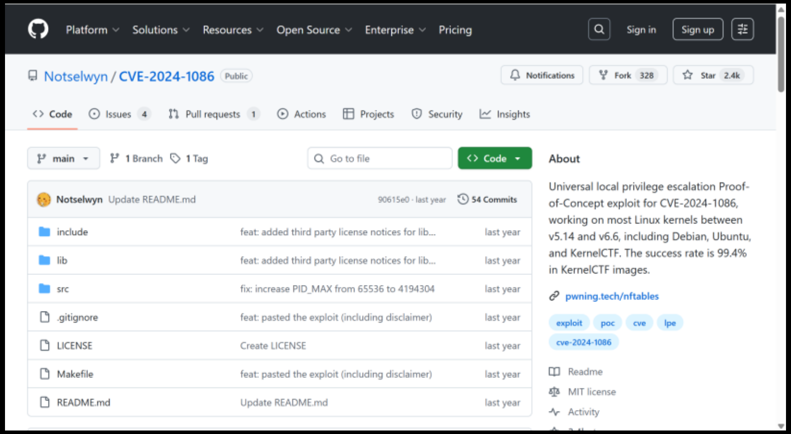
Table Manners:
Diving into Linux pagetables exp techniques

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>



Notselwyn / CVE-2024-1086 Public

Notselwyn Update README.md 90615e0 · last year 54 Commits

include	feat: added third party license notices for lib...	last year
lib	feat: added third party license notices for lib...	last year
src	fix: increase PID_MAX from 65536 to 4194304	last year
.gitignore	feat: pasted the exploit (including disclaimer)	last year
LICENSE	Create LICENSE	last year
Makefile	feat: pasted the exploit (including disclaimer)	last year
README.md	Update README.md	last year

About

Universal local privilege escalation Proof-of-Concept exploit for CVE-2024-1086, working on most Linux kernels between v5.14 and v6.6, including Debian, Ubuntu, and KernelCTF. The success rate is 99.4% in KernelCTF images.

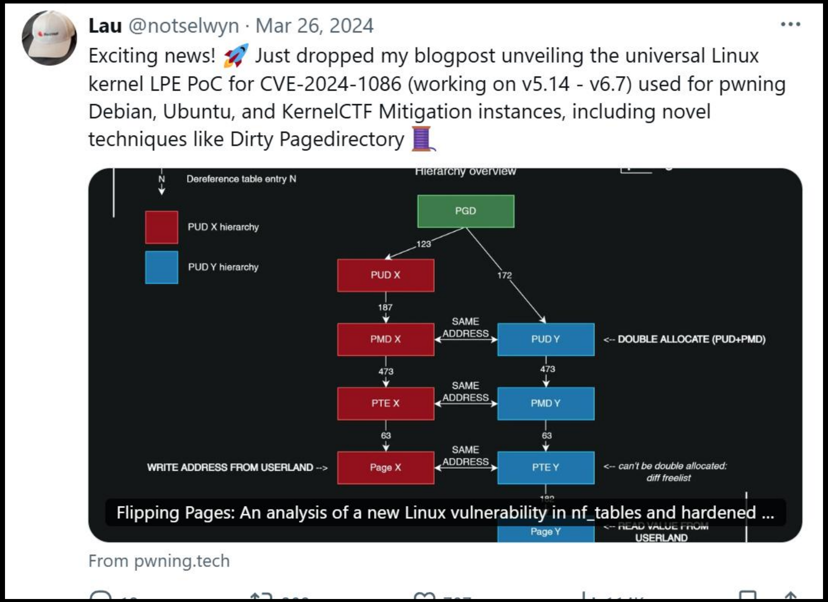
pwning.tech/nftables

exploit poc cve lpe cve-2024-1086

Readme

MIT license

Activity



Lau @notselwyn · Mar 26, 2024

Exciting news! 🚀 Just dropped my blogpost unveiling the universal Linux kernel LPE PoC for CVE-2024-1086 (working on v5.14 - v6.7) used for pwning Debian, Ubuntu, and KernelCTF Mitigation instances, including novel techniques like Dirty Pagedirectory 🧵

Hierarchy overview
N
Dereference table entry N
PUD X hierarchy
PUD Y hierarchy
PGD
123
172
PUD X
187
PMD X
SAME ADDRESS
PUD Y
<-- DOUBLE ALLOCATE (PUD+PMD)
473
473
PTE X
SAME ADDRESS
PMD Y
63
63
WRITE ADDRESS FROM USERLAND -->
Page X
SAME ADDRESS
PTE Y
<-- can't be double allocated: diff freelist
182
Page Y
<-- READ VALUE FROM USERLAND

Flipping Pages: An analysis of a new Linux vulnerability in nf_tables and hardened ...

From pwning.tech

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:

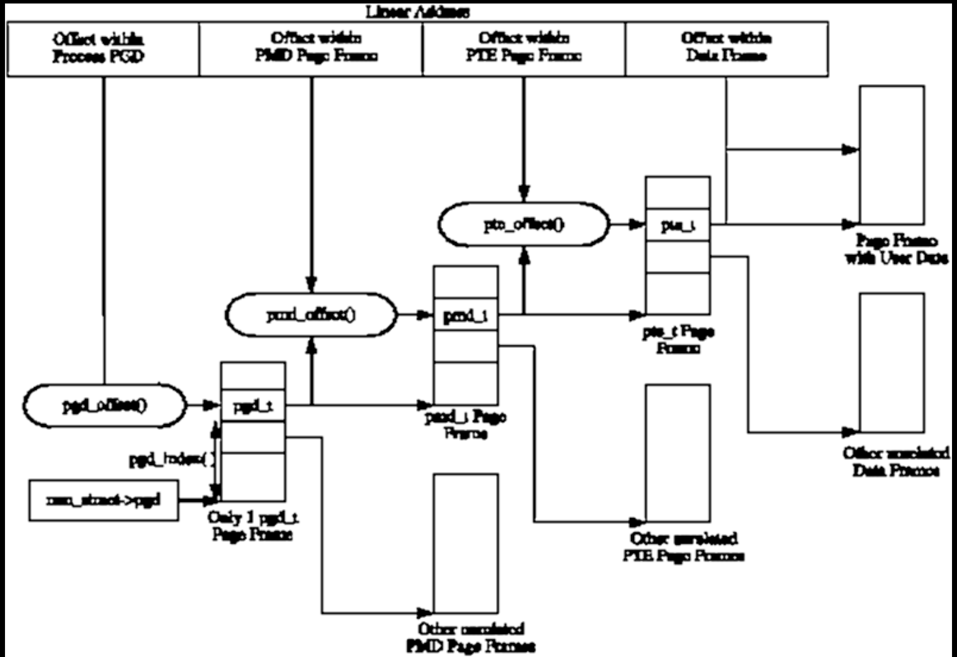
- Help build insane exploits
- Have fun with weird cpu quirks
- Pique your curiosity

