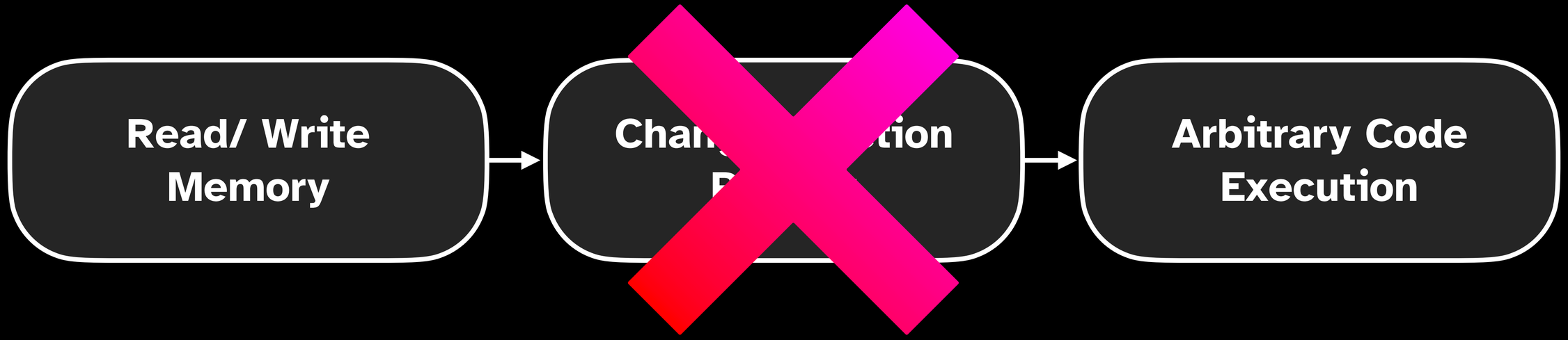
3

Attack on Apple M1.

The idea in 60 seconds.

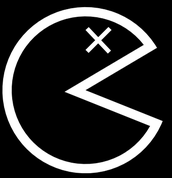
Memory Corruption

Memory Corruption



Pointer Authentication blocks changing pointers

Memory Corruption



Just bruteforce it, right?

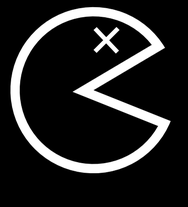
Key Insight:
Avoid crashes using speculative execution!

Agenda

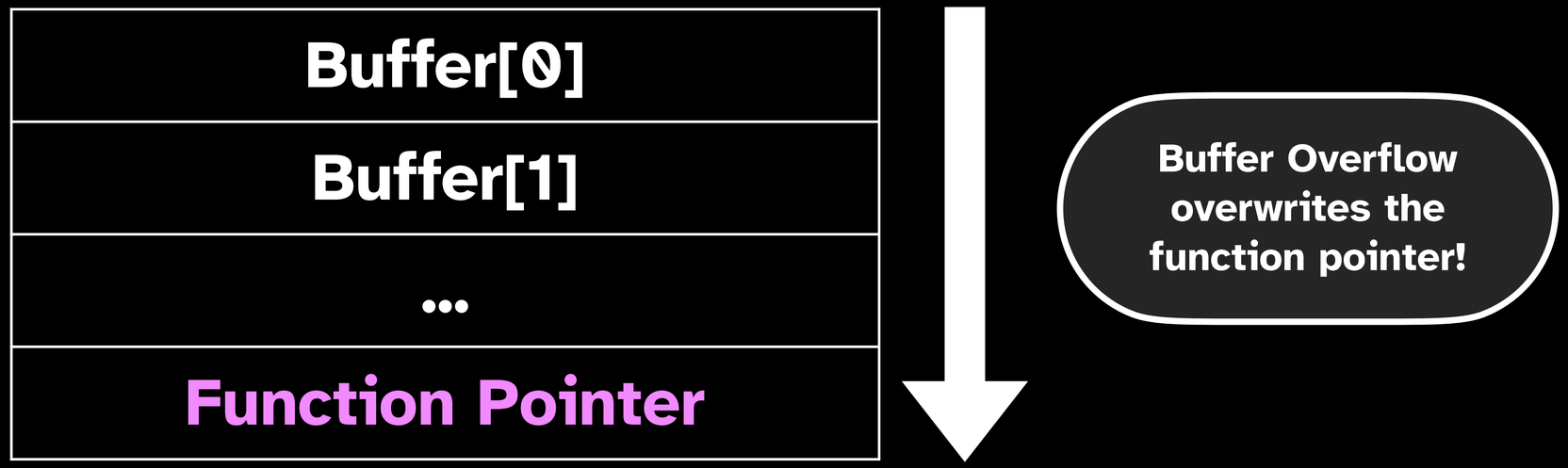
1. Background
2. High Level View
3. Data Attack
4. Instruction Attack
5. Analysis

