I’m High

”by Nicolas A. Economou”

BFS LABS
BLUE FROST SECURITY

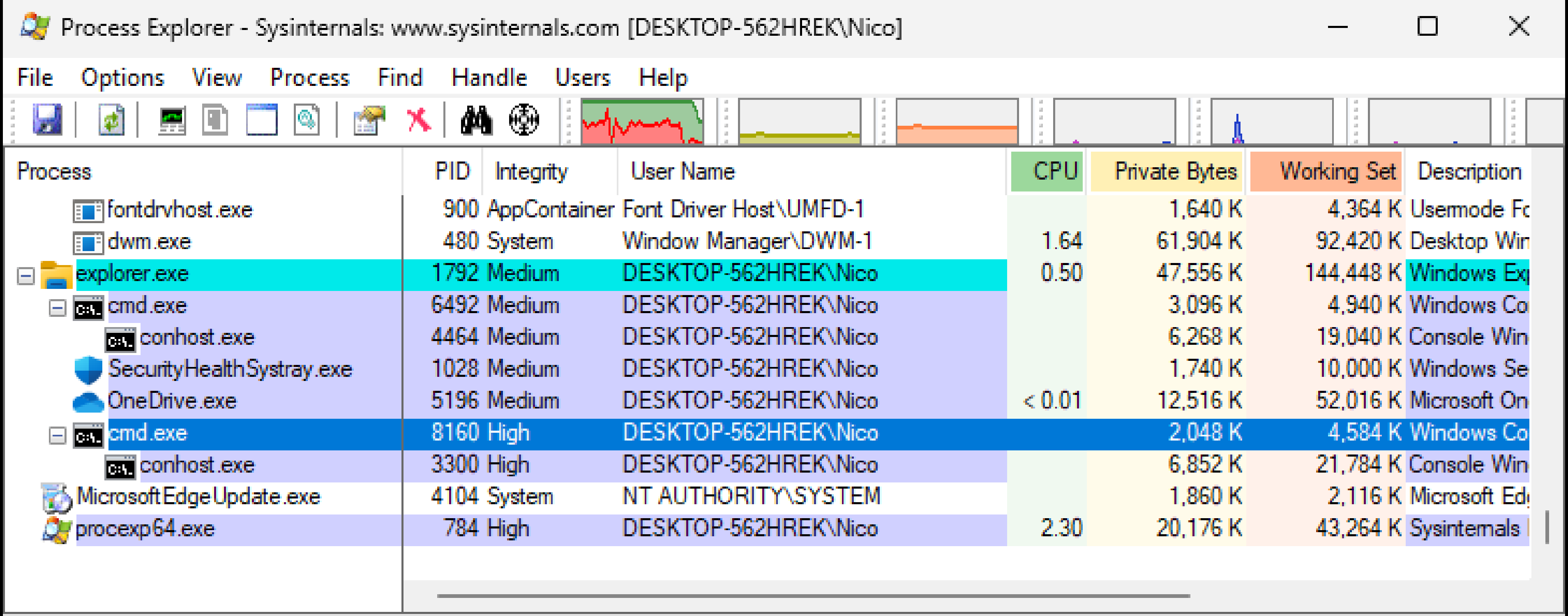
EKOPARTY
SINCE 2001

Who am I?

- Exploit writer & security researcher at *”BFS LABS” (Blue Frost Security)*
- Specialized on Windows exploitation
- Working on security since 2005
- Many talks, advisories, blogposts, tools, etc

Why am I High?

that’s why...



Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-562HREK\Nico]

