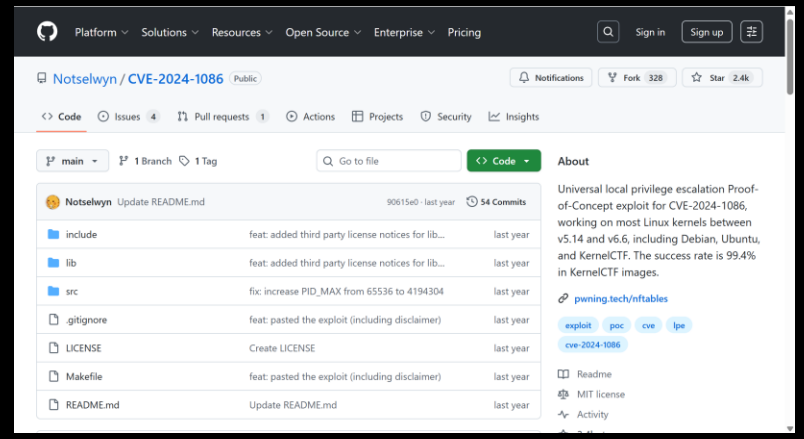
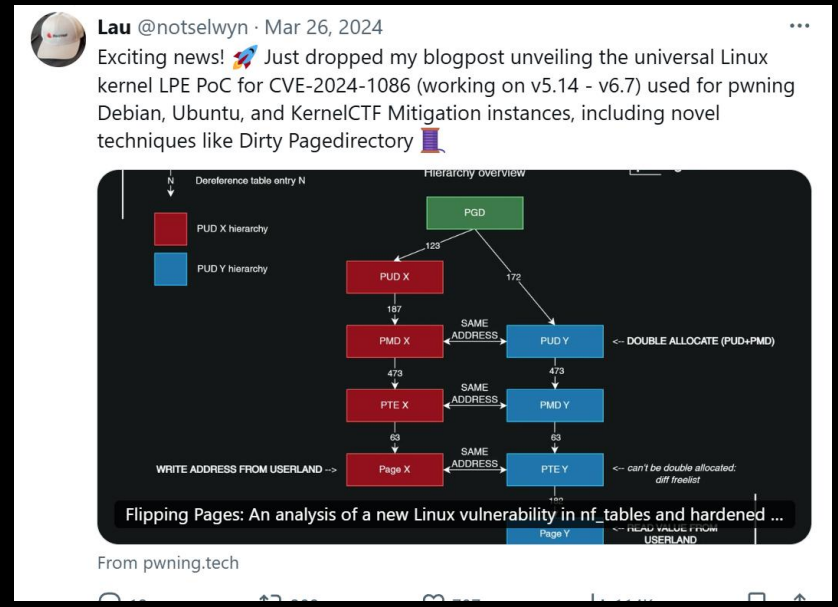
Table Manners:
Diving into Linux pagetables exp techniques

By Lau (@notselwyn)

$ whoami

- Linux enthusiast
- Twitter @notselwyn
- Feats:
  - Owner of pwning.tech
  - Initial ksmbd integration Syzkaller
  - Private ksmbd research
  - Published Linux LPE exploit in March 2024
- Work at <vr lab>





Why pagetable exploitation

- Trivial yet powerful exploit technique
- No address leak necessary
- Guaranteed kernel compromise
- **All Linux kernels**

-> Enables universal exploits

-> Low maintenance cost

Target audience

1. Linux researchers
2. Android researchers
3. ~~iOS researchers~~ (rip: PPL and SPTM)

No techniques were tested on Android or aarch64

All examples are for x64 Linux

Why this talk

I want to:

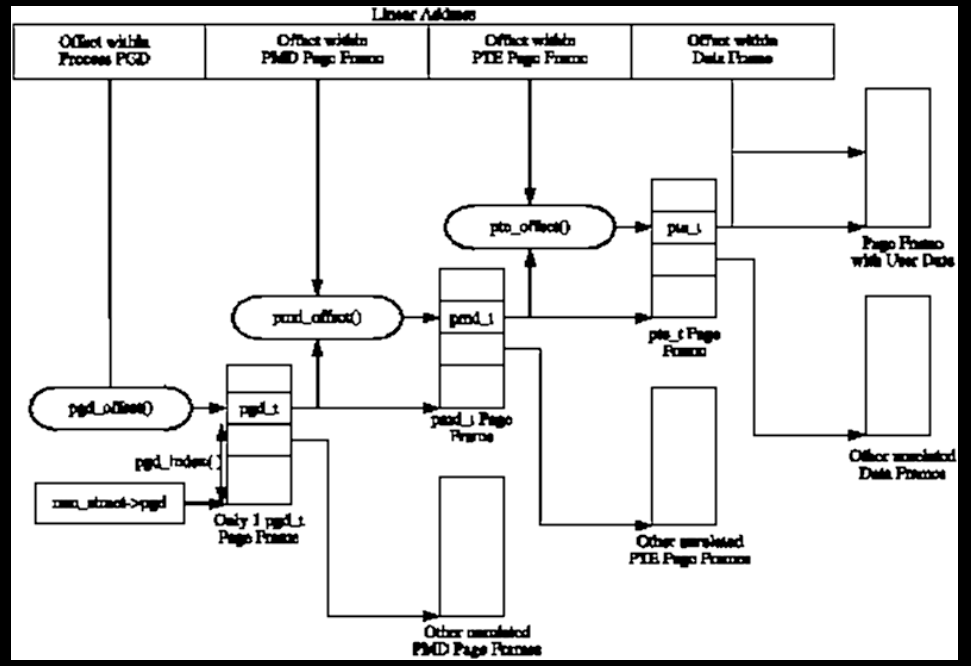
- Help build insane exploits
- Have fun with weird cpu quirks
- Pique your curiosity

