Copy-on-Flip: Hardening ECC Memory Against Rowhammer Attacks

Andrea Di Dio[1], Koen Koning[2], Herbert Bos[1], Cristiano Giuffrida[1]

[1]**Vrije Universiteit Amsterdam**
[2]**Intel**

TL;DR

- Rowhammer is still an unresolved problem

TL;DR

- Rowhammer is still an unresolved problem
- We need to protect e*xisting* systems

TL;DR

- Rowhammer is still an unresolved problem
- We need to protect *existing* systems
- Meet Copy-on-Flip:

TL;DR

- Rowhammer is still an unresolved problem
- We need to protect *existing* systems
- Meet Copy-on-Flip:
  - ECC to detect ongoing Rowhammer attacks

TL;DR

- Rowhammer is still an unresolved problem
- We need to protect *existing* systems
- Meet Copy-on-Flip:
  - ECC to detect ongoing Rowhammer attacks
  - Transparent page migration and offlining for vulnerable pages

TL;DR

- Rowhammer is still an unresolved problem
- We need to protect *existing* systems
- Meet Copy-on-Flip:
  - ECC to detect ongoing Rowhammer attacks
  - Transparent page migration and offlining for vulnerable pages
  - Low overhead with >95% attack surface reduction (including kernel memory)

Background – Rowhammer Attacks

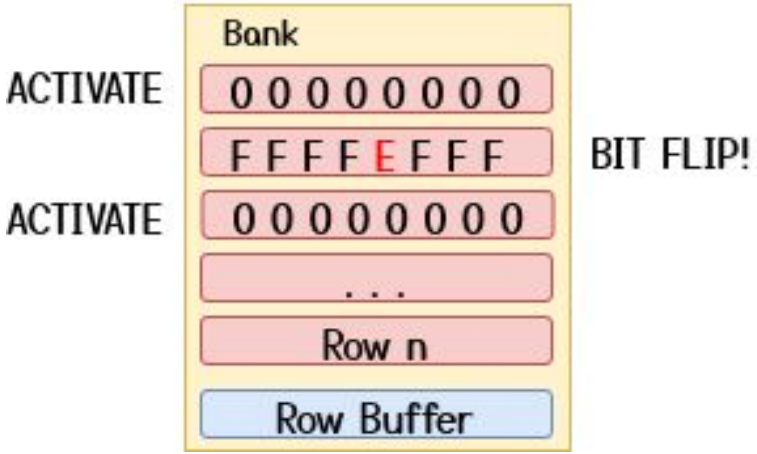
Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Yoongu Kim[1] Ross Daly* Jeremie Kim[1] Chris Fallin* Ji Hye Lee[1]
Donghyuk Lee[1] Chris Wilkerson[2] Konrad Lai Onur Mutlu[1]

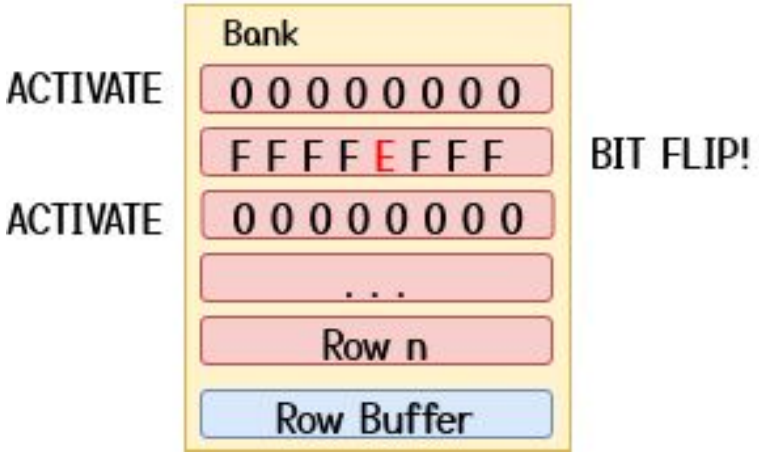
Background – Rowhammer Attacks

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Yoongu Kim[1] Ross Daly* Jeremie Kim[1] Chris Fallin* Ji Hye Lee[1]
Donghyuk Lee[1] Chris Wilkerson[2] Konrad Lai Onur Mutlu[1]



Background – Rowhammer Attacks



Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

ie Kim[1] Chris Fallin* Ji Hye Lee[1]
on[2] Konrad Lai Onur Mutlu[1]

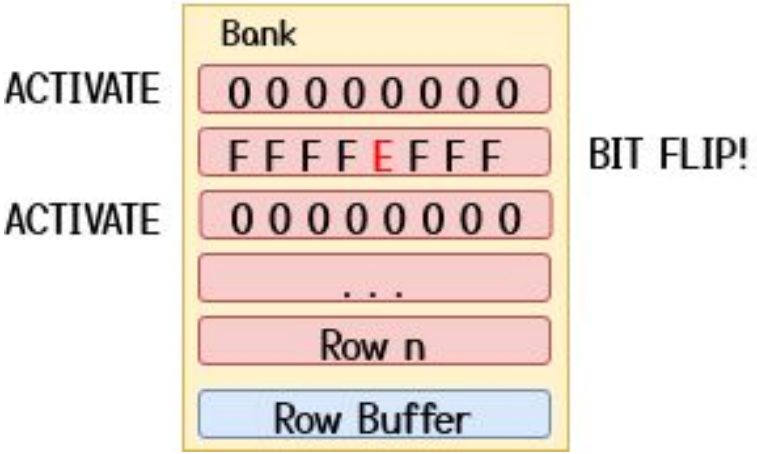
WIRED SUBSCRIBE

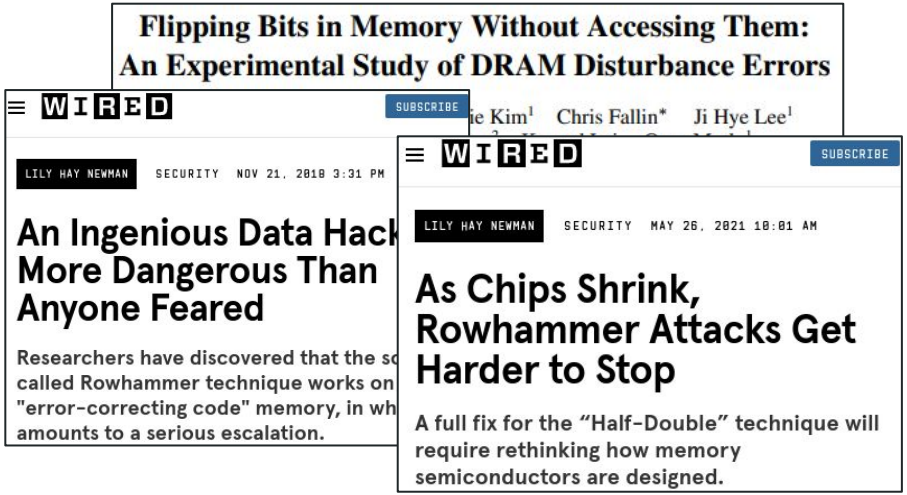
LILY HAY NEWMAN SECURITY NOV 21, 2018 3:31 PM