Buffer Overflow

Buffer[0]
Buffer[1]
...
Function Pointer



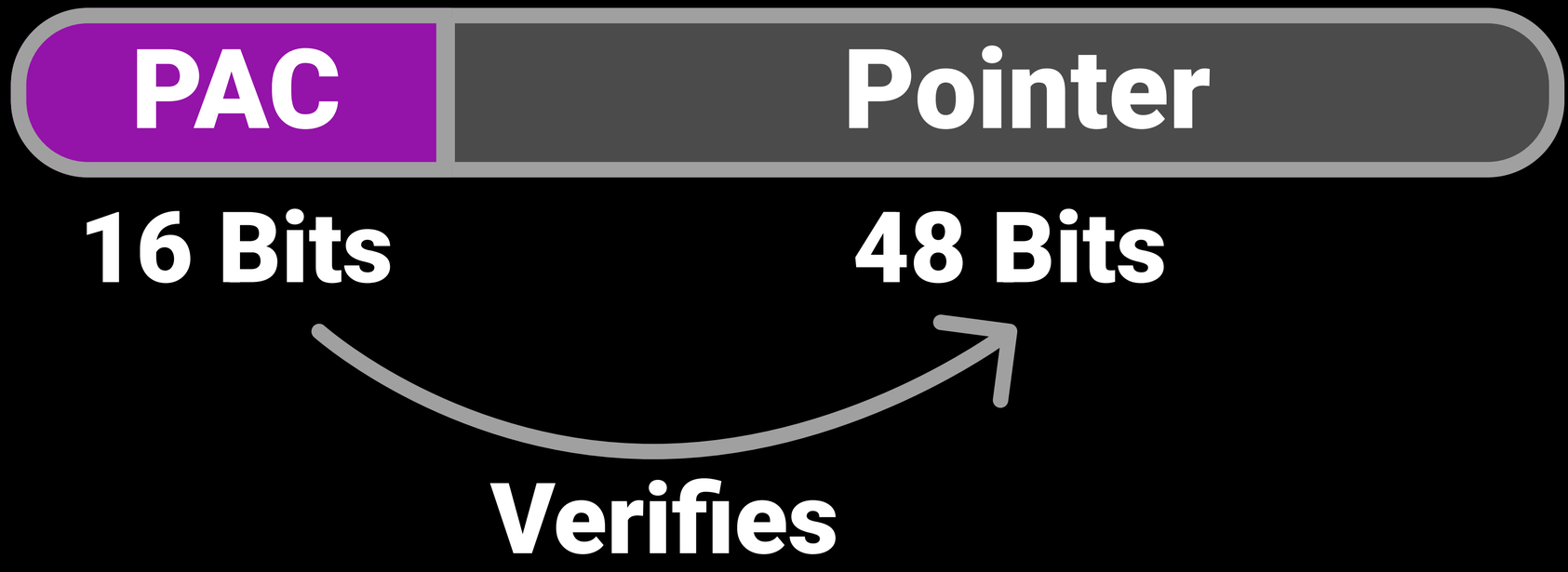
Buffer Overflow



Let's fix this bug with Pointer Authentication.

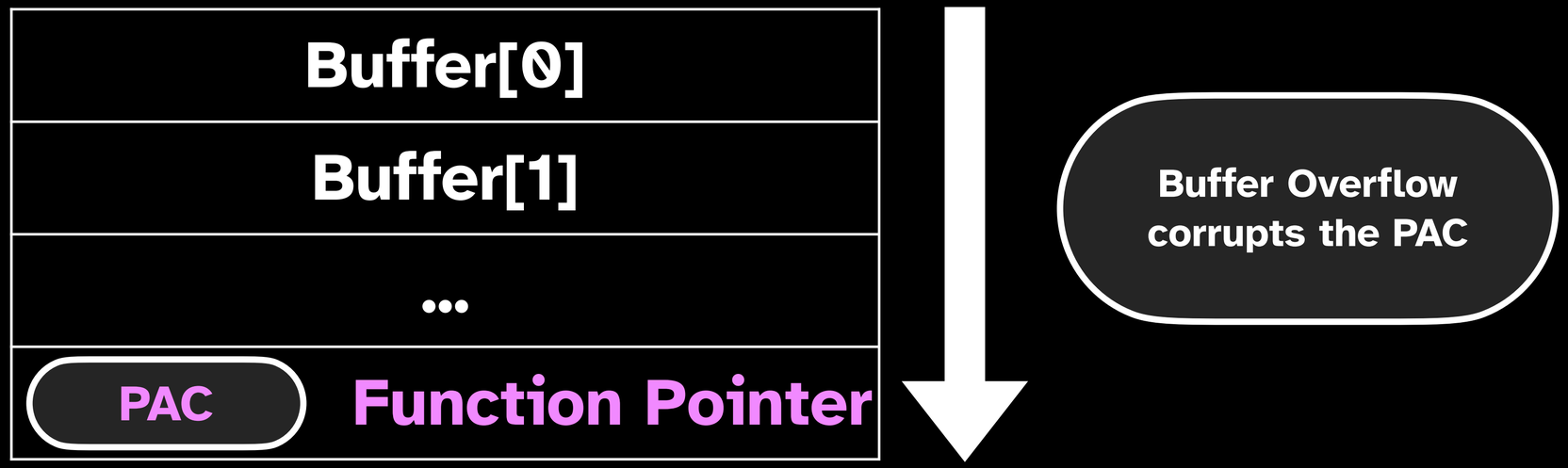
ARM Pointer Authentication

```
PAC = crypto_fn(pointer, salt, key)
```



Buffer Overflow

Buffer Overflow



Invalid PAC means we crash!

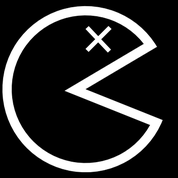
THE

GOAL

Reveal the PAC for an arbitrary pointer without crashing.