Intro to pagetables

Swift recap of pagetables and pagetable entries

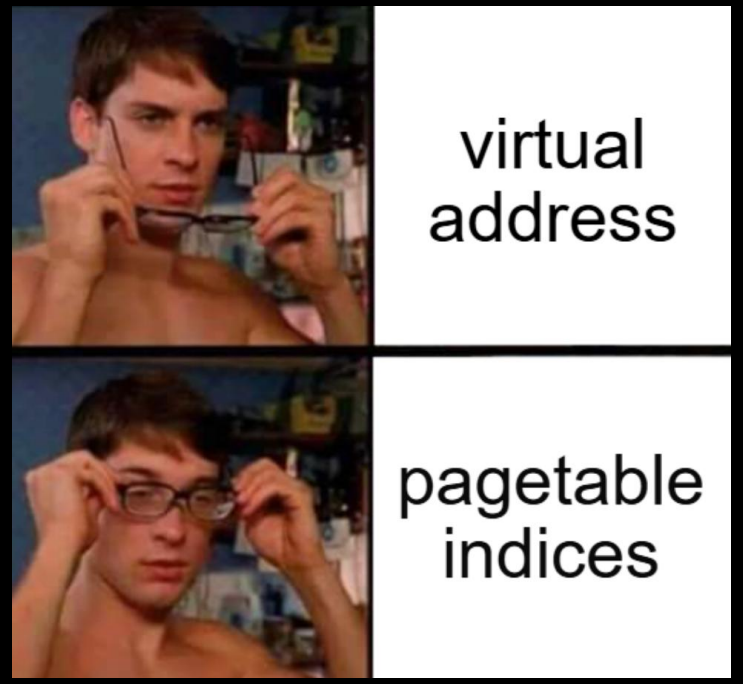
What even is a pagetable

> weird computer science thing
> look online
>



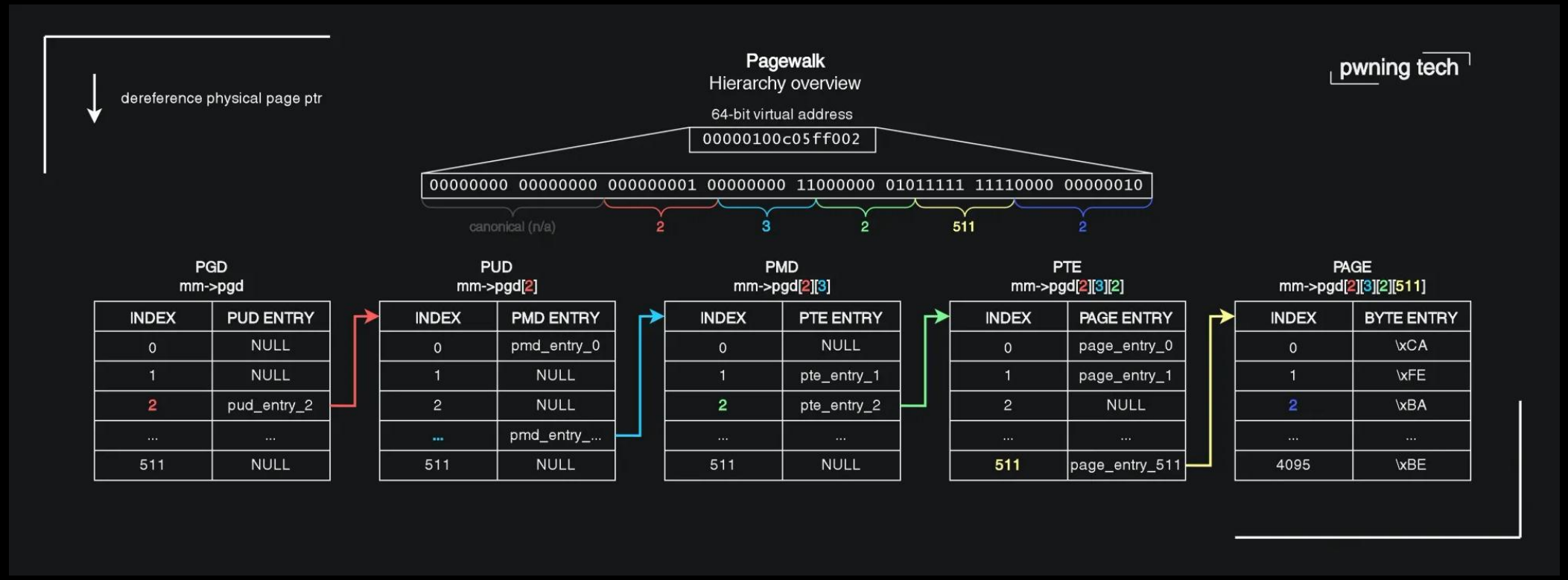
What even is a pagetable

- Converts virtual addresses to physical addresses
- Nested array
- Stores physical addresses
- Stores the page permissions
- Lookups are called a “pagewalk”



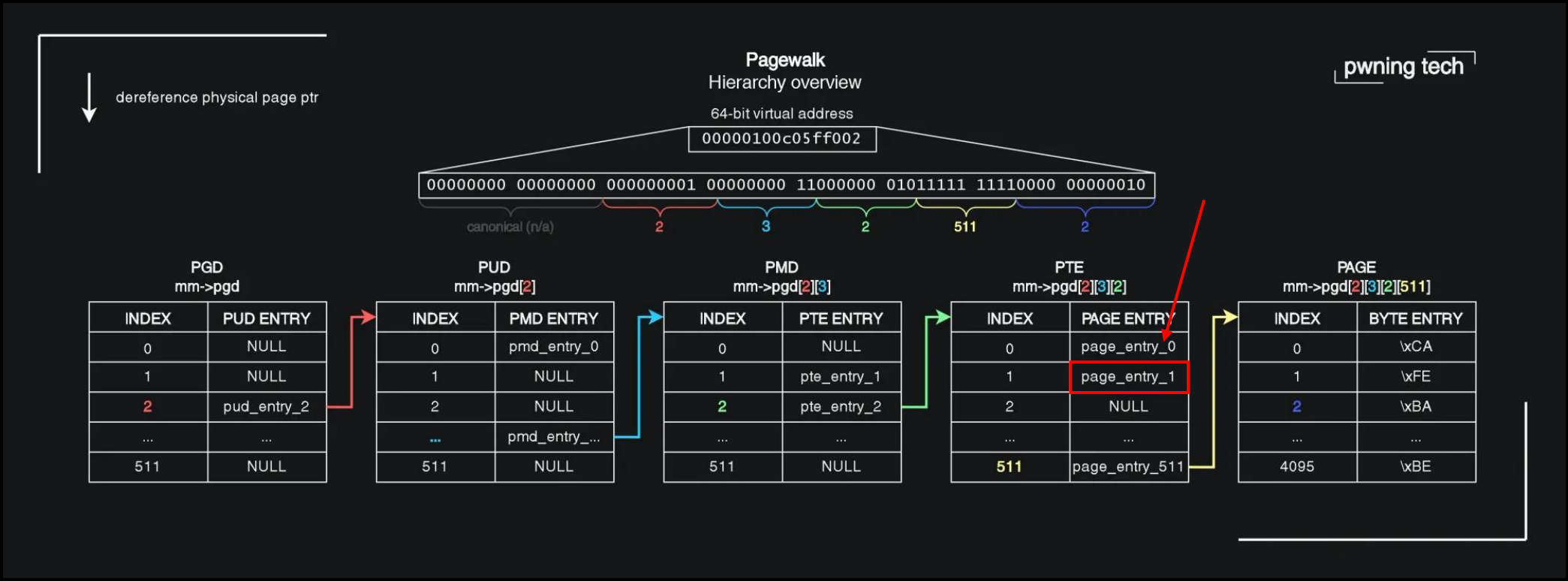