Process	PID	Integrity	User Name	CPU	Private Bytes	Working Set	Description
fontdrvhost.exe	900	AppContainer	Font Driver Host\UMFD-1		1,640 K	4,364 K	Usermode Fc
dwm.exe	480	System	Window Manager\DWM-1	1.64	61,904 K	92,420 K	Desktop Win
⊟ explorer.exe	1792	Medium	DESKTOP-562HREK\Nico	0.50	47,556 K	144,448 K	Windows Exp
⊟ cmd.exe	6492	Medium	DESKTOP-562HREK\Nico		3,096 K	4,940 K	Windows Co
conhost.exe	4464	Medium	DESKTOP-562HREK\Nico		6,268 K	19,040 K	Console Win
SecurityHealthSystray.exe	1028	Medium	DESKTOP-562HREK\Nico		1,740 K	10,000 K	Windows Se
OneDrive.exe	5196	Medium	DESKTOP-562HREK\Nico	< 0.01	12,516 K	52,016 K	Microsoft On
⊟ cmd.exe	8160	High	DESKTOP-562HREK\Nico		2,048 K	4,584 K	Windows Co
conhost.exe	3300	High	DESKTOP-562HREK\Nico		6,852 K	21,784 K	Console Win
MicrosoftEdgeUpdate.exe	4104	System	NT AUTHORITY\SYSTEM		1,860 K	2,116 K	Microsoft Ed
procexp64.exe	784	High	DESKTOP-562HREK\Nico	2.30	20,176 K	43,264 K	Sysinternals

how it started?

how it started?

- *I was reversing some Windows services (CSRSS.EXE)*
- *Working on manifests files (“.manifest”)*
- *Trying to understand how they work*

how it started?

- *I saw the presentation "The Print Spooler Bug that Wasn't" at 'OffensiveCon 2023'*
- *Given by "James Farshaw" & "Maddie Stone" (Google Project Zero)*
- *Talk about a* ***0-day*** *intercepted in the wild (CVE-2022-41073)*

how it started?

- *The exploit used a manifest file to get execution*
- *The exploitation was done from Medium Integrity Level*
- *The exploit remapped the ’C:’ drive (what???)*

Remapping ‘C:’ drive

Remapping C: drive

- *It consists on changing the base directory of ‘**C:**’*
- *It can be done by using a symbolic link*
- *E.g: “c:” → “c:\users\public”*
  - *New “system32”: “c:\windows\system32” → “c:\users\public\windows\system32”*

Remapping C: drive

- *The function to do that is “DefineDosDevice”*
- *It can remap almost any drive from Medium IL*
- *Except the ones that were previously mapped...*

Remapping C: drive

- *A low level function exists which allows that*
- *The NtCreateSymbolicLinkObject function*
- *It was used by the exploit in the wild!*

Remapping C: drive

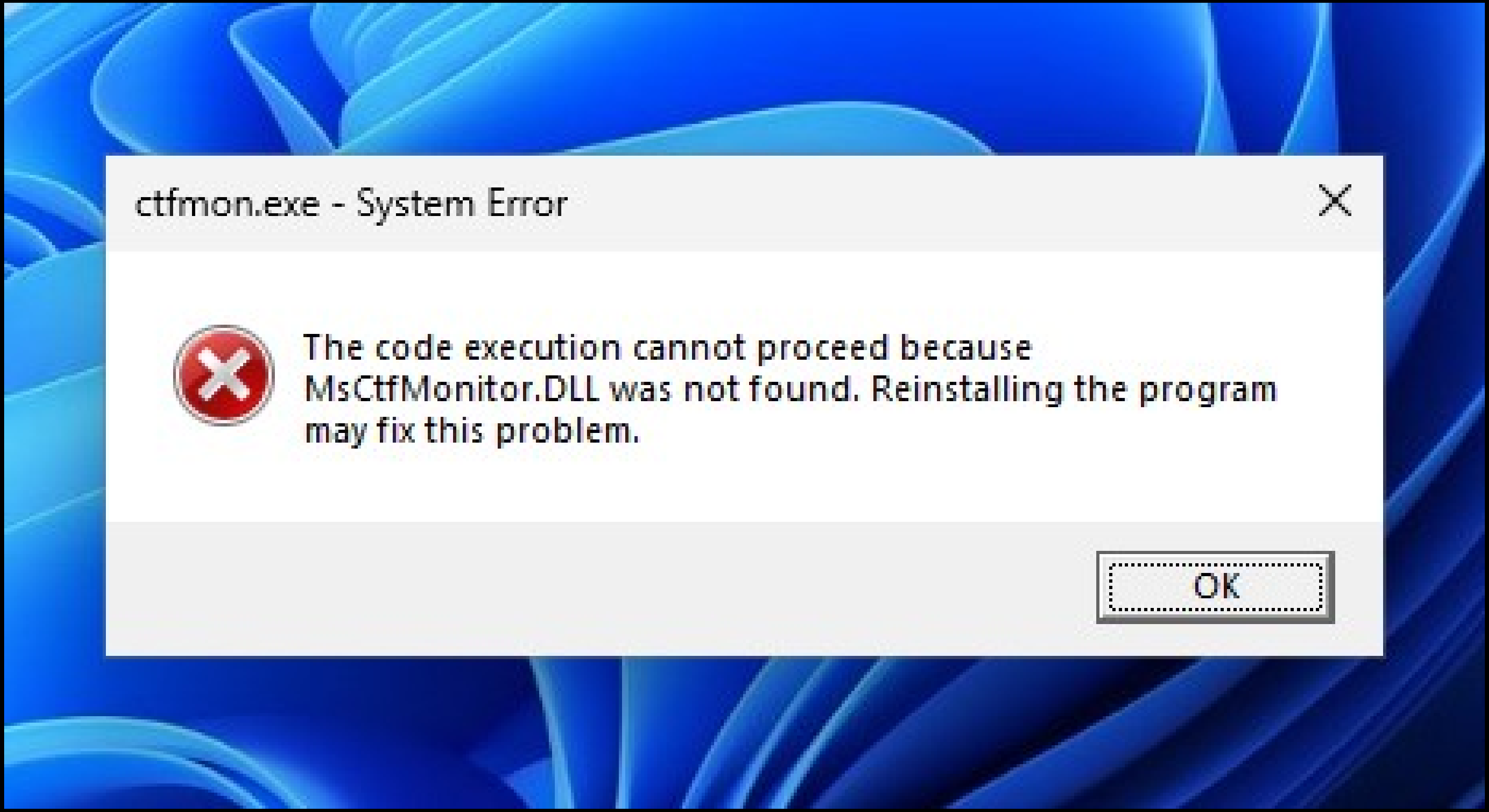
- *It only affects the current user*
- *Services which impersonate the current user are affected*
- *The Windows kernel is affected in some syscalls*