Intro to pagetables

Swift recap of pagetables and pagetable entries

What even is a pagetable

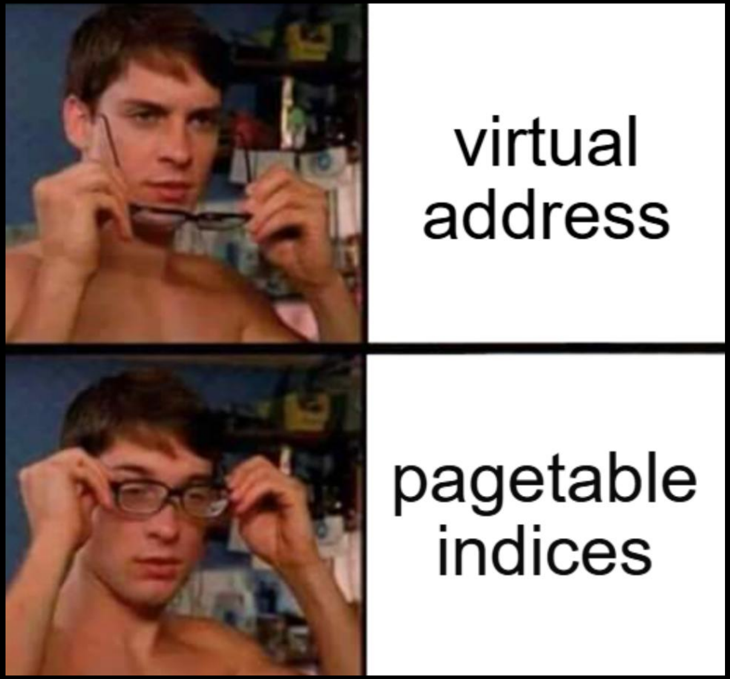
> weird computer science thing
> look online
>



Linear Address
Offset within Process PGD
Offset within PMD Page Frame
Offset within PTE Page Frame
Offset within Data Frame
pte_offset()
pte_t
Page Frame with User Data
pmd_offset()
pmd_t
pte_t Page Frame
pgd_offset()
pgd_t
pmd_t Page Frame
pgd_index()
mm_struct->pgd
Only 1 pgd_t Page Frame
Other unrelated Data Frames
Other unrelated PTE Page Frames
Other unrelated PMD Page Frames

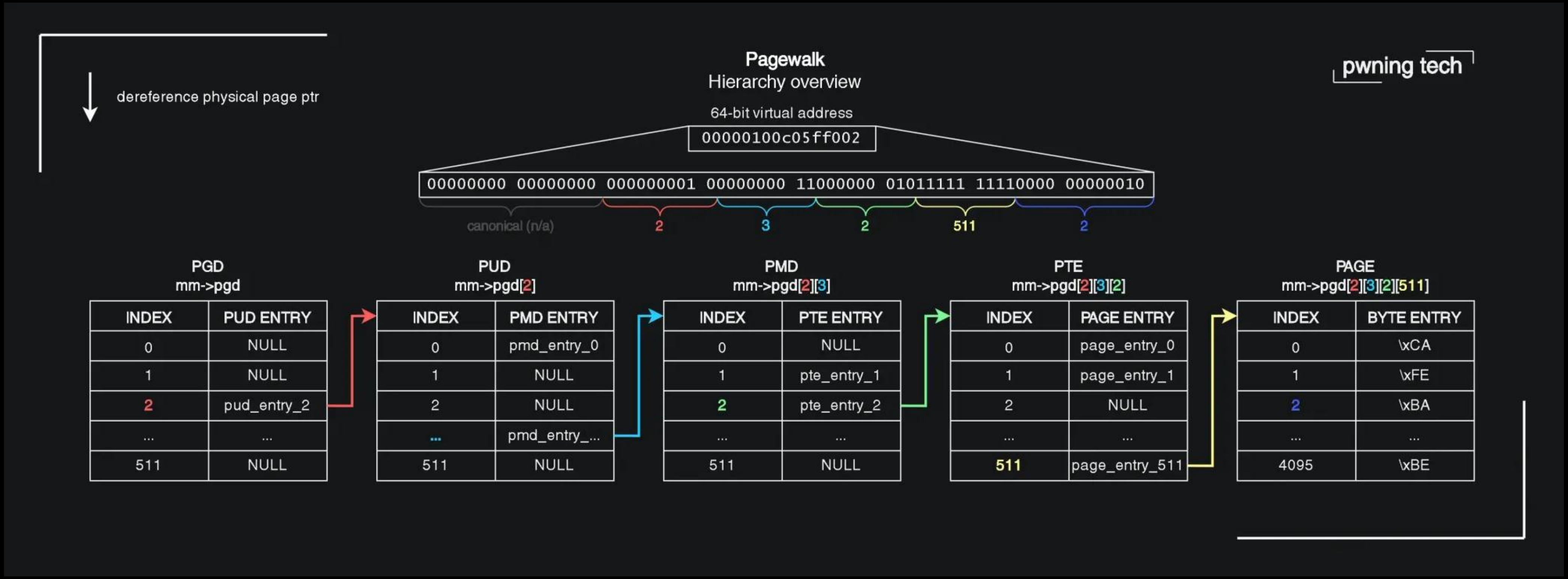
What even is a pagetable

- 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



dereference physical page ptr

pwning tech

Pagewalk
Hierarchy overview

64-bit virtual address

00000100c05ff002

00000000 00000000 000000001 00000000 11000000 01011111 11110000 00000010

PGD
mm->pgd

INDEX	PUD ENTRY
0	NULL
1	NULL
2	pud_entry_2
...	...
511	NULL

PUD
mm->pgd[2]