An Ingenious Data Hack Is More Dangerous Than Anyone Feared

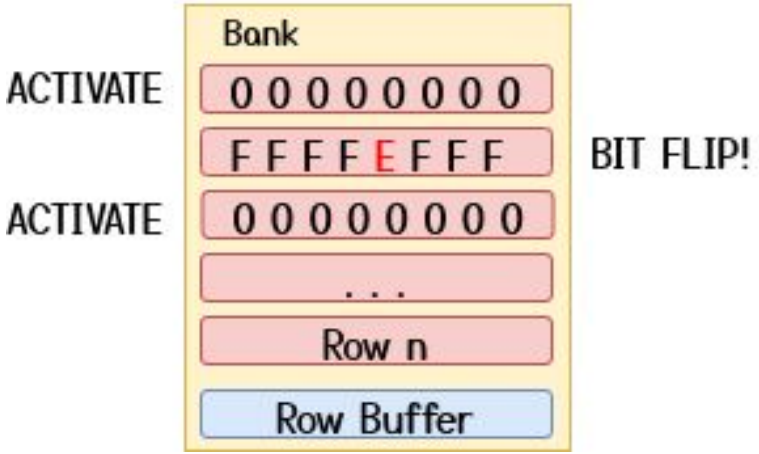
Researchers have discovered that the so-called Rowhammer technique works on "error-correcting code" memory, in what amounts to a serious escalation.

Background – Rowhammer Attacks

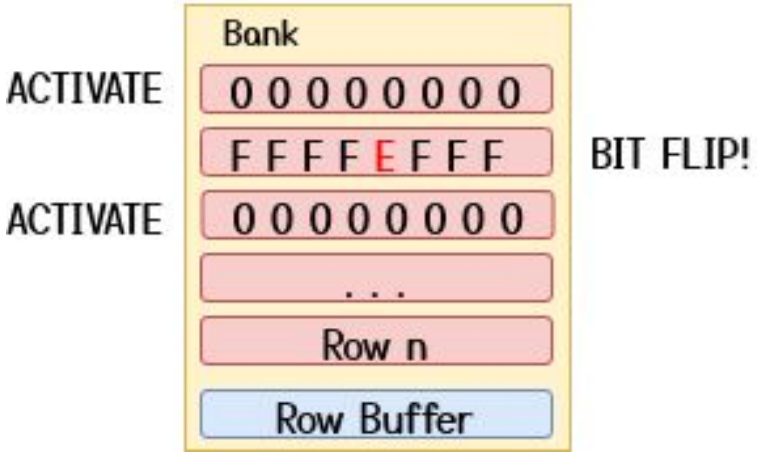


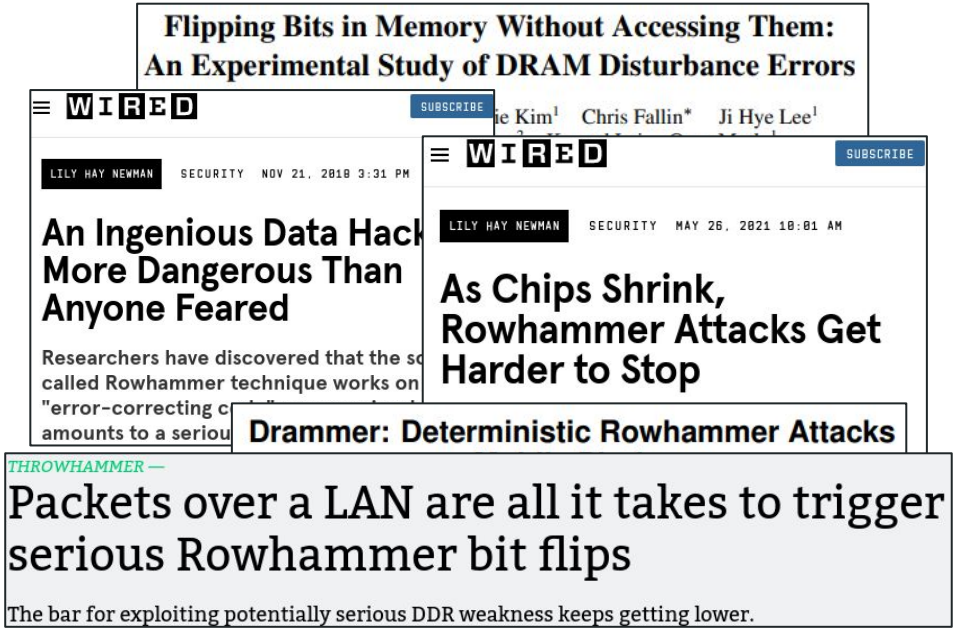


Background – Rowhammer Attacks

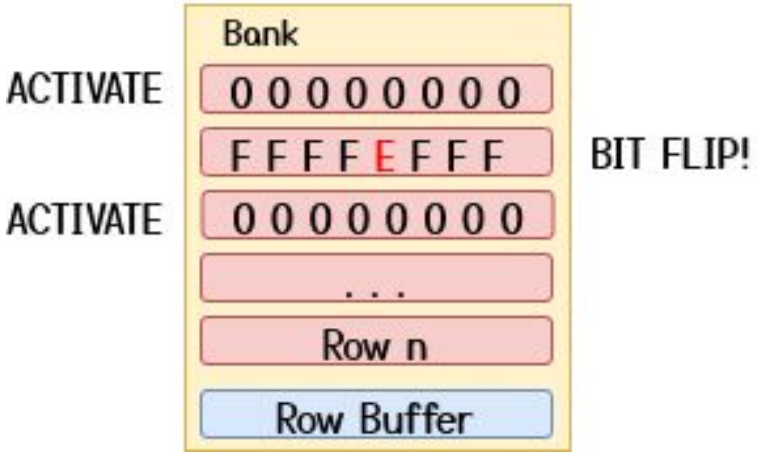


Background – Rowhammer Attacks





Background – Rowhammer Attacks



Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

WIRED SUBSCRIBE

LILY HAY NEWMAN SECURITY NOV 21, 2018 3:31 PM

An Ingenious Data Hack More Dangerous Than Anyone Feared

Researchers have discovered that the so called Rowhammer technique works on "error-correcting c amounts to a seriou