What even is a pagetable

- Virtual address = 4x 9 bit pagetable indices and page offset



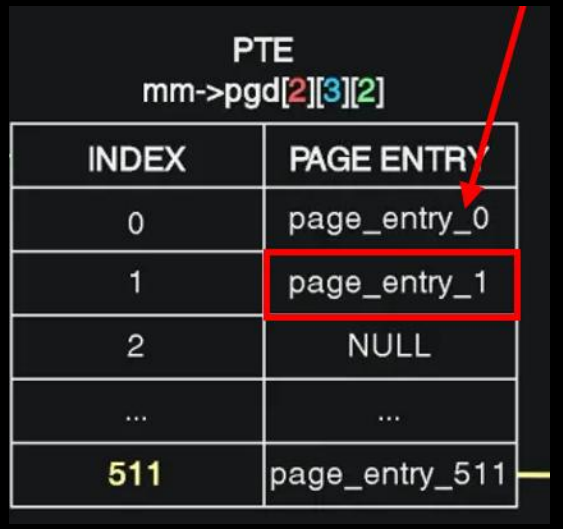
Pagetable entries

- Last pagetable contains “pagetable entries”
- Points to individual page



Pagetable entries

- Shortname is PTE entry
- 8 bytes each
- An entry consists of:
  - Physical page address (“PFN”)
  - Page permissions: read/write/execute
  - Obscure metadata (is page dirty, is page huge, ...)