INDEX	PMD ENTRY
0	pmd_entry_0
1	NULL
2	NULL
...	pmd_entry_...
511	NULL

PMD
mm->pgd[2][3]

INDEX	PTE ENTRY
0	NULL
1	pte_entry_1
2	pte_entry_2
...	...
511	NULL

PTE
mm->pgd[2][3][2]

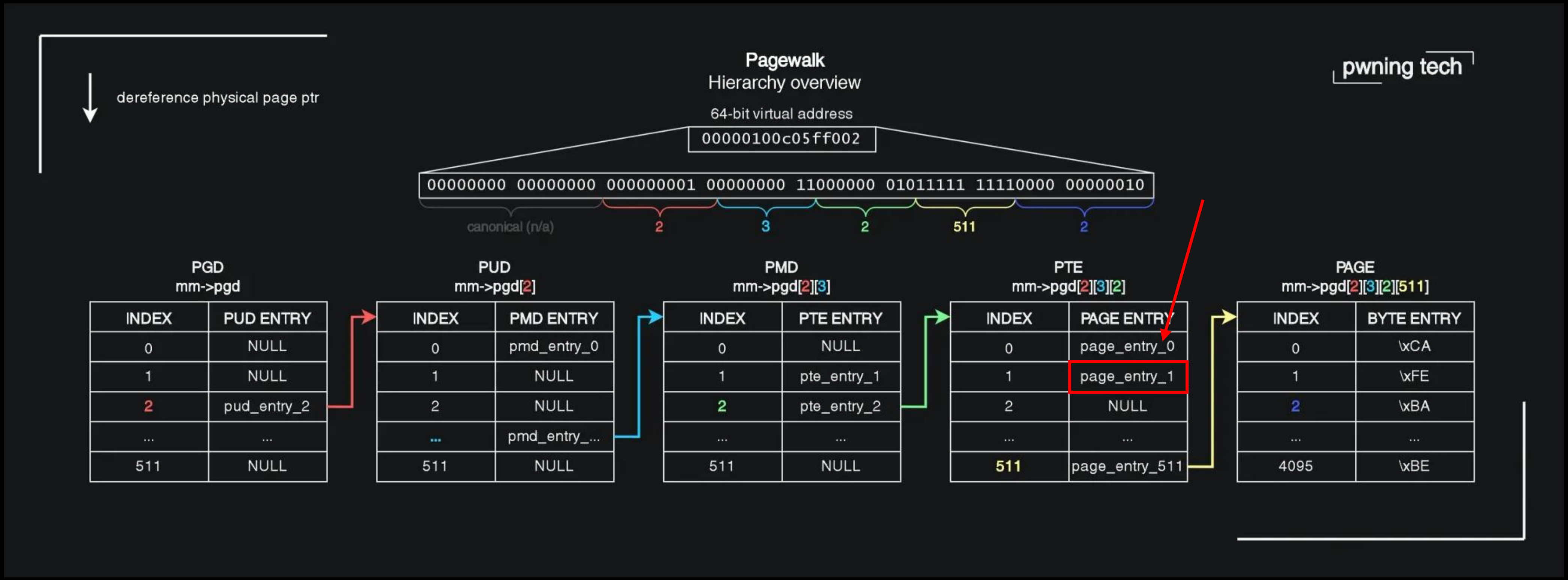
INDEX	PAGE ENTRY
0	page_entry_0
1	page_entry_1
2	NULL
...	...
511	page_entry_511

PAGE
mm->pgd[2][3][2][511]

INDEX	BYTE ENTRY
0	\xCA
1	\xFE
2	\xBA
...	...
4095	\xBE

Pagetable entries

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



PGD
mm->pgd

INDEX	PUD ENTRY
0	NULL
1	NULL
2	pud_entry_2
...	...
511	NULL

PUD
mm->pgd[2]

INDEX	PMD ENTRY
0	pmd_entry_0
1	NULL
2	NULL
...	pmd_entry_...
511	NULL

PMD
mm->pgd[2][3]

INDEX	PTE ENTRY
0	NULL
1	pte_entry_1
2	pte_entry_2
...	...
511	NULL

PTE
mm->pgd[2][3][2]

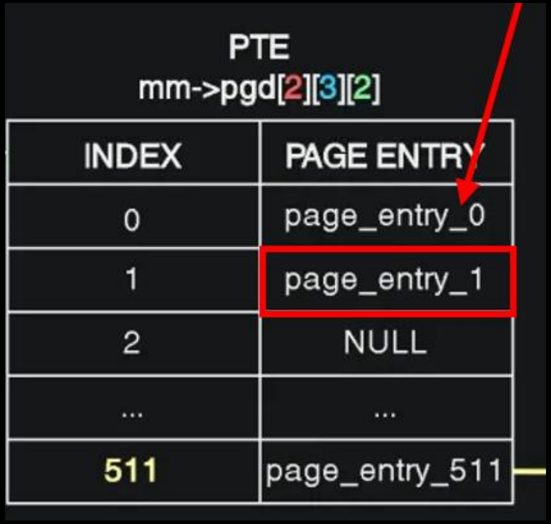
INDEX	PAGE ENTRY
0	page_entry_0
1	page_entry_1
2	NULL
...	...
511	page_entry_511

PAGE
mm->pgd[2][3][2][511]

INDEX	BYTE ENTRY
0	\xCA
1	\xFE
2	\xBA
...	...
4095	\xBE

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, …)



PTE
mm->pgd[2][3][2]