WIRED SUBSCRIBE

LILY HAY NEWMAN SECURITY MAY 26, 2021 10:01 AM

As Chips Shrink, Rowhammer Attacks Get Harder to Stop

Drammer: Deterministic Rowhammer Attacks

THROWHAMMER —

Packets over a LAN are all it takes to trigger serious Rowhammer bit flips

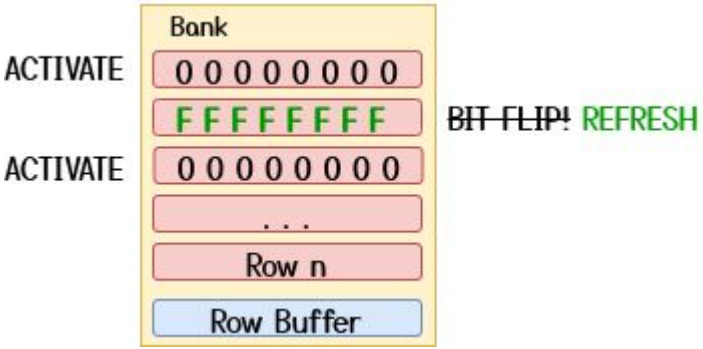
The b

BLACKSMITH —

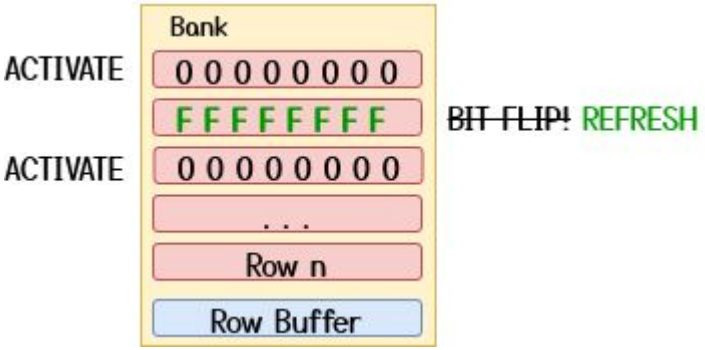
DDR4 memory protections are broken wide open by new Rowhammer technique

Researchers build "fuzzer" that supercharges potentially serious bitflipping exploits.

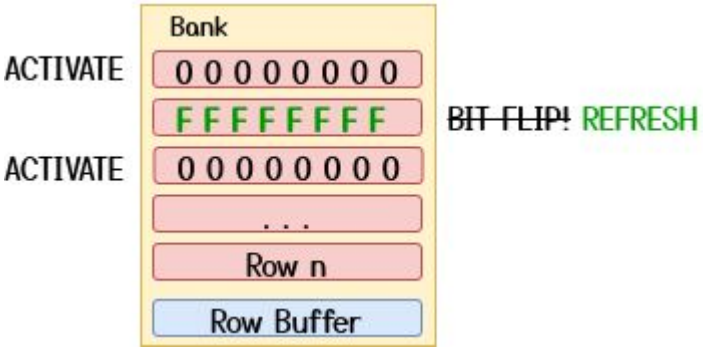
Background – Rowhammer Defenses

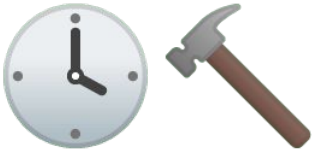


Background – Rowhammer Defenses



Background – Rowhammer Defenses





Background – Rowhammer & ECC

- Single Error Correction, Double Error Detection (SECDED)

Background – Rowhammer & ECC

- Single Error Correction, Double Error Detection (SECDED)
- ECC was thought to be a strong defense against Rowhammer

Background – Rowhammer & ECC

- Single Error Correction, Double Error Detection (SECDED)
- ECC was thought to be a strong defense against Rowhammer
- One-bit-at-a-time templating

Background – Rowhammer & ECC

- Single Error Correction, Double Error Detection (SECDED)
- ECC was thought to be a strong defense against Rowhammer
- One-bit-at-a-time templating
- ≥ 3 bit flips to evade ECC

Copy-on-Flip - Why?

- SW Defenses are impractical
  - High overhead
  - Unrealistic assumptions

Copy-on-Flip - Why?

- SW Defenses are impractical
  - High overhead
  - Unrealistic assumptions
- HW Defenses *may* end Rowhammer in the future

Copy-on-Flip - Why?

- SW Defenses are impractical
  - High overhead
  - Unrealistic assumptions
- HW Defenses *may* end Rowhammer in the future
  - Slow to deploy 🚧

Copy-on-Flip - Why?

- SW Defenses are impractical
  - High overhead
  - Unrealistic assumptions
- HW Defenses *may* end Rowhammer in the future
  - Slow to deploy 🚧
- We need to protect *existing* systems **now** 🚨

Copy-on-Flip - What?

- Target: Systems equipped with ECC memory

Copy-on-Flip - What?

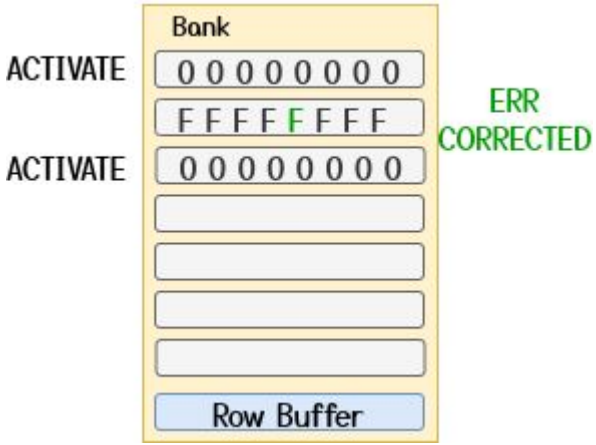
- Target: Systems equipped with ECC memory
- Stop ECC-aware Rowhammer at the *templating* phase of the attack

Copy-on-Flip - What?