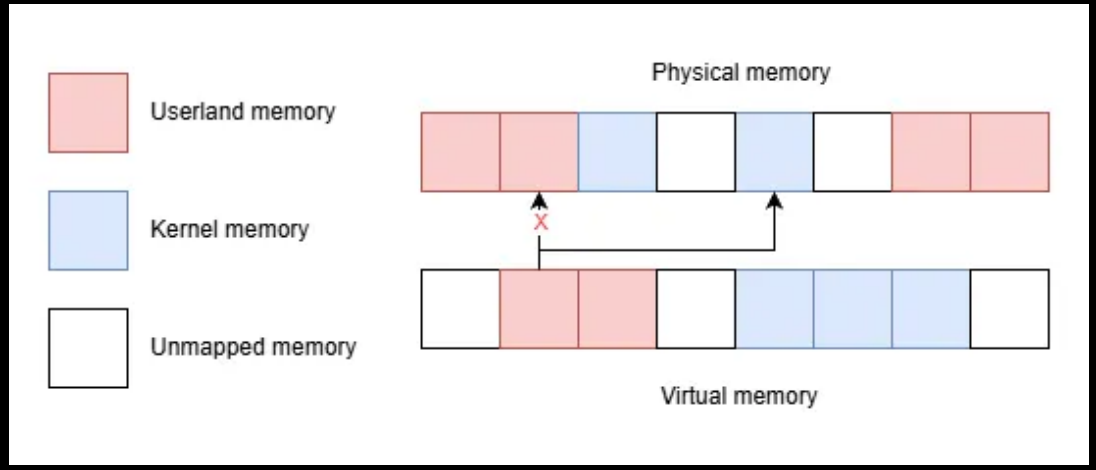


Pagetable exp techniques

Overview of interesting exploitation techniques

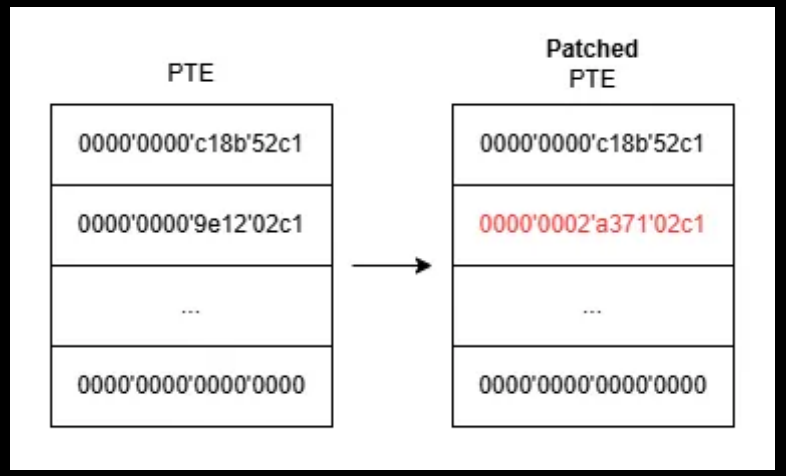
KSMA

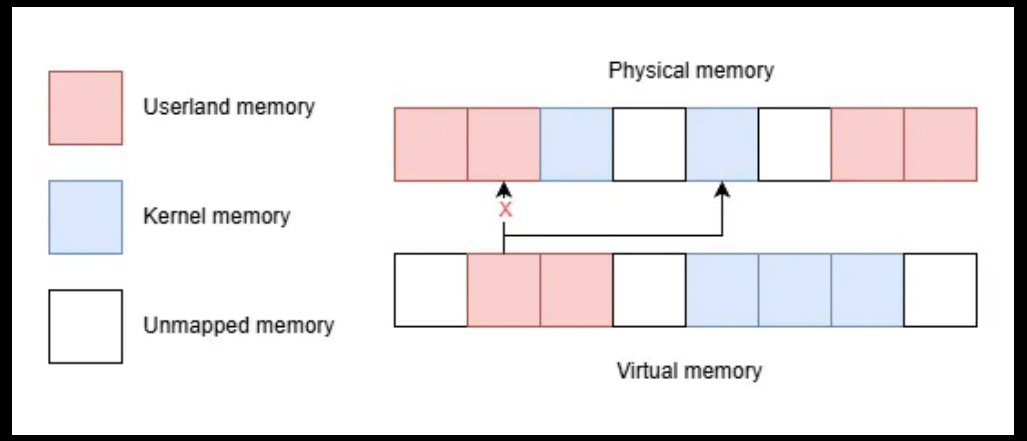
- Kernel space mirroring attack (KSMA)
- Category of pagetable techniques
- Ex: overwrite a PTE
- Map userland virtual memory to kernel physical memory
- Read/write kernel memory