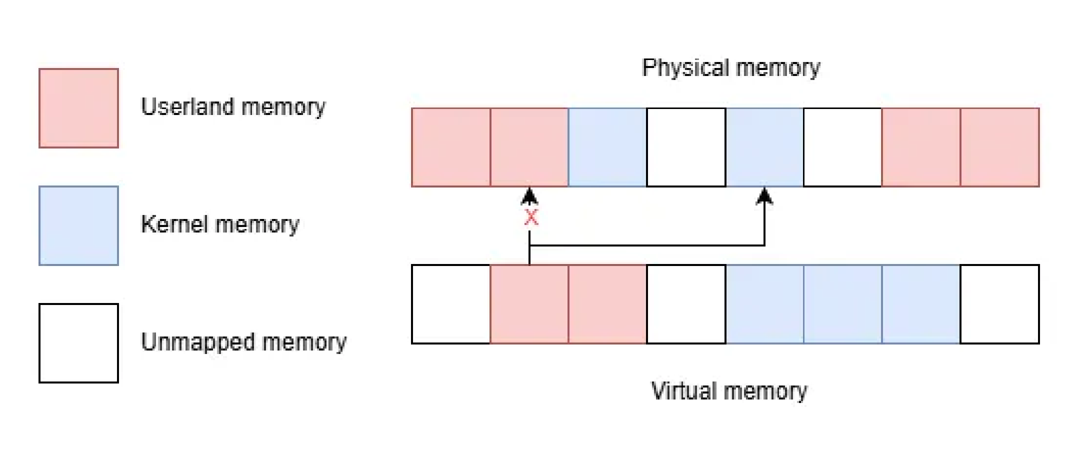
INDEX	PAGE ENTRY
0	page_entry_0
1	page_entry_1
2	NULL
...	...
511	page_entry_511

Pagetable exp techniques

Overview of interesting exploitation techniques

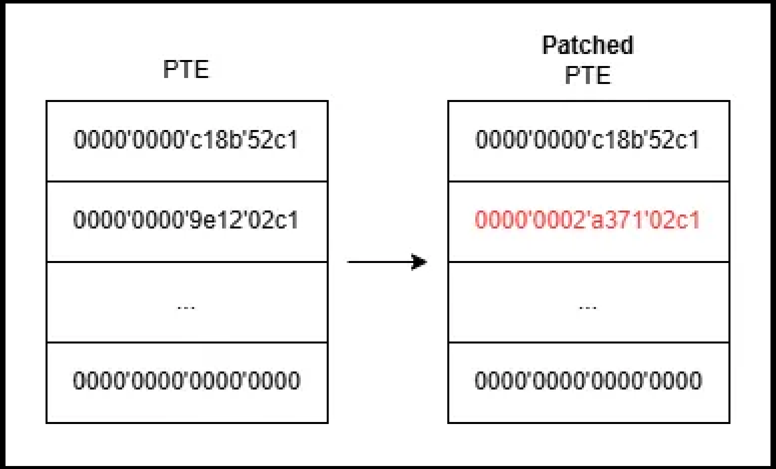
KSMA

- 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

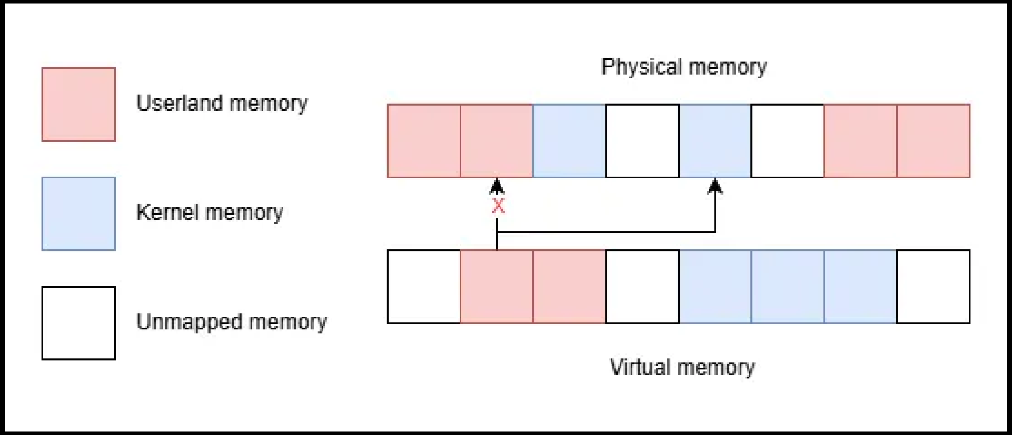


Dirty Pagetable – 1/3

- 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

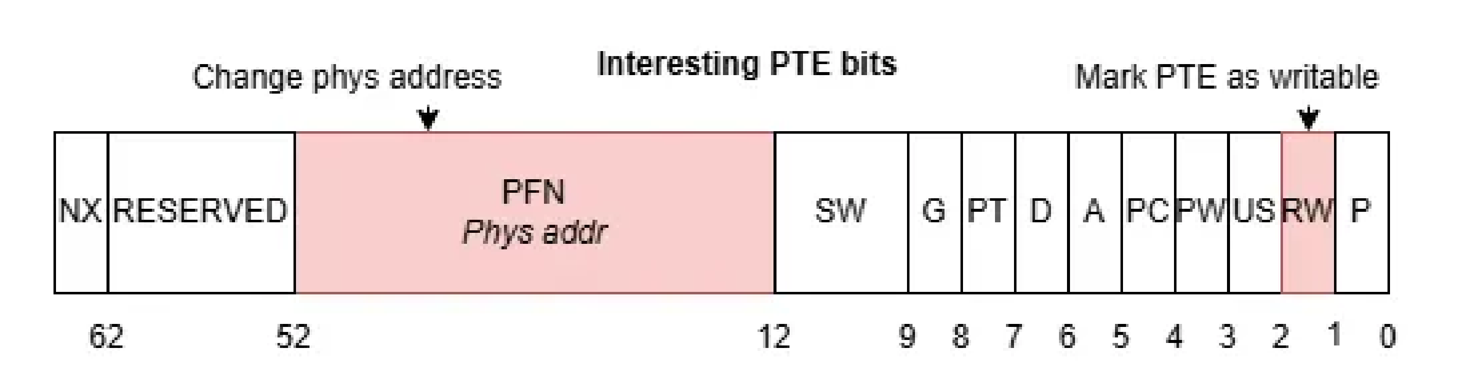


PTE	Patched PTE
0000'0000'c18b'52c1	0000'0000'c18b'52c1
0000'0000'9e12'02c1	0000'0002'a371'02c1
...	...
0000'0000'0000'0000	0000'0000'0000'0000