- Target: Systems equipped with ECC memory
- Stop ECC-aware Rowhammer at the *templating* phase of the attack
- Simple design to protect vulnerable memory at runtime

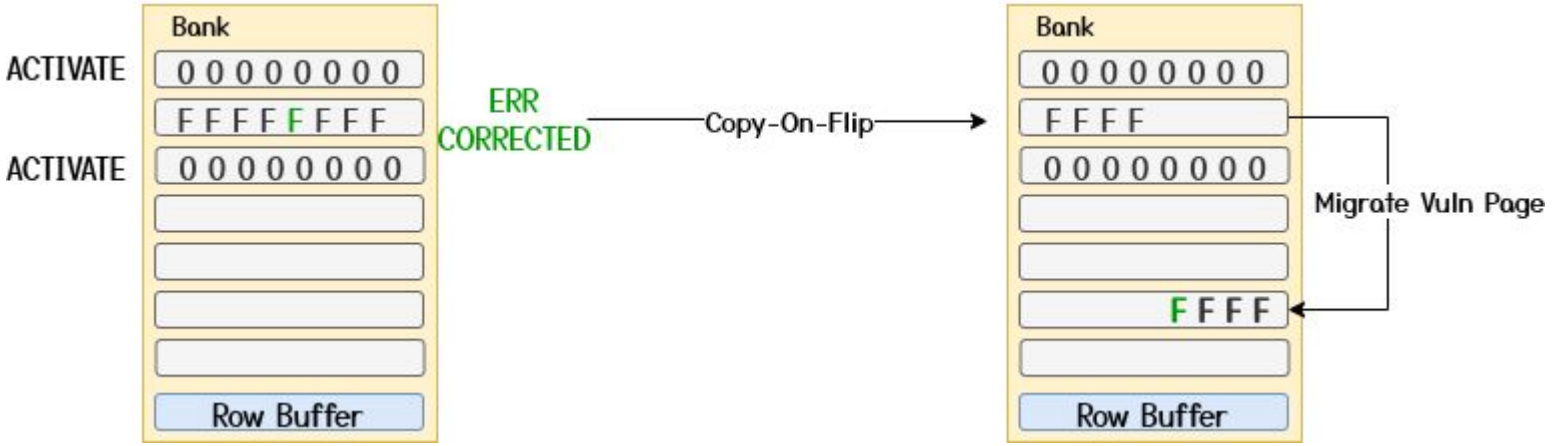
Copy-on-Flip - What?

- Target: Systems equipped with ECC memory
- Stop ECC-aware Rowhammer at the *templating* phase of the attack
- Simple design to protect vulnerable memory at runtime



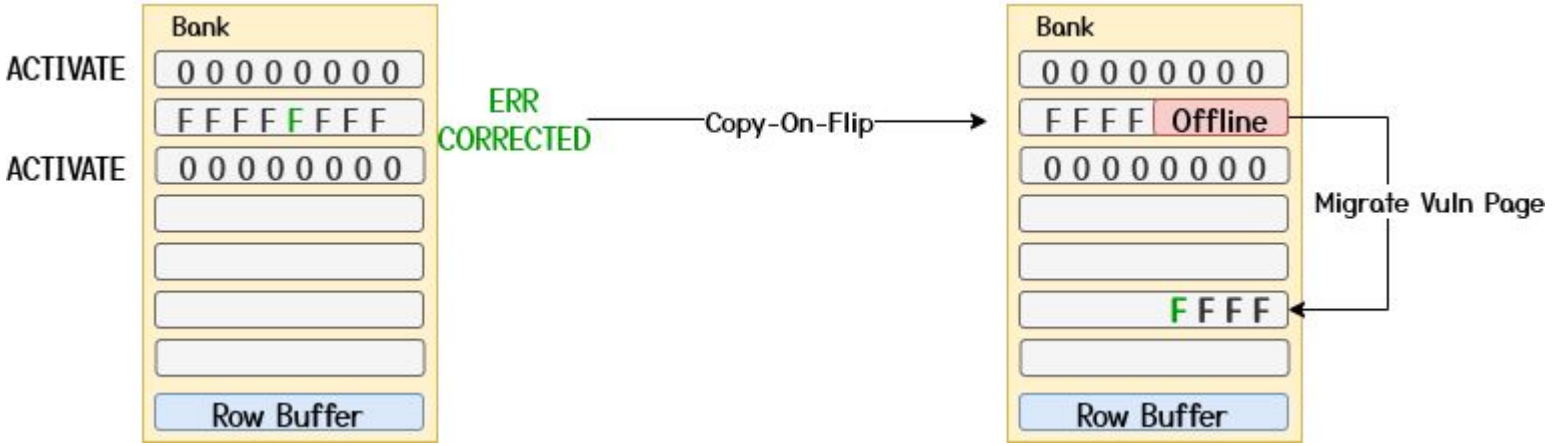
Copy-on-Flip - What?

- Target: Systems equipped with ECC memory
- Stop ECC-aware Rowhammer at the *templating* phase of the attack
- Simple design to protect vulnerable memory at runtime

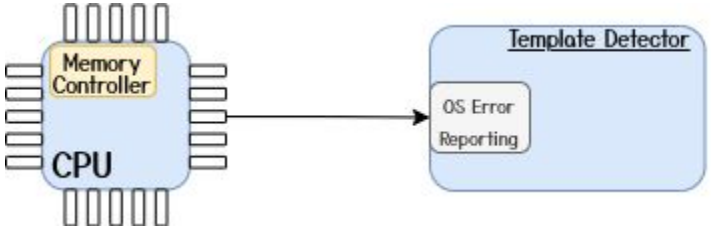


Copy-on-Flip - What?

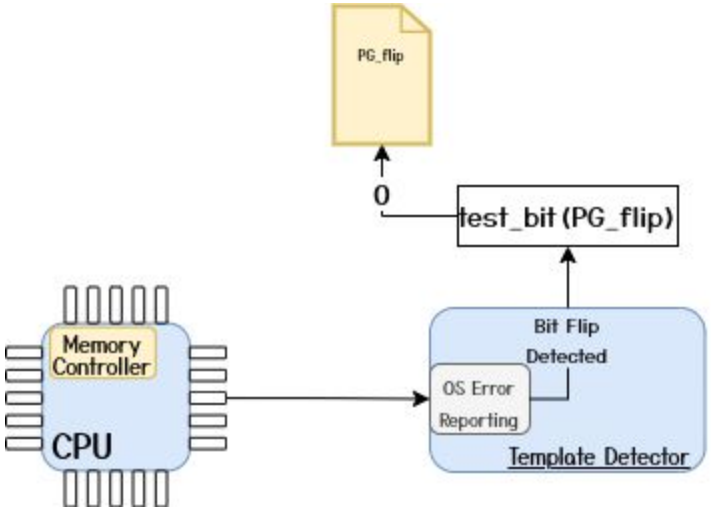
- Target: Systems equipped with ECC memory
- Stop ECC-aware Rowhammer at the *templating* phase of the attack
- Simple design to protect vulnerable memory at runtime