Dirty Pagetable – 1/3

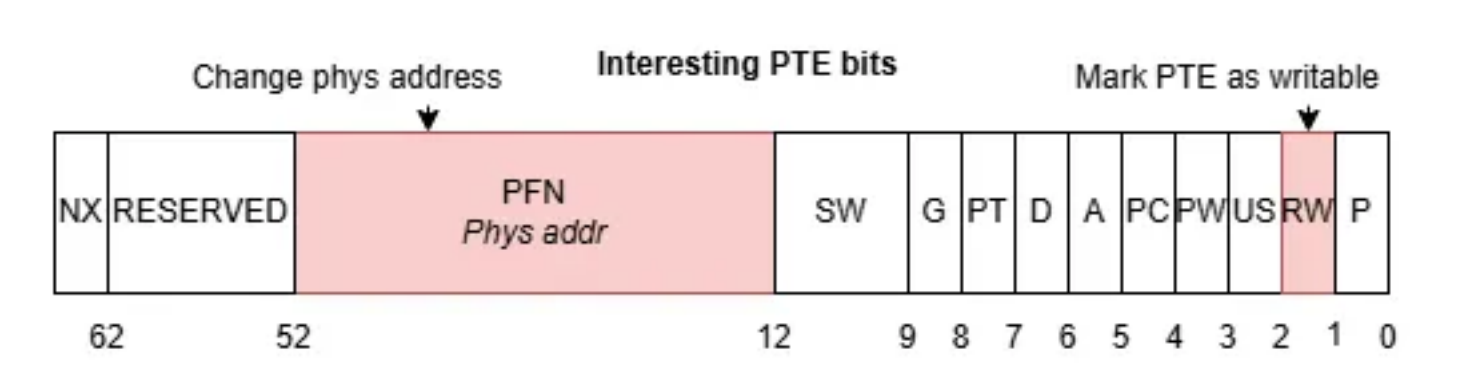
- Overwrite user PTE entry with limited kernel write primitive
- Access a kernel page in userland
- Overwrite userland PTE entry's physical address
- Requires knowing phys kernel base address





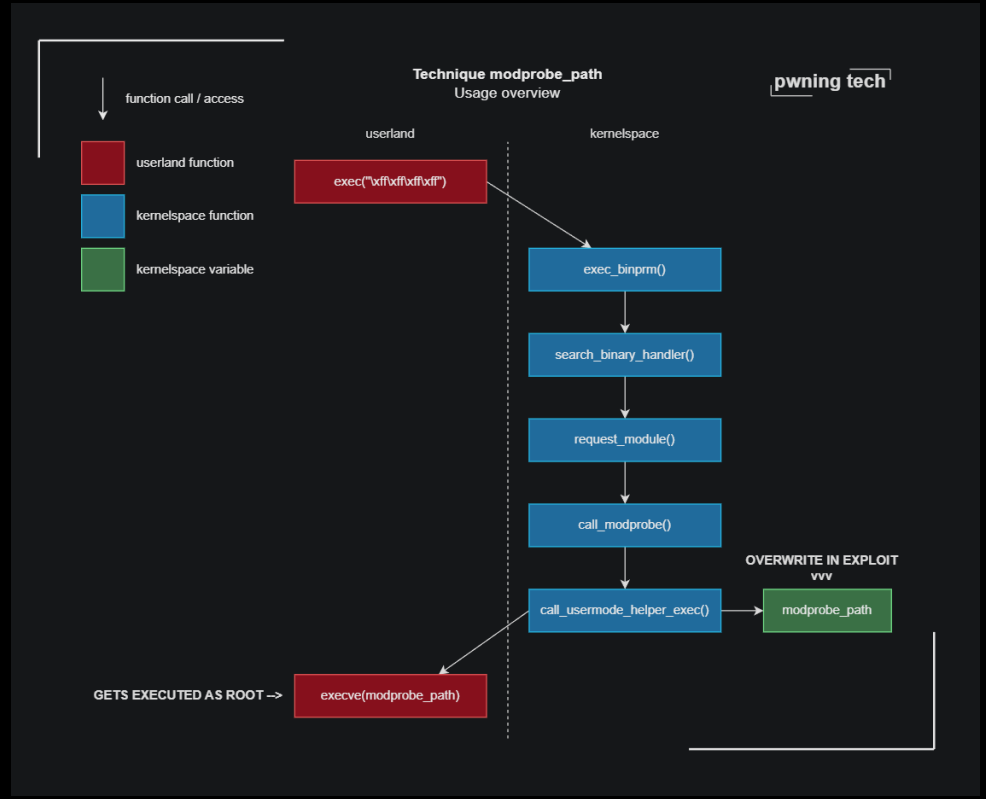
Dirty Pagetable – 2/3

- Use it to overwrite data
- You can write to read-only data
- Can overwrite code too, but not on modern Android