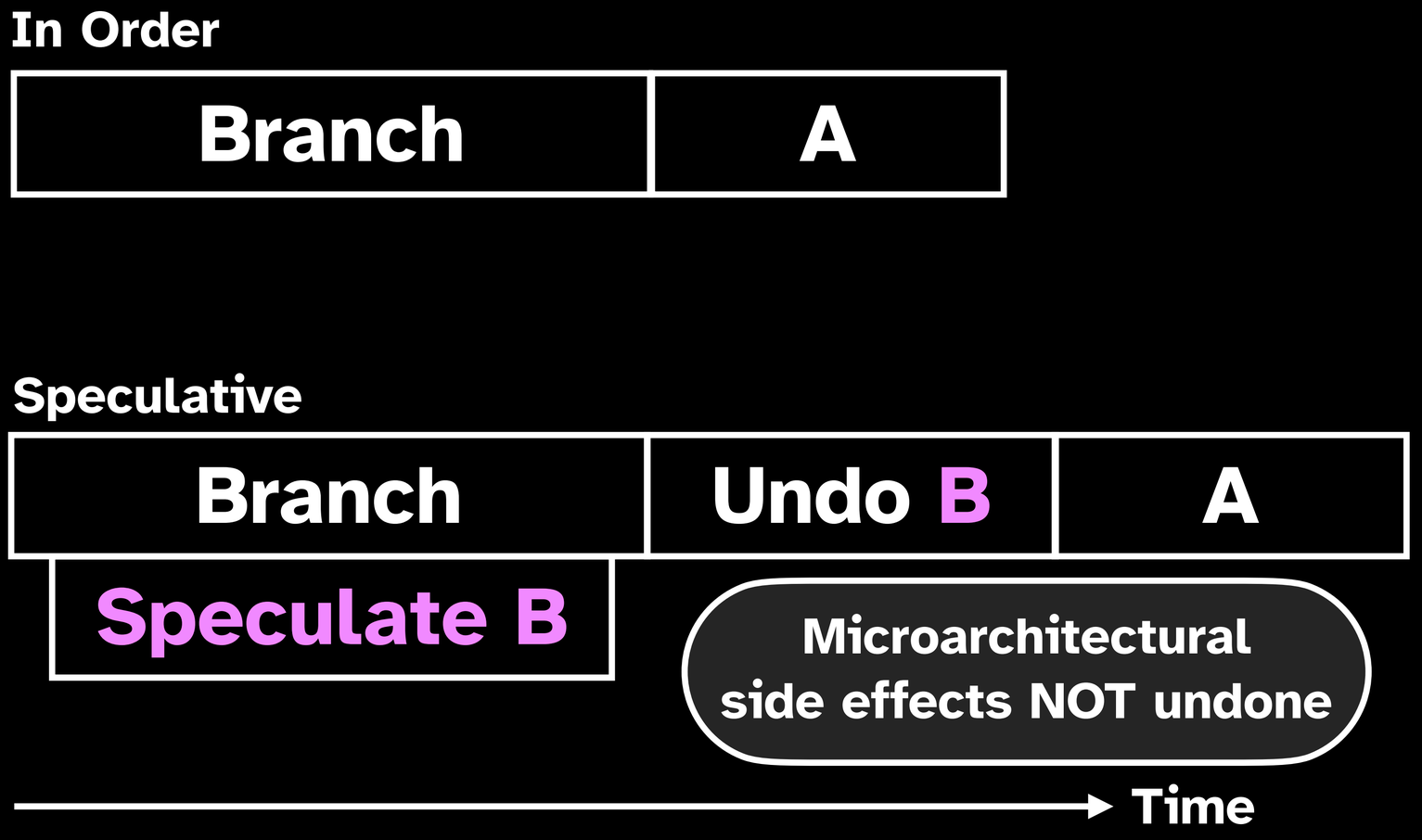
Break PAC with Hardware Attacks

- Guess a PAC **speculatively** to prevent crashes
- Leak verification results via side channel



Speculative Execution

```
if (true)
    A
else
    B
```

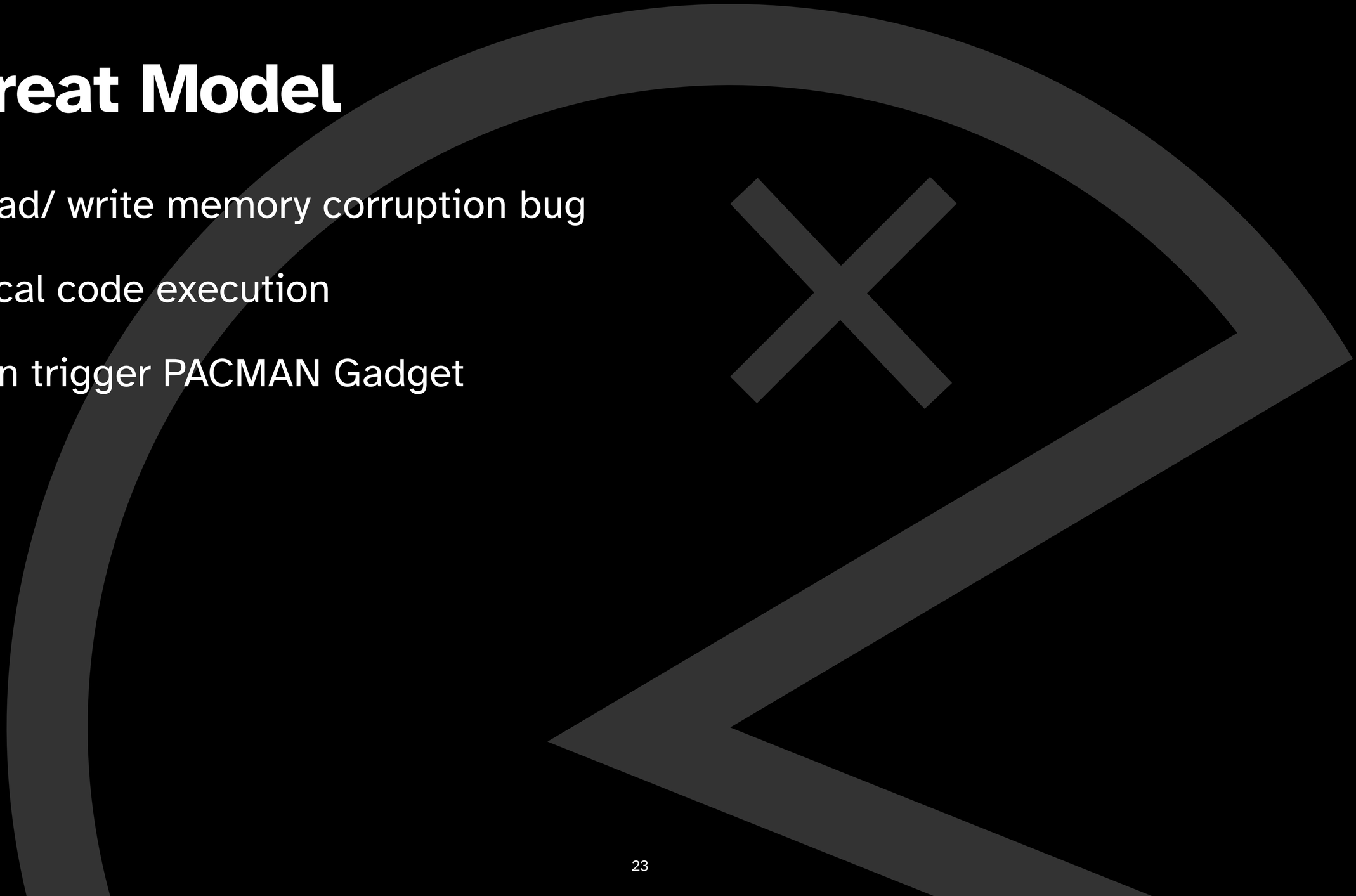


PACMAN Gadget

We use side channels to transmit the verification results of a pointer.