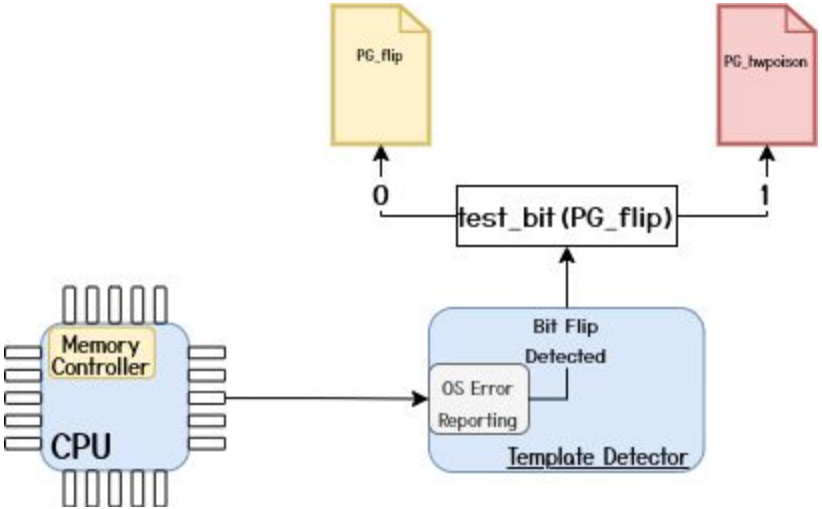
Copy-on-Flip – Template Detector



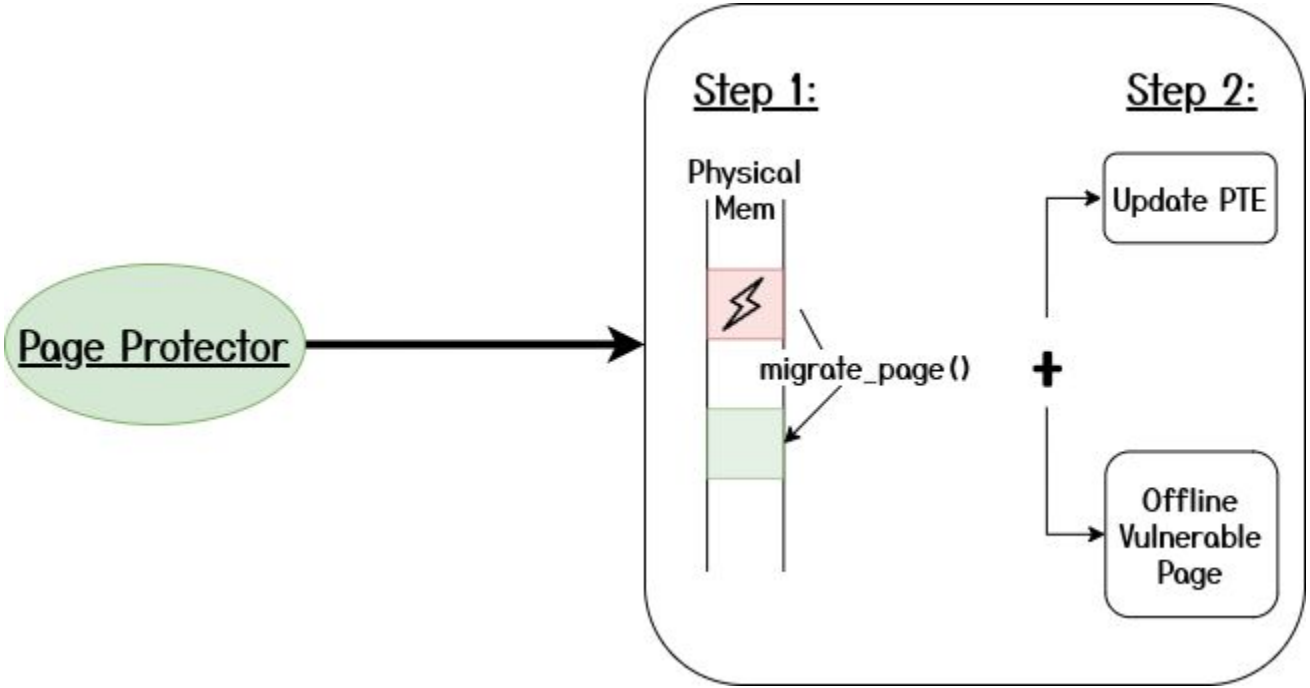
Copy-on-Flip – Template Detector



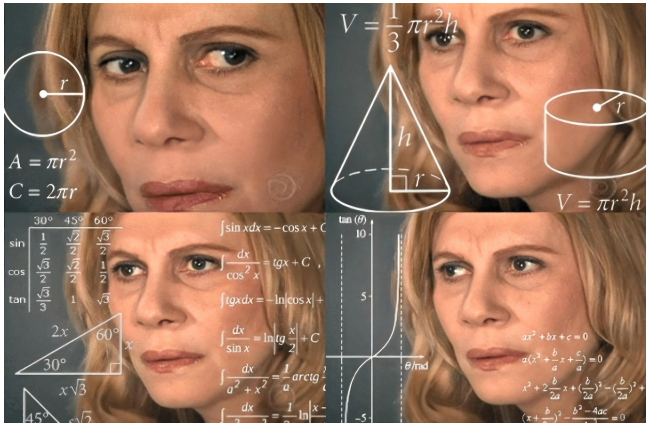
Copy-on-Flip – Template Detector



Copy-on-Flip – Page Protector

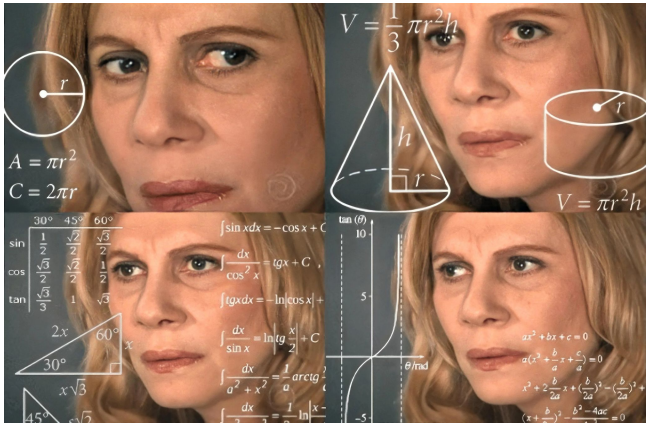


What is Vulnerable Memory?