Dirty Pagetable – 3/3

- Getting root shell
- Overwrite modprobe_path
- Overwrite usermode helper path
- Overwrite core_pattern:
  - No pid bruteforce needed for ns escape
  - Aka Docker

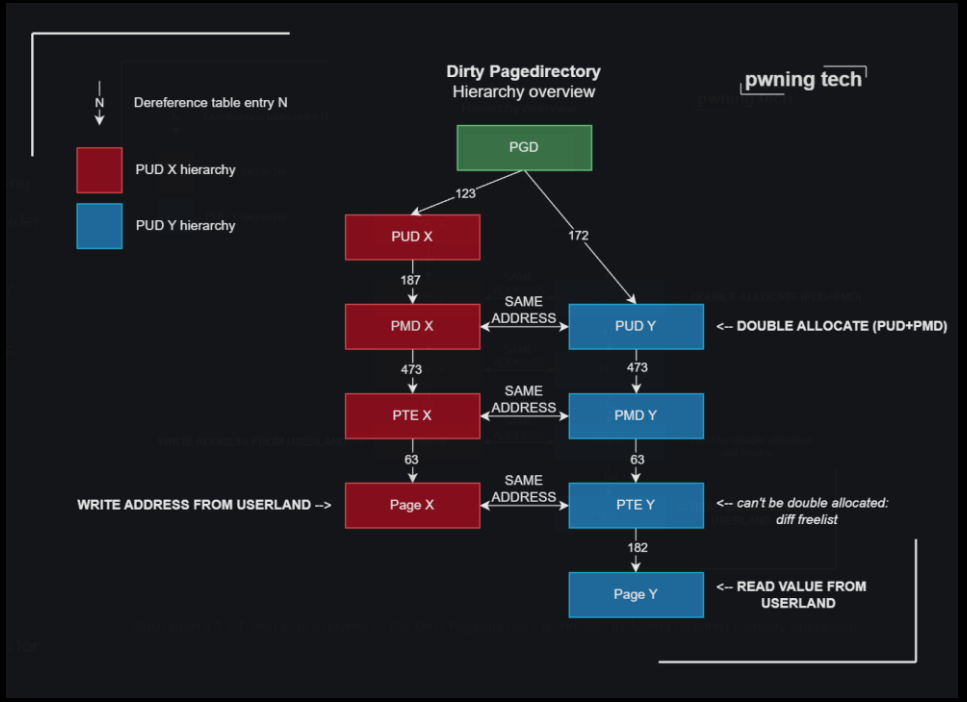


Advanced pagetable techniques

Nice to know techniques that can make really good exploits

Dirty Pagedirectory

- Overlap an PMD page with an PTE page using double-free
- Forge PTE entries from userland
- Can read/write all physical memory



File permissions 1/3

- Overwrite a read-only file using PTE entry corruption
- Address leak may not be required
- Example: disable root password in /etc/passwd
- Android idea: overwrite binaries & .so ran by root

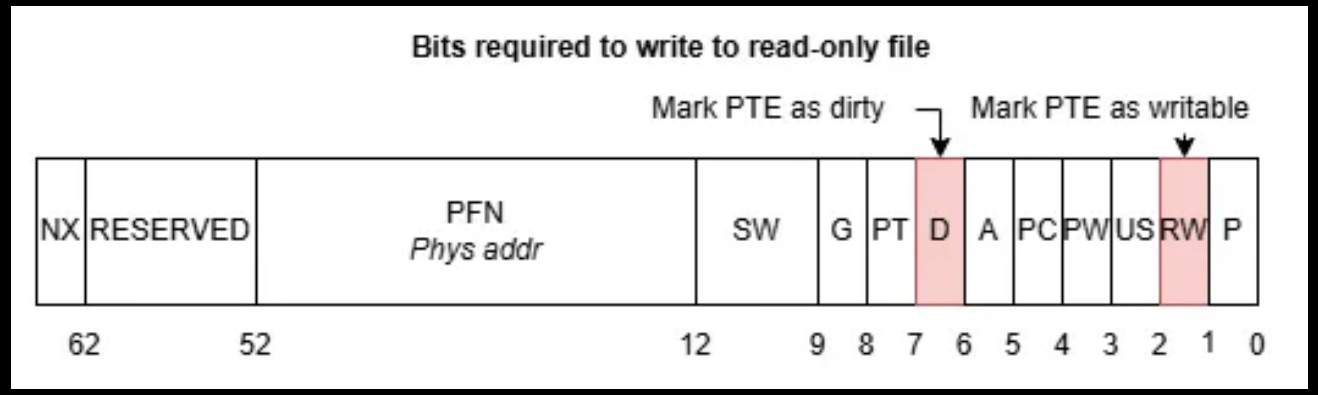
```
user@syzkaller:~$ /mnt/rofile-exploit/exploit
[*] first 8 bytes of 'x.txt': 'hellowor'
[*] press enter to continue exploit

[*] writing AAAABBBB to file...
[*] closing file...
[*] reopening file...
[*] first 8 bytes of 'x.txt': 'AAAABBBB'
```