Threat Model

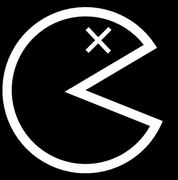
- Read/ write memory corruption bug
- Local code execution
- Can trigger PACMAN Gadget



Bird's Eye View

Data Gadget

```
if (condition):
  verified_ptr = check_pac(guess_ptr)
  load(verified_ptr)
```

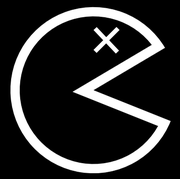


Data Attack

```
if (condition):
  verified_ptr = check_pac(guess_ptr)
  load(verified_ptr)
```

Correct PAC

Mispredict Branch →



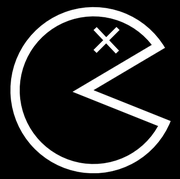
Data Attack

```
if (condition):
  verified_ptr = check_pac(guess_ptr)
  load(verified_ptr)
```

Correct PAC

Mispredict Branch →

PAC Check Succeeds

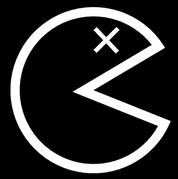


Data Attack

```
if (condition):
  verified_ptr = check_pac(guess_ptr)
  load(verified_ptr)
```

Correct PAC

Mispredict Branch → PAC Check Succeeds → Speculative **Load!** ✓



Data Attack

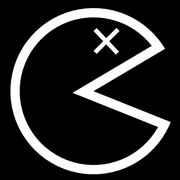
```
if (condition):
  verified_ptr = check_pac(guess_ptr)
  load(verified_ptr)
```

Correct PAC

Mispredict Branch → PAC Check Succeeds → Speculative Load!

Incorrect PAC

Mispredict Branch →



Data Attack

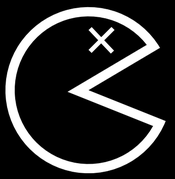
```
if (condition):
  verified_ptr = check_pac(guess_ptr)
  load(verified_ptr)
```

Correct PAC

Mispredict Branch → PAC Check Succeeds → Speculative Load!

Incorrect PAC

Mispredict Branch → PAC Check Fails



Data Attack

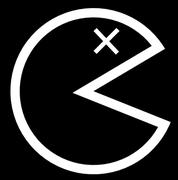
```
if (condition):
  verified_ptr = check_pac(guess_ptr)
  load(verified_ptr)
```

Correct PAC

Mispredict Branch → PAC Check Succeeds → Speculative **Load!** ✓

Incorrect PAC

Mispredict Branch → PAC Check Fails → Speculative **Exception** ✗



Instruction Gadget

```
if (condition): #BR1
  verified_ptr = check_pac(guess_ptr)
  call(verified_ptr) #BR2
```

TARGET

The world's first desktop CPU
that supports Pointer Authentication.

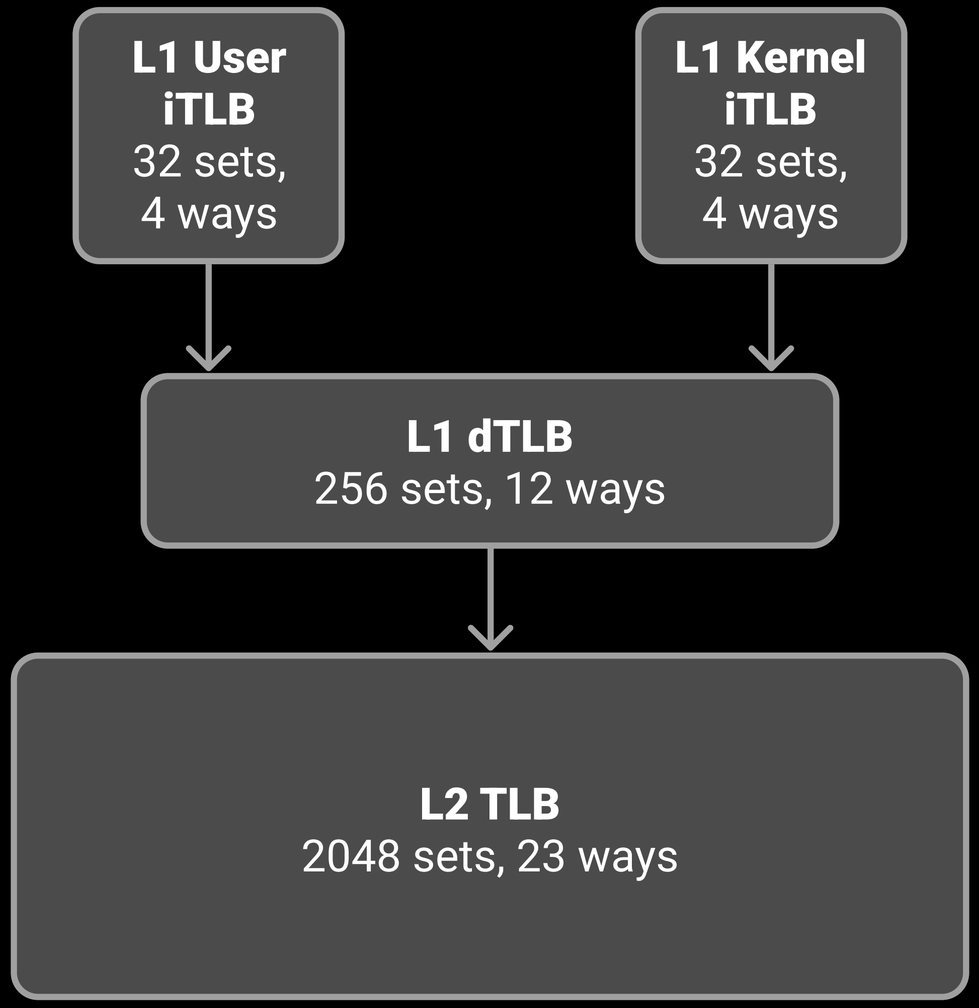
Image: Apple ("Apple Unleashes M1")