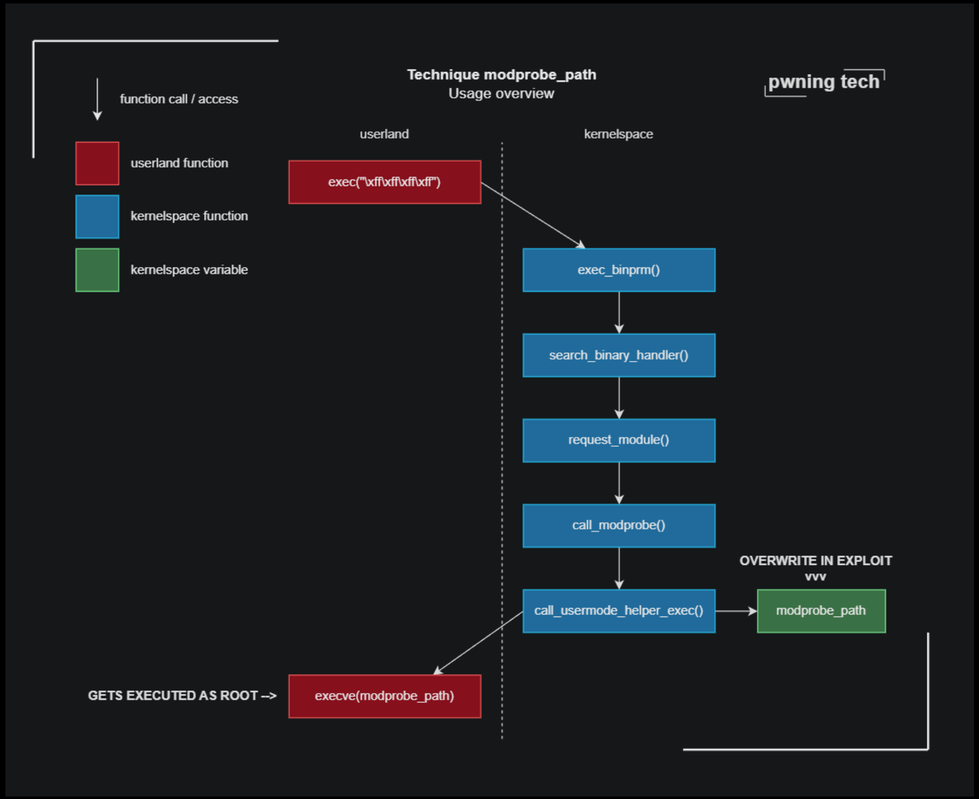
Dirty Pagetable – 2/3

- 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

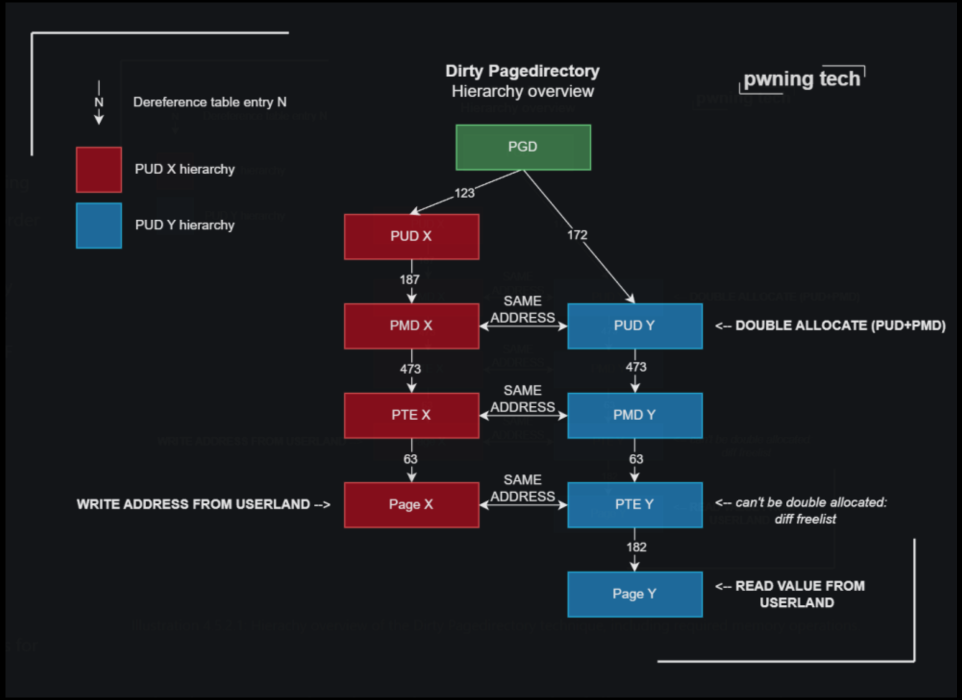


Advanced pagetable techniques

Nice to know techniques that can make really good exploits

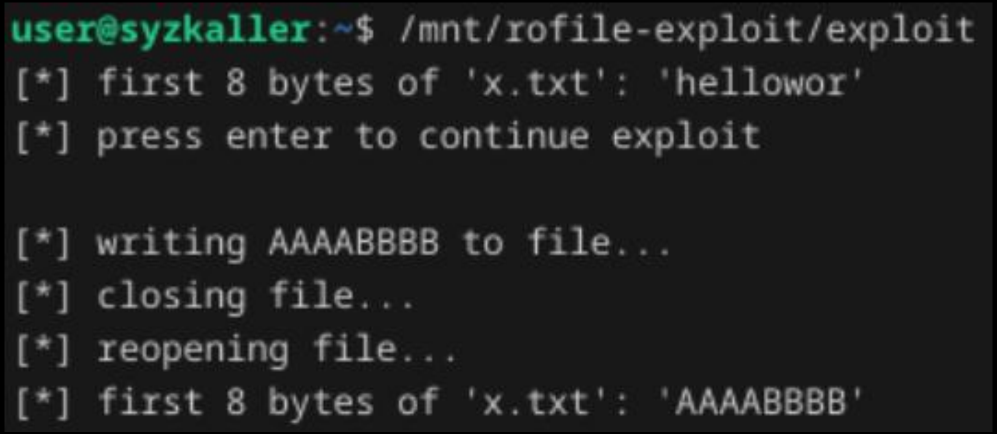
Dirty Pagedirectory

- 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



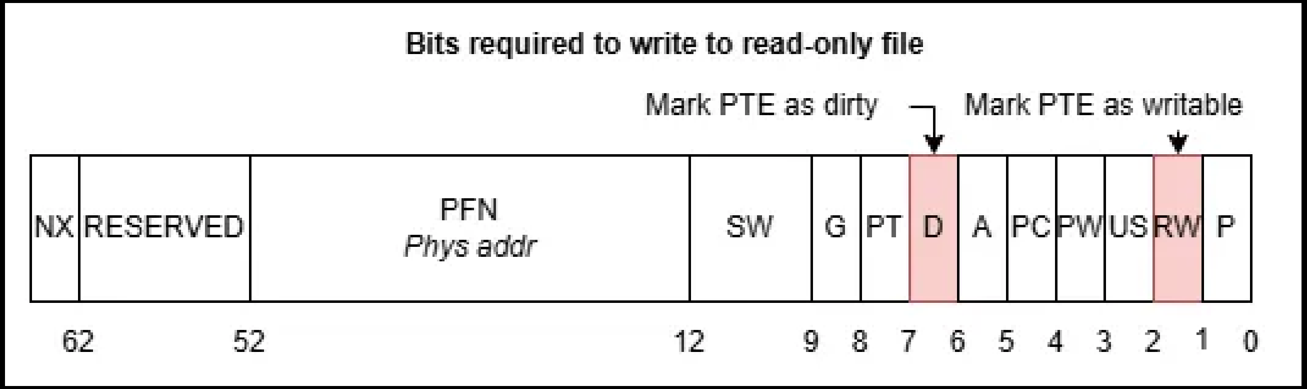
```
user@syzkaller:~$ /mnt/rofile-exploit/exploit
[*] first 8 bytes of 'x.txt': 'hellowor'
[*] press enter to continue exploit

[*] 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

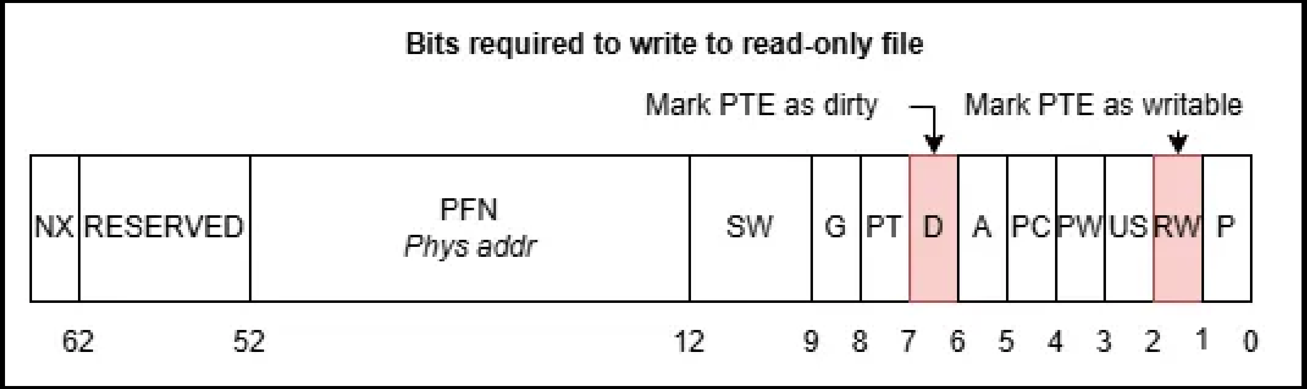
*Dirty flag: write-back to disk