File permissions 2/3

1. Load a file with mmap as read-only
2. Overwrite its PTE entry with writable and dirty flags*
3. Write to file's virt mem

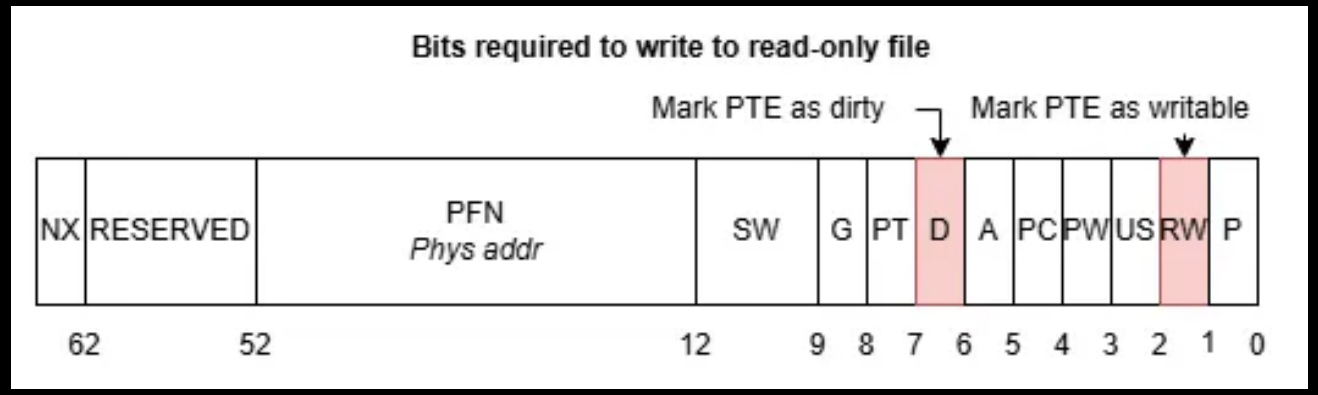
*Dirty flag: write-back to disk



File permissions 3/3

- If 1-byte write primitive, no address leak is necessary
- Else address leak is necessary