Challenges of Real World HW

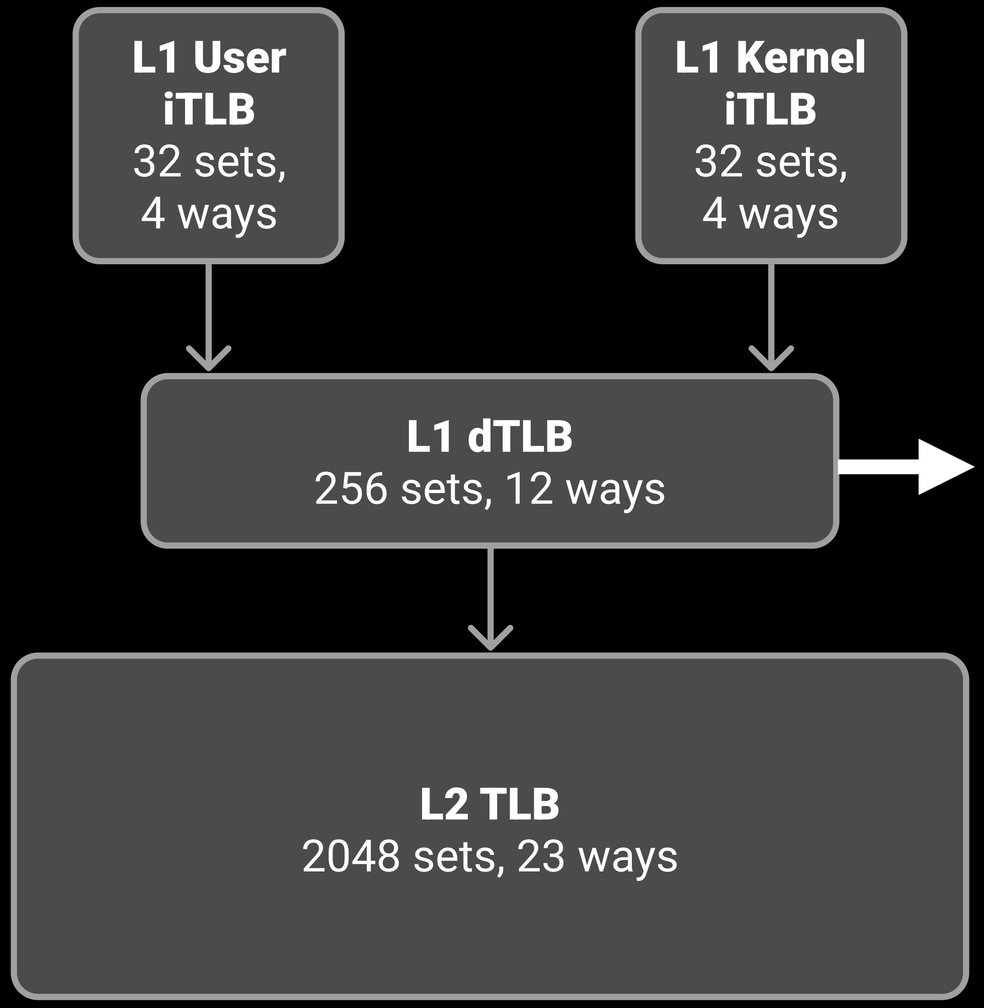
- No documentation of microarchitectural details.
- No high resolution timer.
- macOS is a difficult system to integrate attacks on.

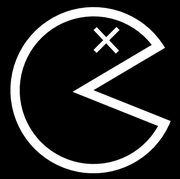
Essentially, we had to reinvent the wheel.

Conjectured TLB Hierarchy

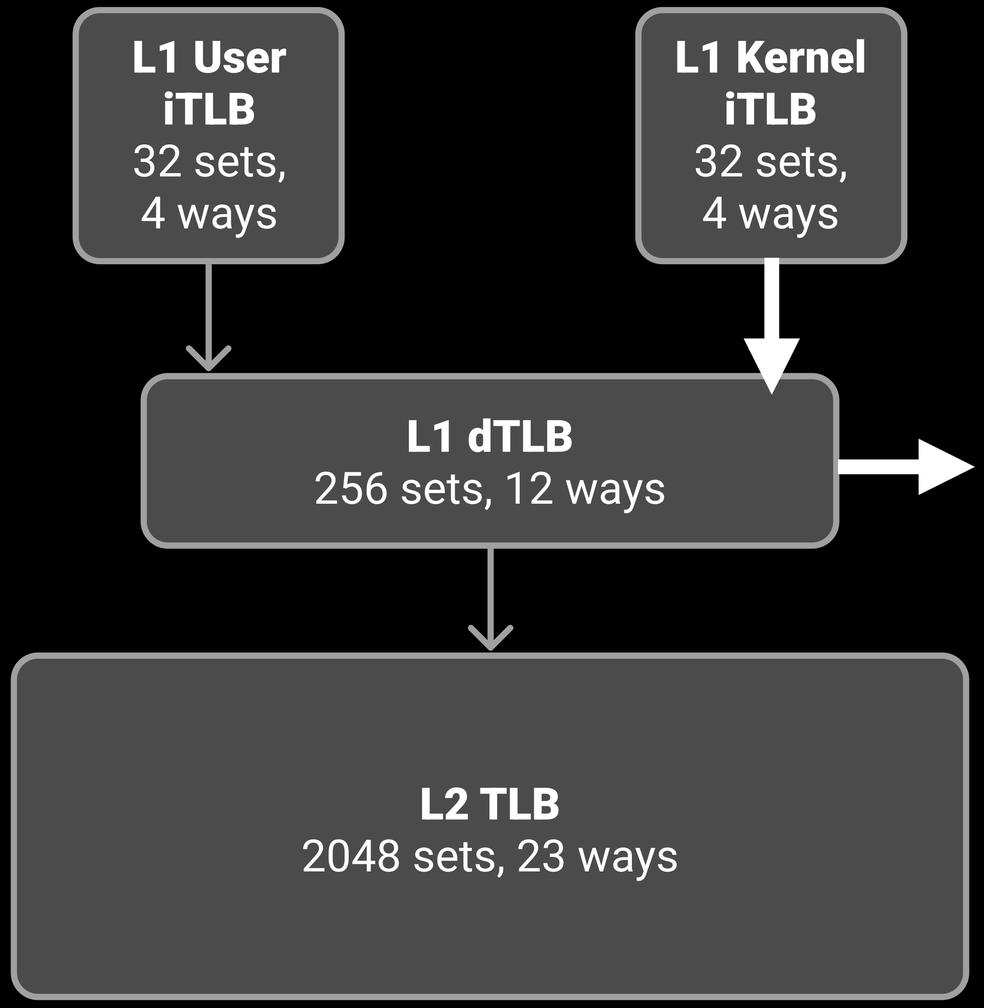


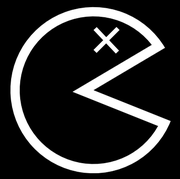
Conjectured TLB Hierarchy





Conjectured TLB Hierarchy





Experiment Testbed:

We insert a vulnerable kernel extension.