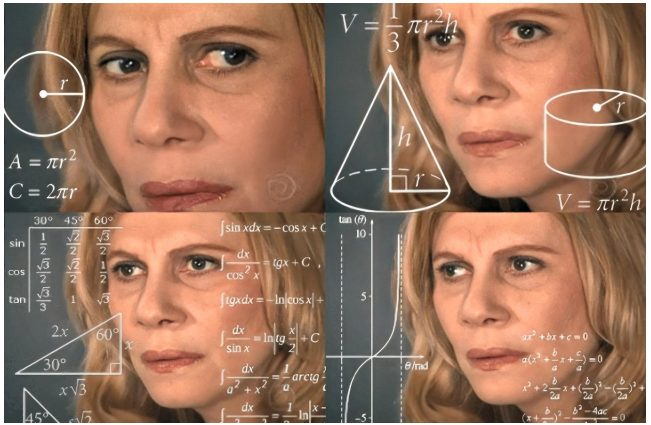
What is Vulnerable Memory?

- Pages attacker can allocate/use for templating



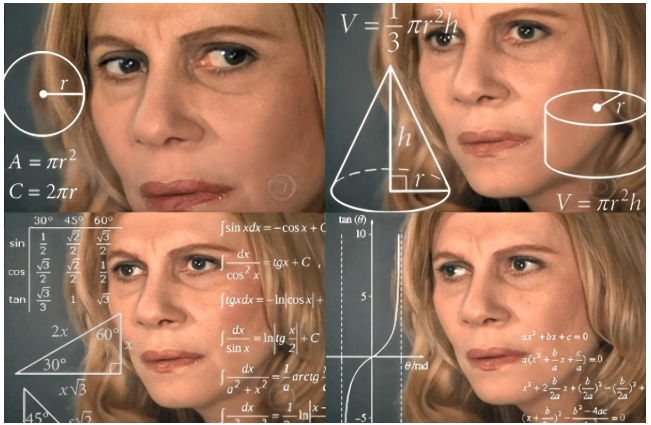
What is Vulnerable Memory?

- Pages attacker can allocate/use for templating
  - Userspace pages



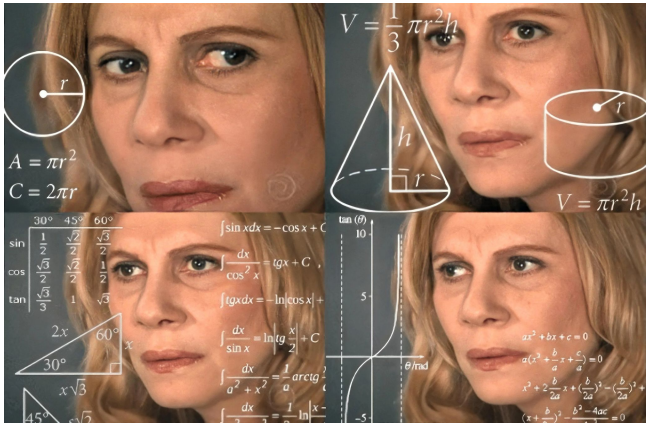
What is Vulnerable Memory?

- Pages attacker can allocate/use for templating
  - Userspace pages
- Pages allocated by the kernel on behalf of the attacker



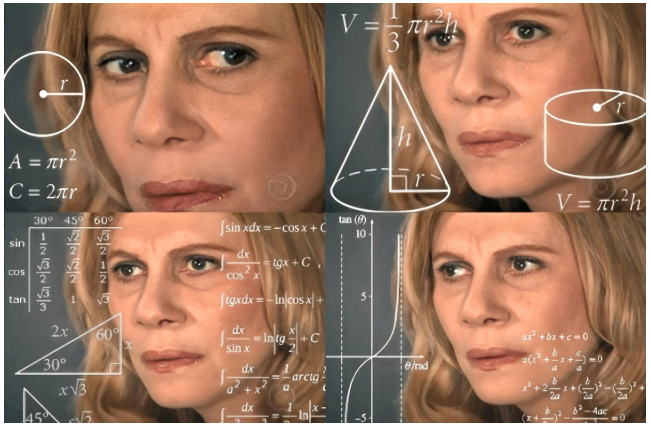
What is Vulnerable Memory?

- Pages attacker can allocate/use for templating
  - Userspace pages
- Pages allocated by the kernel on behalf of the attacker
  - Page Tables



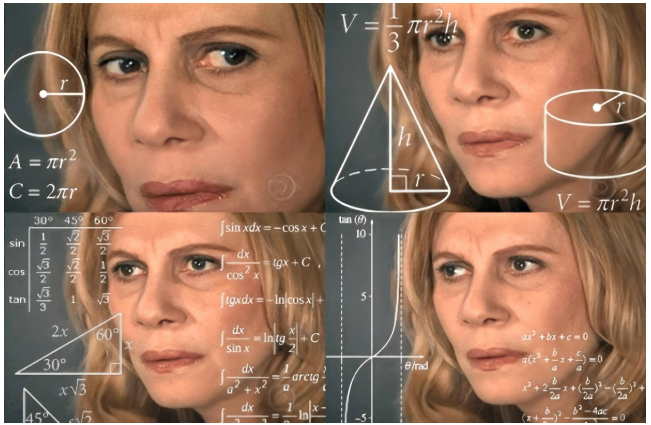
What is Vulnerable Memory?

- Pages attacker can allocate/use for templating
  - Userspace pages
- Pages allocated by the kernel on behalf of the attacker
  - Page Tables
  - Page Cache



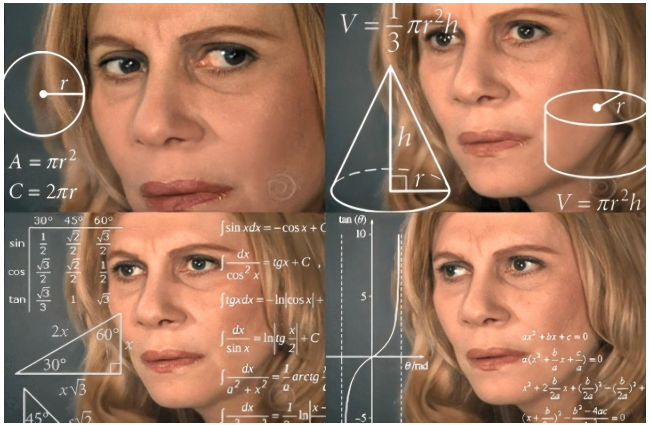
What is Vulnerable Memory?