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



Bits required to write to read-only file

Mark PTE as dirty

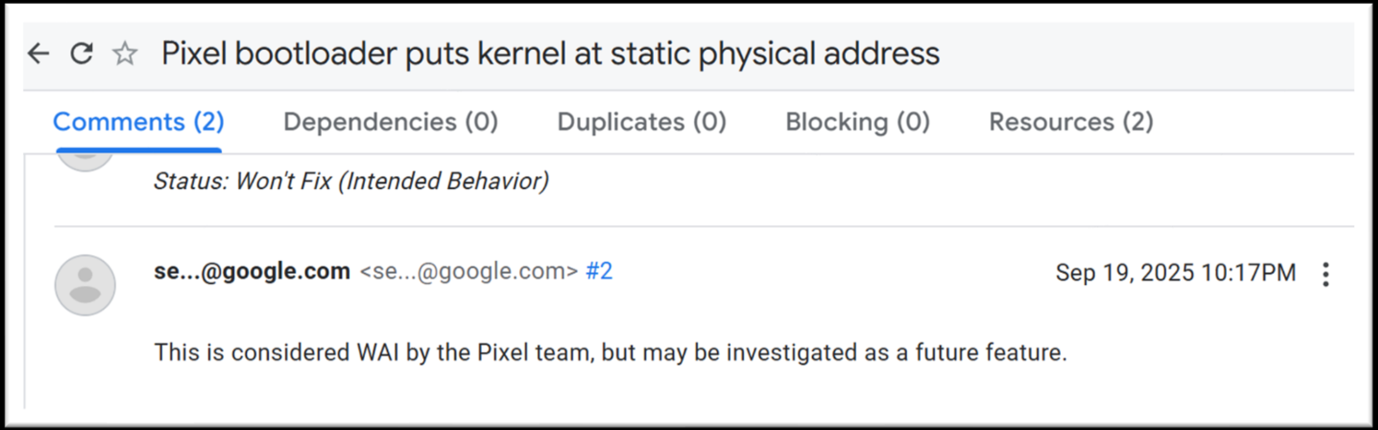
Mark PTE as writable

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



Pixel bootloader puts kernel at static physical address

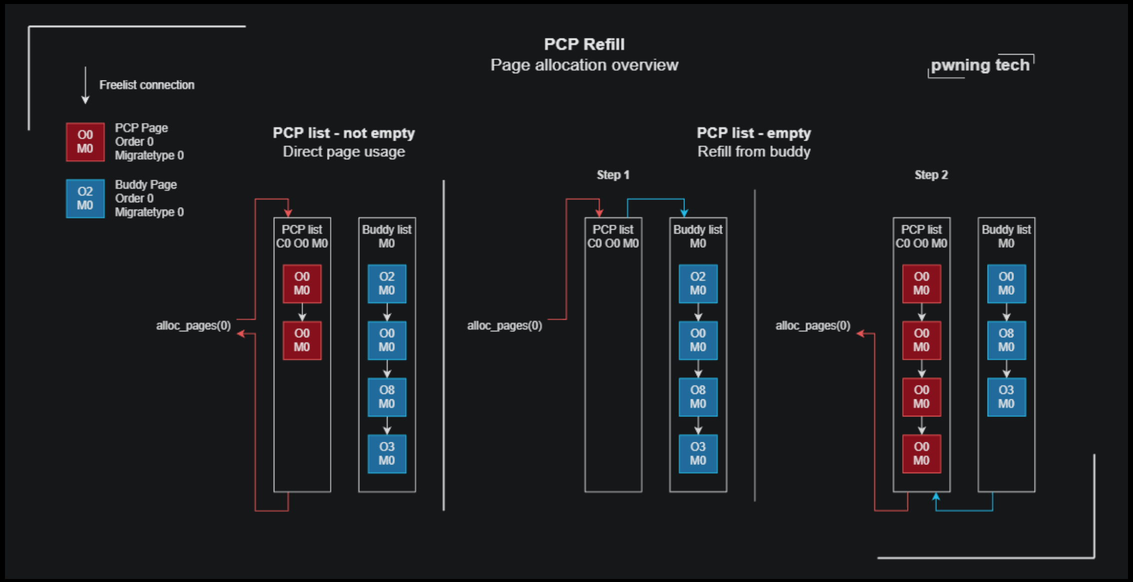
Status: Won't Fix (Intended Behavior)

se...@google.com <se...@google.com> #2 Sep 19, 2025 10:17PM

This is considered WAI by the Pixel team, but may be investigated as a future feature.

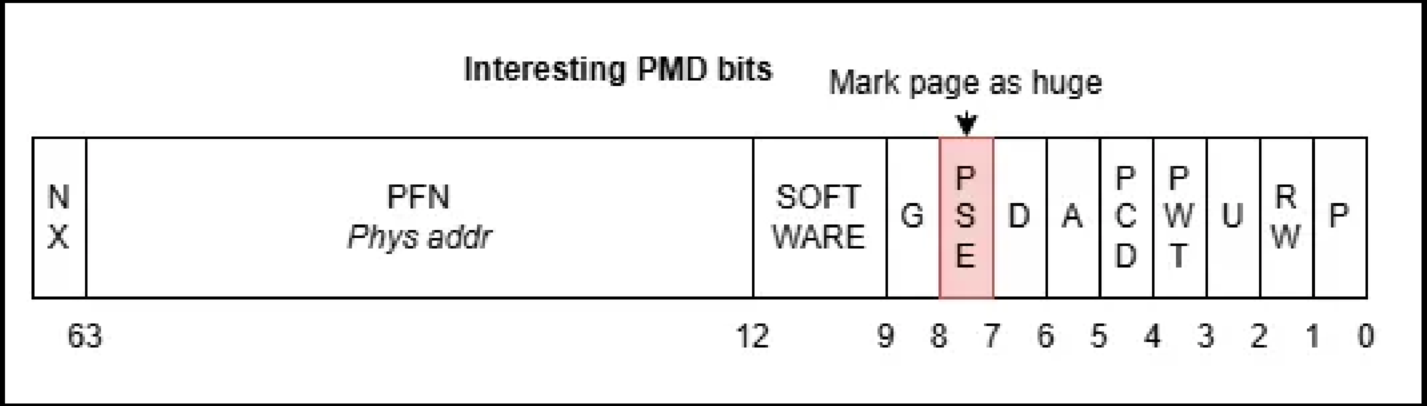
Trivia – Heap shaping

- 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

Audience puzzle

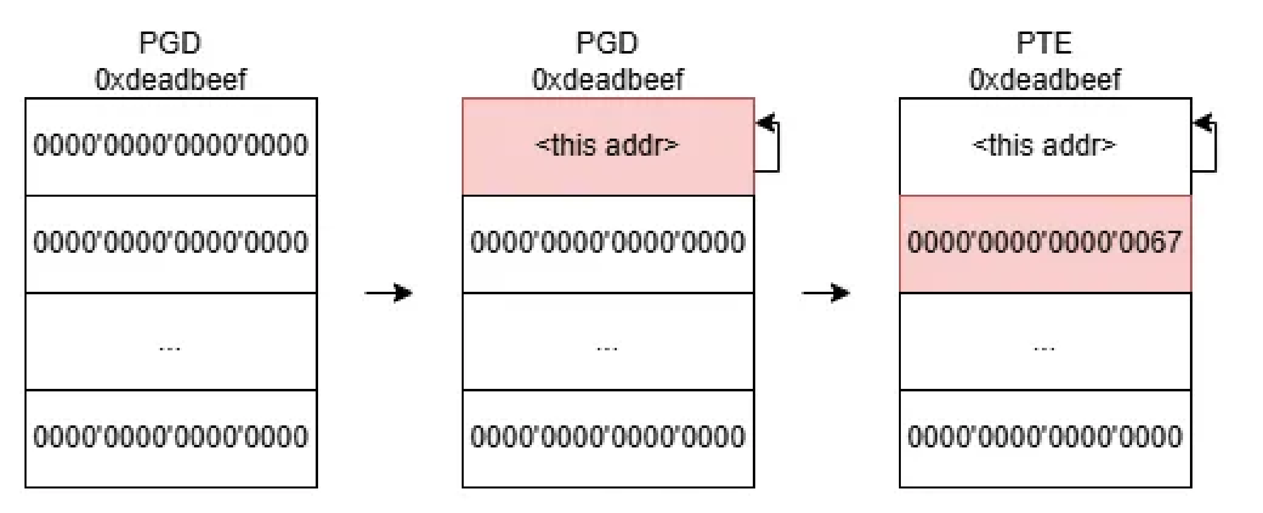