https://ptr-yudai.hatenablog.com/entry/2025/09/14/180326

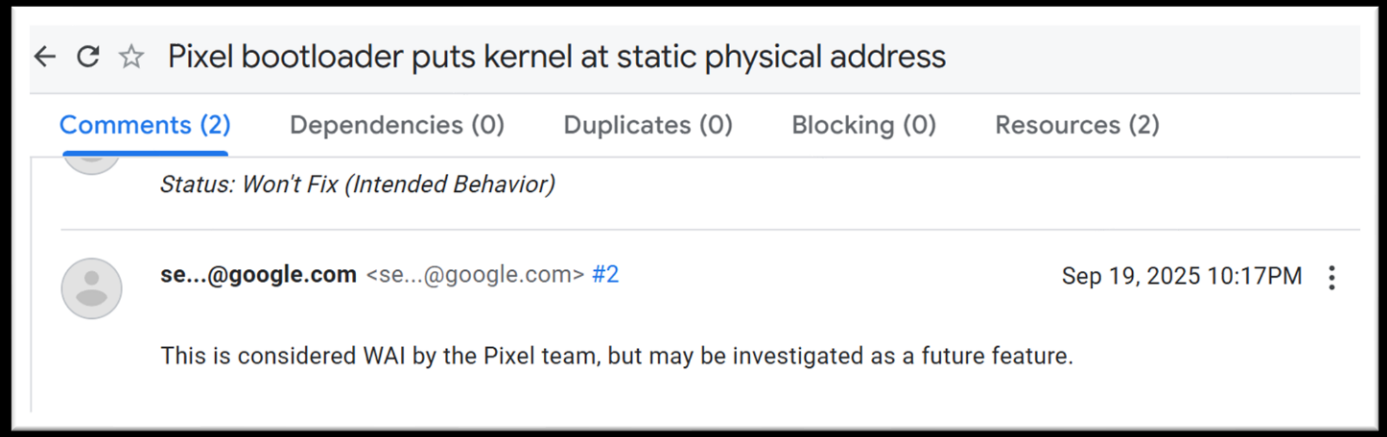


Trivia

Fun facts for building exploits

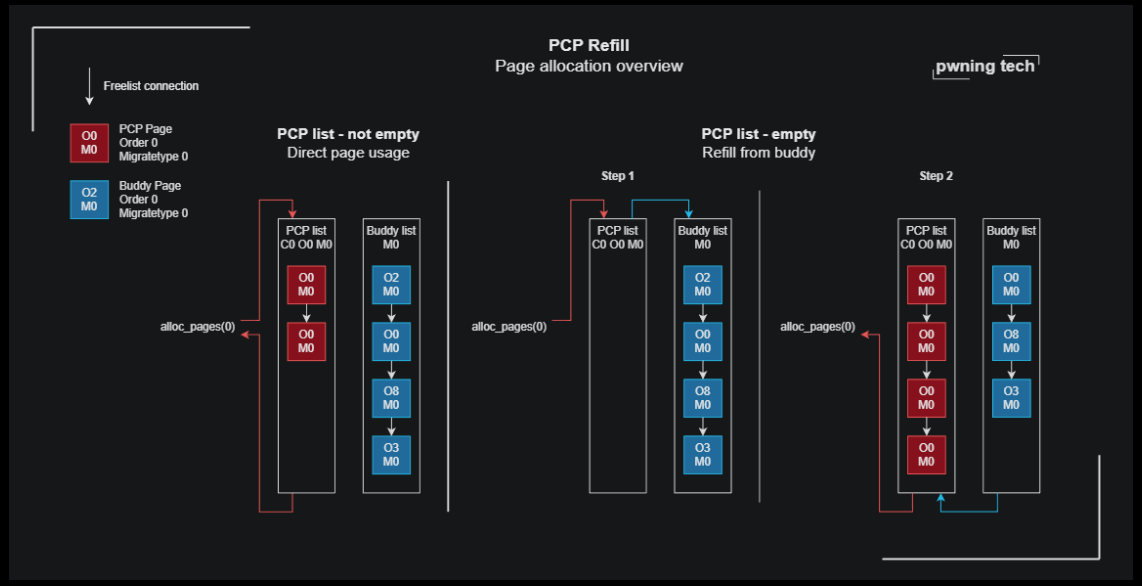
Trivia – Google Pixels

- Google Pixels do not have physical kaslr
- Root Pixels with an 8-byte write!(?)
- No leaks required



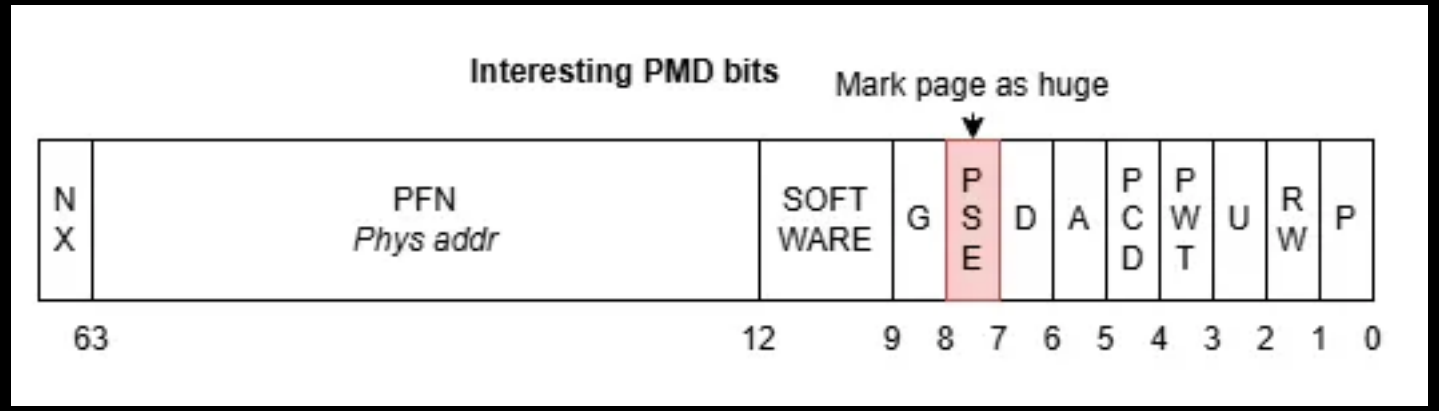
Trivia – Heap shaping

- Pagetables are allocated using the PCP allocator
- No slab allocator mitigations
- Allows for easier UAF-write and OOB-write exploitation



Trivia – Huge pages

- PUD and PMD entries allow for huge pages
- Allows directly mapping 1 GiB and 2MiB
- Access the entire kernel image with a single overwrite !



Trivia – Preventing crashes

- You can access memory that’s not mmap’ed if it has valid PTs
- No segfaults etc
- To prevent crashes: forge PTEs outside valid memory
- The process will cleanly exit

Nice pagetables resources

- https://sam4k.com/page-table-kernel-exploitation/ ← Overview
- https://pwning.tech/nftables ← Univ. Linux LPE
- https://kuzey.rs/posts/Dirty_Page_Table/ ← Code overwrite
- https://www.longterm.io/samsung_rkp.html ← Samsung HV

Thanks for listening

- It was a huge honor speaking for you all
- Hope you learned something new
- Questions?

Twitter: @notselwyn
Mail: notselwyn@pwning.tech