- Pages attacker can allocate/use for templating
  - Userspace pages
- Pages allocated by the kernel on behalf of the attacker
  - Page Tables
  - Page Cache
  - Slab



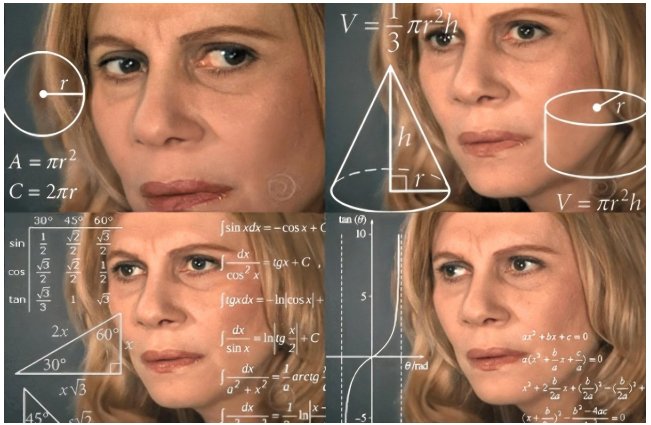
What is Vulnerable Memory?

- Pages attacker can allocate/use for templating
  - Userspace pages
- Pages allocated by the kernel on behalf of the attacker
  - Page Tables
  - Page Cache
  - Slab
  - vmalloc-like



What is Vulnerable Memory?

- Pages attacker can allocate/use for templating
  - Userspace pages
- Pages allocated by the kernel on behalf of the attacker
  - Page Tables
  - Page Cache
  - Slab
  - vmalloc-like
  - Kernel stacks



Challenge – Kernel Pages

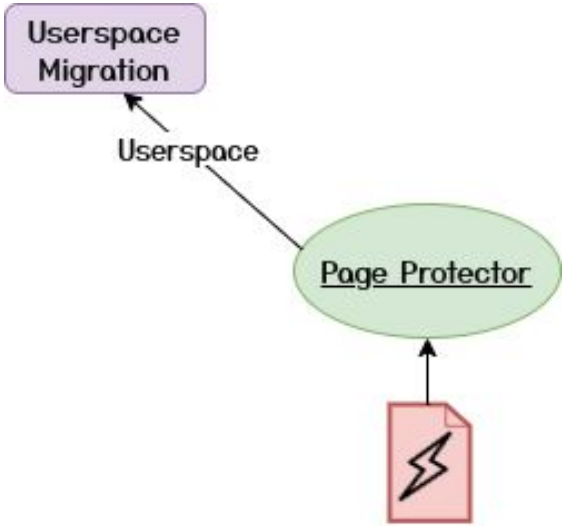
- Existing page offlining implementations in Linux ignore kernel pages

```
886  /*
887   * Error hit kernel page.
888   * Do nothing, try to be lucky and not touch this instead. For a few cases we
889   * could be more sophisticated.
890   */
891  static int me_kernel(struct page_state *ps, struct page *p)
892  {
893          unlock_page(p);
894          return MF_IGNORED;
895  }
```