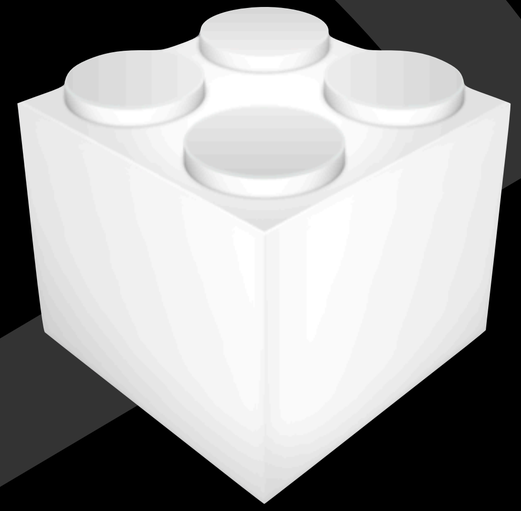
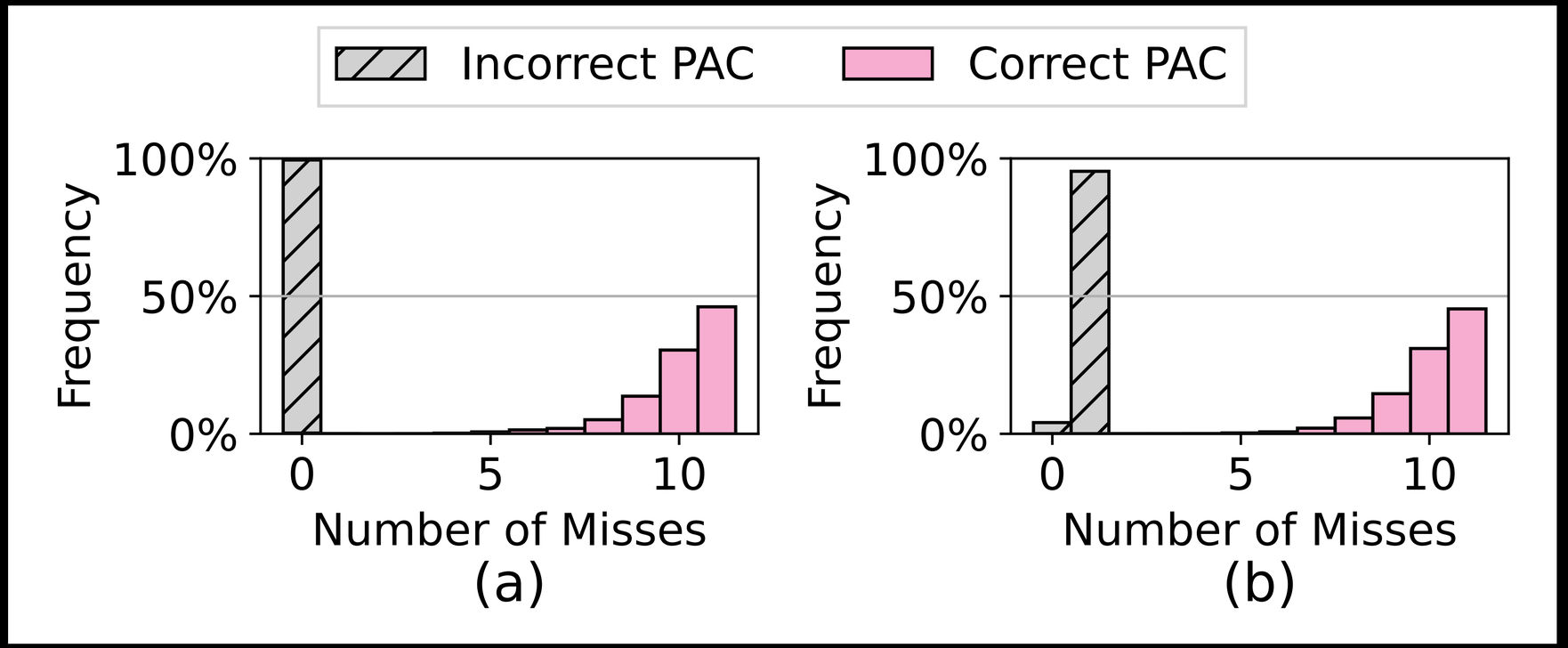


Image: Screenshot from macOS 12.3

PAC Oracle Accuracy



Data

Instructions

Under the PACMAN kext, we find each run takes 2.69ms.

This will likely be longer for real kernel code.

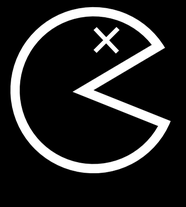
We can bruteforce an entire 16-bit PAC (from 0x0000 to 0xFFFF) in under 3 minutes.

XNU-8019.80.24

Data Gadgets	Instruction Gadgets	Total
13,867	41,292	55,159

PACMAN Gadgets are readily available in large codebases.

This list is not exhaustive, and no exploitability analysis was performed.



More in the Paper!