Winner gets an 0day (this is a joke jeremy)

Audience puzzle

- You get:
  - PGD physical address
  - A single arbitrary 8-byte write in the kernel
- How to pwn a Linux device?

Audience puzzle: solution 1/2

1. Set a PGD entry to the PGD address
2. Deref PGD entry as PUD, PMD, PTE, user page
3. Using userpage, forge PTE entries in PGD page
4. Unlimited phys read/write



PGD
0xdeadbeef

0000'0000'0000'0000

0000'0000'0000'0000

...

0000'0000'0000'0000

→

PGD
0xdeadbeef

<this addr>

0000'0000'0000'0000

...

0000'0000'0000'0000

→

PTE
0xdeadbeef

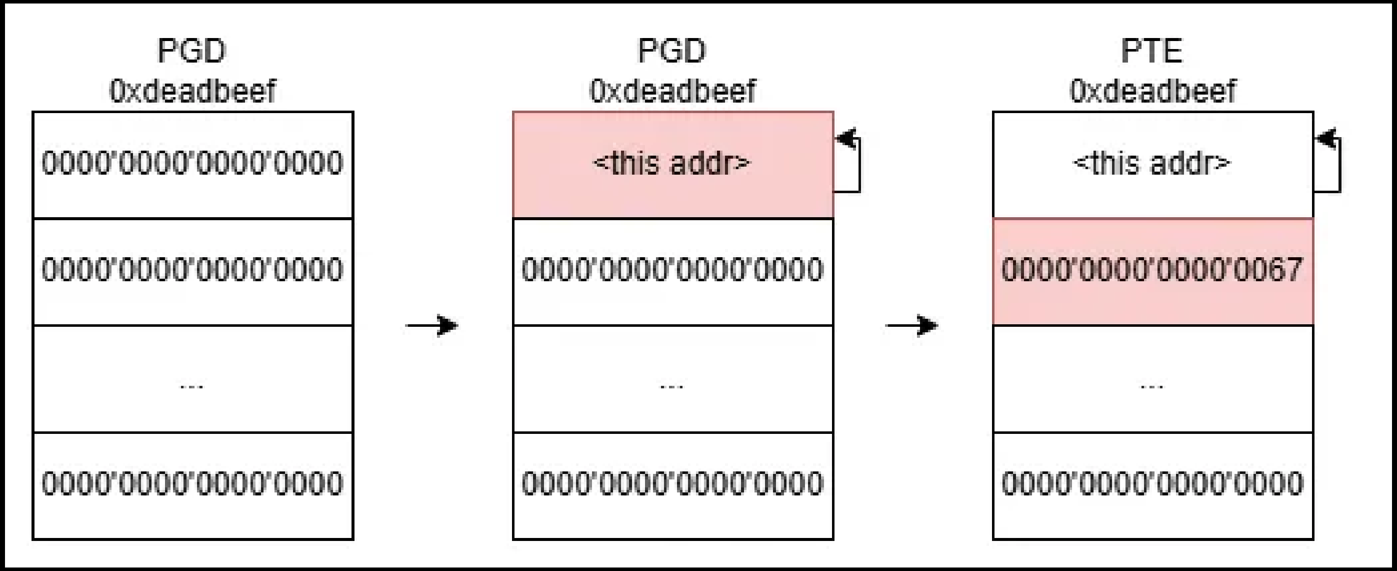
<this addr>

0000'0000'0000'0067

...

0000'0000'0000'0000

Audience puzzle: solution 2/2



PGD