mm/memory-failure.c

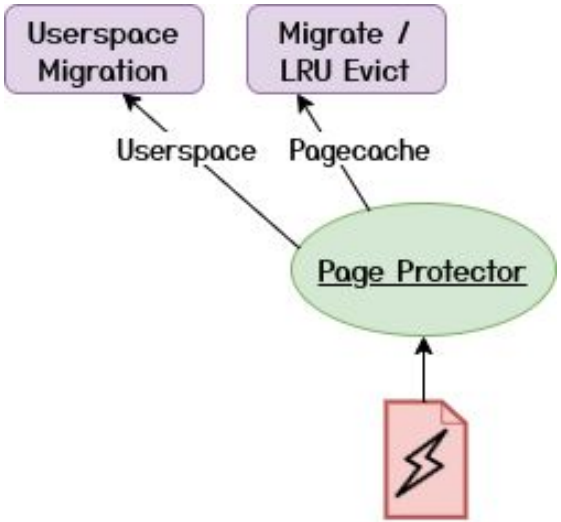
Copy-on-Flip - Page Protector

- Transparent page migration and offlining



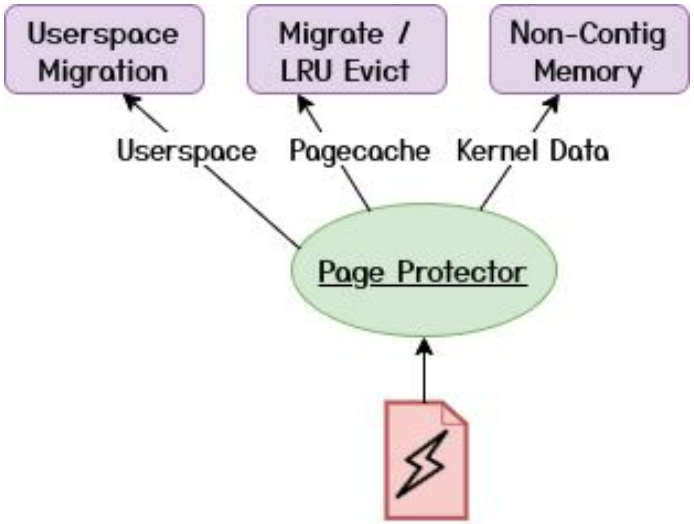
Copy-on-Flip - Page Protector

- Transparent page migration and offlining



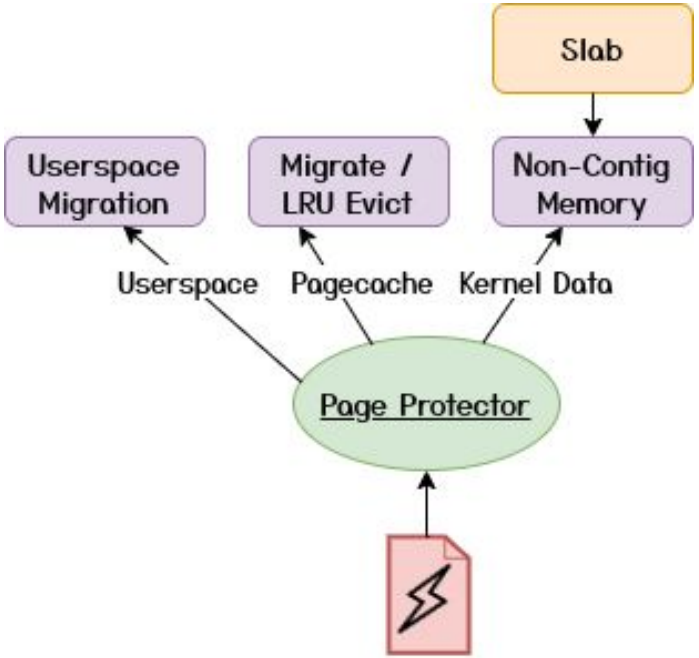
Copy-on-Flip - Page Protector

- Transparent page migration and offlining



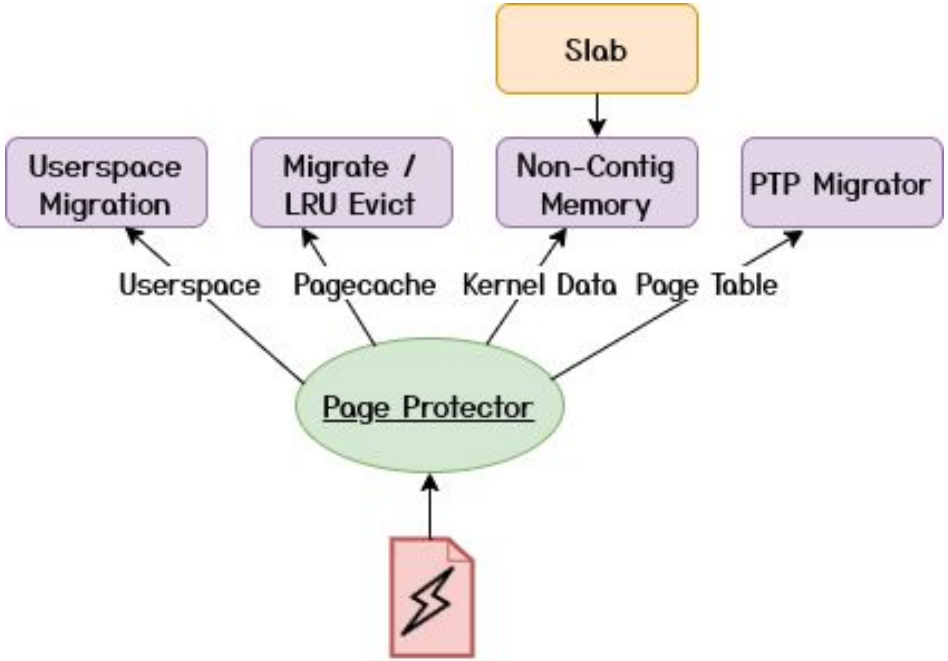
Copy-on-Flip - Page Protector

- Transparent page migration and offlining



Copy-on-Flip - Page Protector

- Transparent page migration and offlining



Evaluation – Security

- HW Error Injection to test defense

Evaluation – Security

- HW Error Injection to test defense
- >95% of memory is movable

Evaluation – Security

- HW Error Injection to test defense
- >95% of memory is movable
- System-wide protection

Evaluation – Security

- HW Error Injection to test defense
- >95% of memory is movable
- System-wide protection
- No assumptions on Rowhammer variant

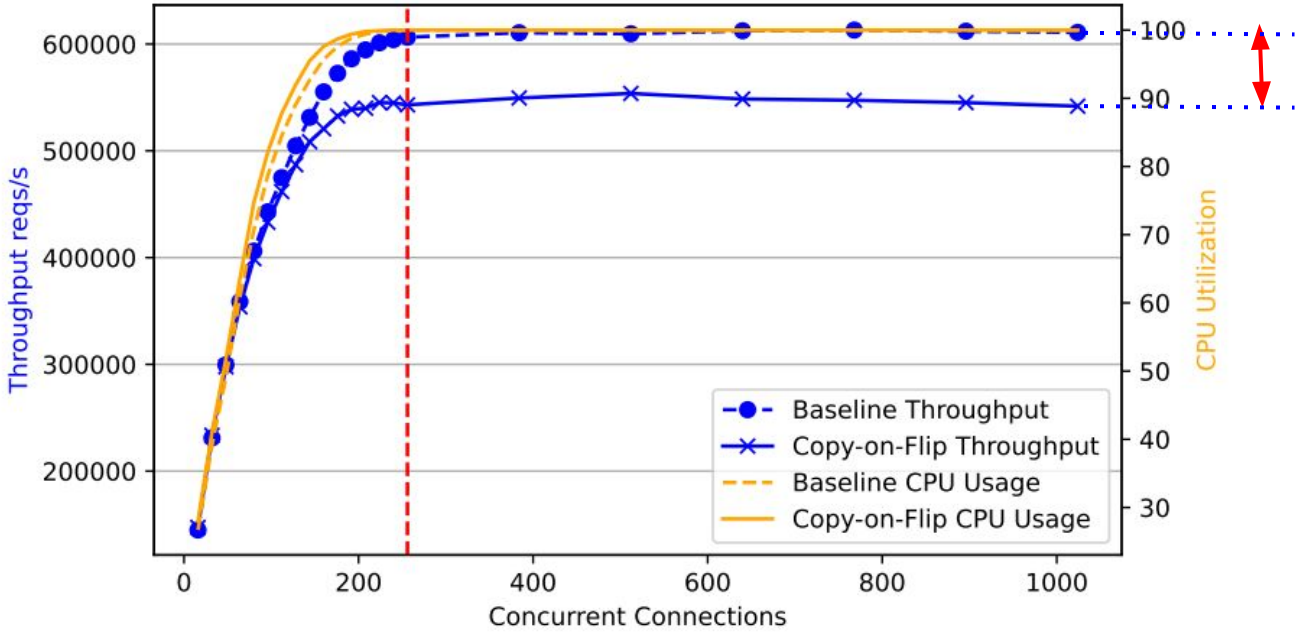
Evaluation – Performance

- SPEC CPU2017 geometric mean overhead: 0.2%
- LMBench geometric mean overhead: 1.9%
- Kraken on Google Chrome geometric mean overhead: 1.1%

Evaluation – Performance

- SPEC CPU2017 geometric mean overhead: 0.2%
- LMBench geometric mean overhead: 1.9%
- Kraken on Google Chrome geometric mean overhead: 1.1%
- Negligible memory overhead under normal conditions

Evaluation – Nginx



~10% median overhead

Conclusion

- Modern systems are still vulnerable to Rowhammer
- Copy-on-Flip design + open-source implementation
- Low overhead and high attack surface reduction

More in The Paper

- Linux implementation details
- More evaluation results
- Discussion on other OSes
- Paper: https://download.vusec.net/papers/cof_ndss23.pdf
- Code: https://github.com/vusec/Copy-on-Flip

@hammertux @vu5ec

Evaluation - Residual Attack Surface

- >95% pages are now protected in Copy-on-Flip
- Non-movable pages
  - DMA
  - Direct Linear Mapping

Evaluation – Performance Under Attack