Reverse Engineering Experiments

Example jump2win C++ Attack

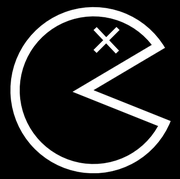
Countermeasures

TLB Details

CPU Cache Details

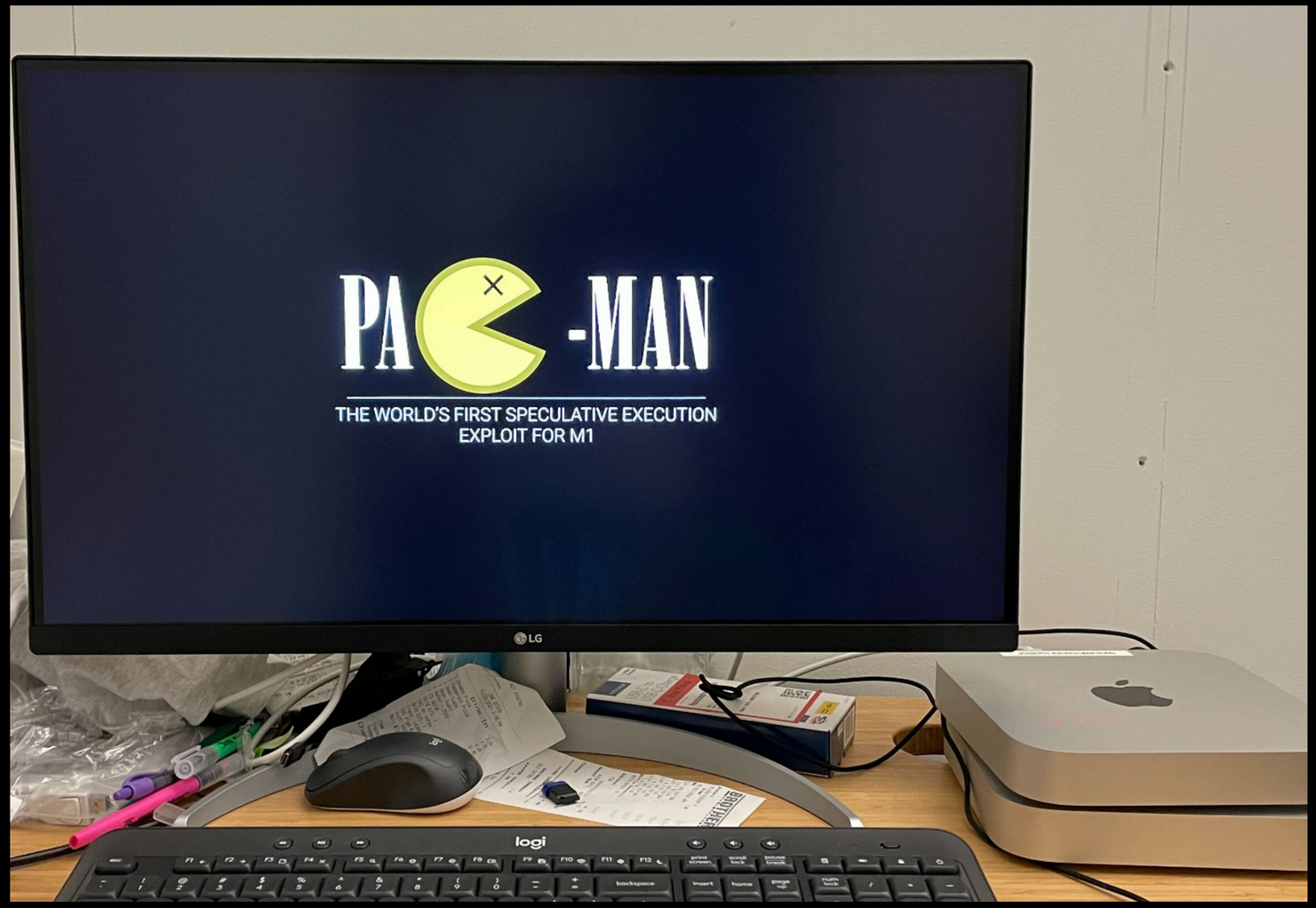
Timers on M1

And more!



PacmanOS

A Rust-based bare metal
environment for performing experiments.



Top news

digitaltrends

The M1 has a big security loophole, and Apple can't patch it

4 hours ago

TechCrunch

MIT researchers uncover ‘unpatchable’ flaw in Apple M1 chips

51 minutes ago

HARDWARE

MIT Finds New Arm Vulnerability Present in Apple M1, Demos PACMAN Attack

4 hours ago

9TO5Mac

PACMAN M1 chip attack defeats ‘the last line of security’ – but requires physical access

2 hours ago

All coverage

Phoronix

Apple M1 Affected By "PACMAN" Hardware Vulnerability In Arm Pointer Authentication