0xdeadbeef

0000'0000'0000'0000
0000'0000'0000'0000
...
0000'0000'0000'0000

→

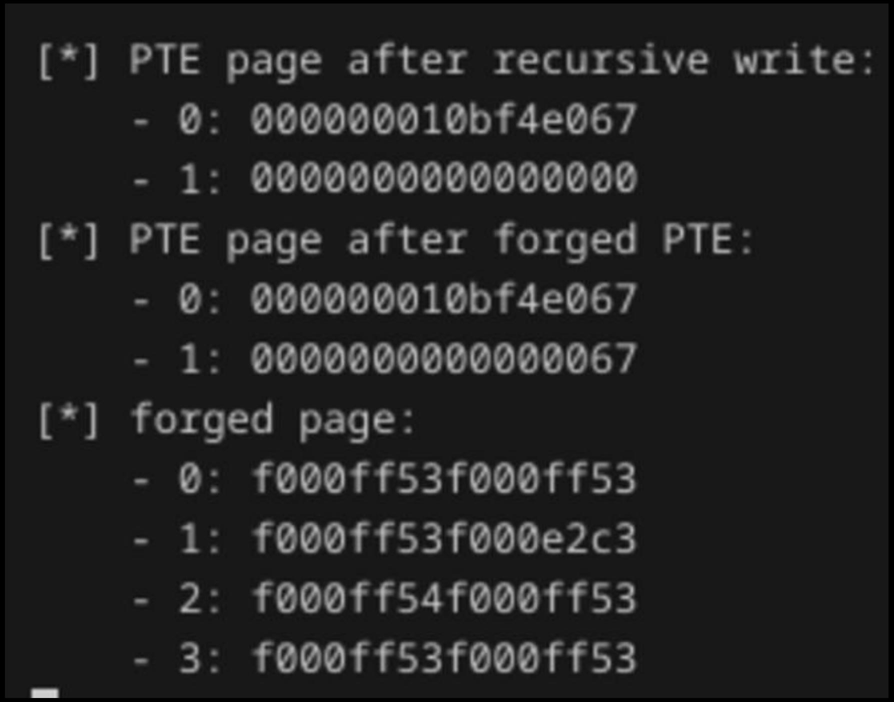
PGD
0xdeadbeef

<this addr>
0000'0000'0000'0000
...
0000'0000'0000'0000

→

PTE
0xdeadbeef

<this addr>
0000'0000'0000'0067
...
0000'0000'0000'0000



```
[*] PTE page after recursive write:
    - 0: 000000010bf4e067
    - 1: 0000000000000000
[*] PTE page after forged PTE:
    - 0: 000000010bf4e067
    - 1: 0000000000000067
[*] forged page:
    - 0: f000ff53f000ff53
    - 1: f000ff53f000e2c3
    - 2: f000ff54f000ff53
    - 3: f000ff53f000ff53
```

```
*(ulong*)0x0 = 0 | 0x067 // rw
printf(“%llx”, *(ulong*)0x0)
```

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