4 hours ago

VentureBeat

MIT researchers discover Apple M1 chip vulnerability

3 hours ago

The Register

Apple M1 chip contains hardware vulnerability that bypasses memory defense

4 hours ago

DARKReading

Design Weakness Discovered in Apple M1 Kernel Protections

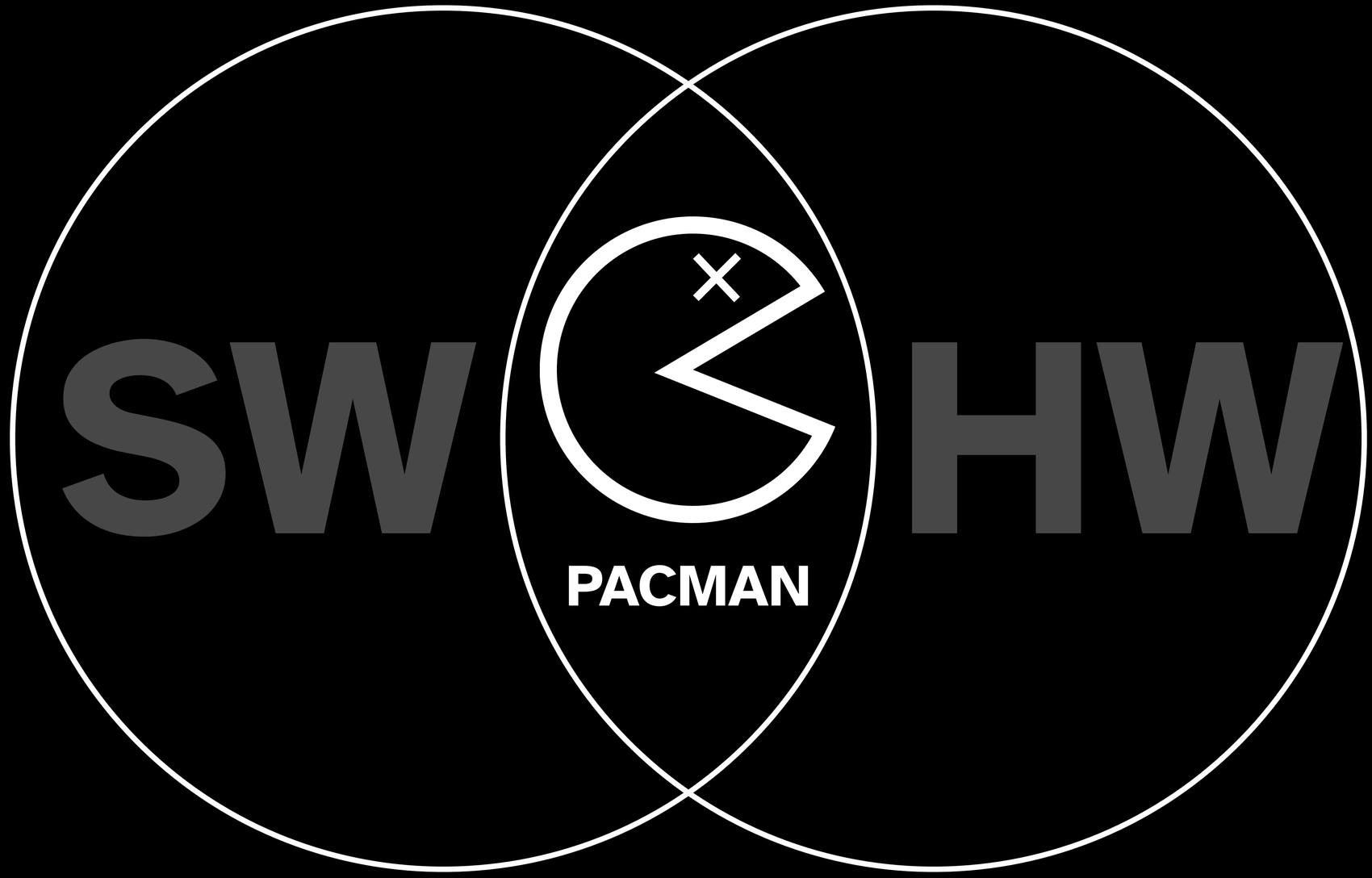
3 hours ago

Macworld

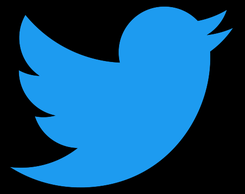
Experts warn of 'PACMAN' flaw in M1 chip that can't be patched

1 hour ago

PACMAN: Attacking ARM Pointer Authentication with Speculative Execution

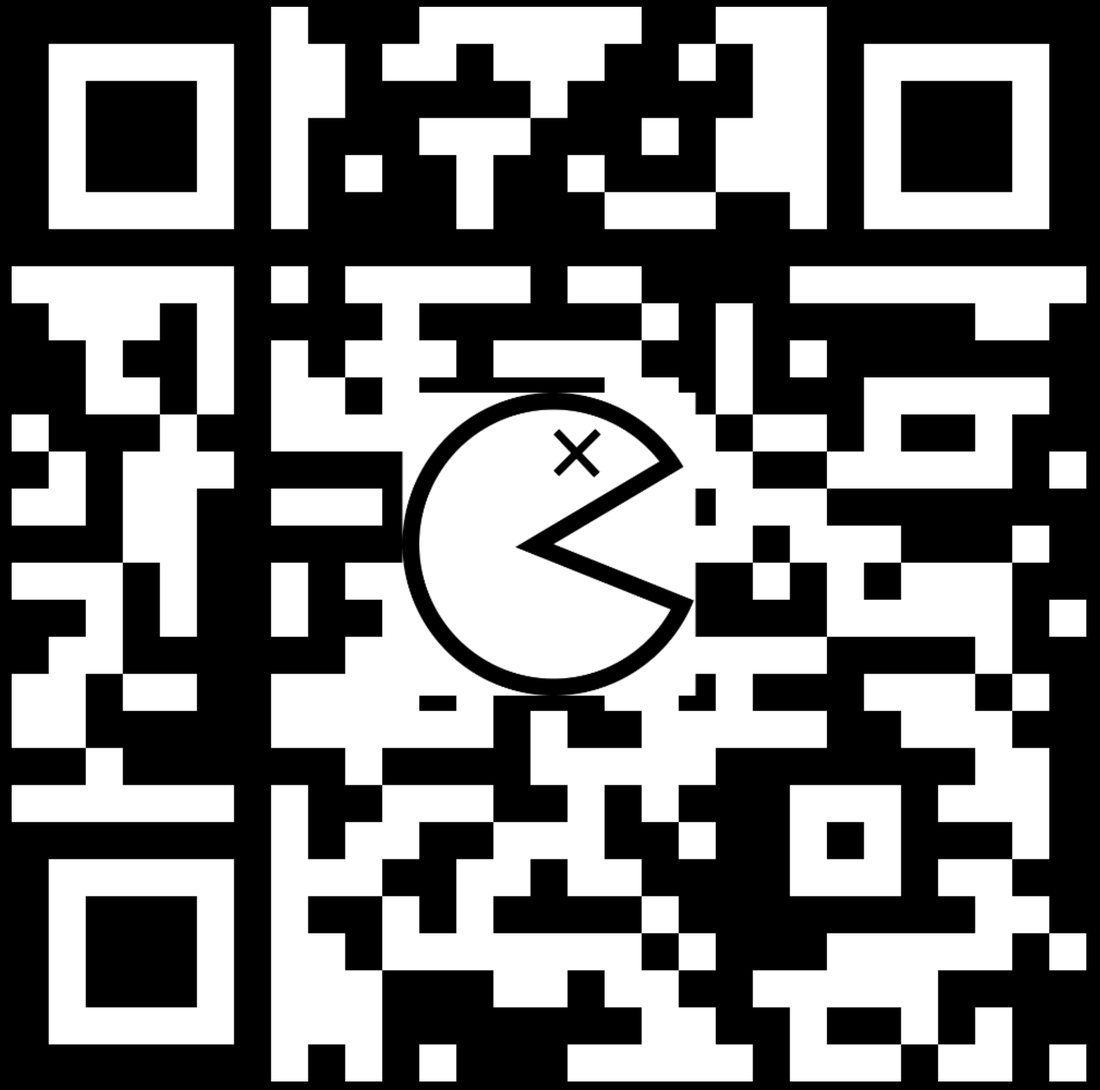


Follow us on Twitter!



@0xjprx

@weon_taek_na



PACMANATTACK.COM

MIT CSAIL

compute. collaborate. create.