Bug found

Report to MSRC

- *It was reported to Microsoft on August 25th*
- *MSRC Case 81895*
- *Still unfixed (0-day)...*

Error found



ctfmon.exe - System Error

The code execution cannot proceed because MsCtfMonitor.DLL was not found. Reinstalling the program may fix this problem.

At the beginning...

Type of bug

- *It's a DLL hijacking bug*
- *It allows us to inject DLL's in some privileged processes*
- *The user loader is affected*

Bug benefits

- *Used to escalate privileges (aka EoP) from Medium IL*
- *Get code execution in High integrity level (or kind of)*
- *Deterministic exploitation (always works)*

Bug requirements

- *The affected executables have an embedded manifest*
- *Tags required: level="asInvoker" + uiAccess="true"*
- *”<autoElevate>” tag is not required*

Embedded manifest example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">

<assemblyIdentity
    version="5.1.0.0"
    processorArchitecture="amd64"
    name="Test"
    type="win32"
/>

<description>Test description</description>

<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
  <security>
        <requestedPrivileges>
               <requestedExecutionLevel
                    level="asInvoker"
                    uiAccess="true"
               />
        </requestedPrivileges>
    </security>
</trustInfo>
</assembly>
```

Vulnerable versions

- *Vulnerable Windows versions:*

  - *Windows 11 (23H2 - release 25977 – Canary Channel)*

  - *Windows 11 (22H2 and previous)*

  - *Windows 10 (22H2 and previous)*

  - *Windows Servers (not tested, probably vulnerable)*

  - *Windows 8.1 (not tested, probably vulnerable)*

  - *Windows 7 (vulnerable)*

Vulnerable program list

- *Some programs on “Windows 11” 22H2:*

  - ctfmon.exe, EaseOfAccessDialog.exe

  - EoAExperiences.exe, Magnify.exe

  - Narrator.exe, osk.exe

  - psr.exe, rdpinput.exe

  - rdpshell.exe, VoiceAccess.exe

  - msra.exe (it has “AutoElevate” tag)

Root cause

- *Process groups and privileges are identical to regular processes (affected by remapping)*

- *Only Mandatory Label is different (High)*

- *Searchable DLLs are only affected*

Loader path

- *Loader module path:*

→ *...*

→ *ntdll!LdrpInitializeProcess*

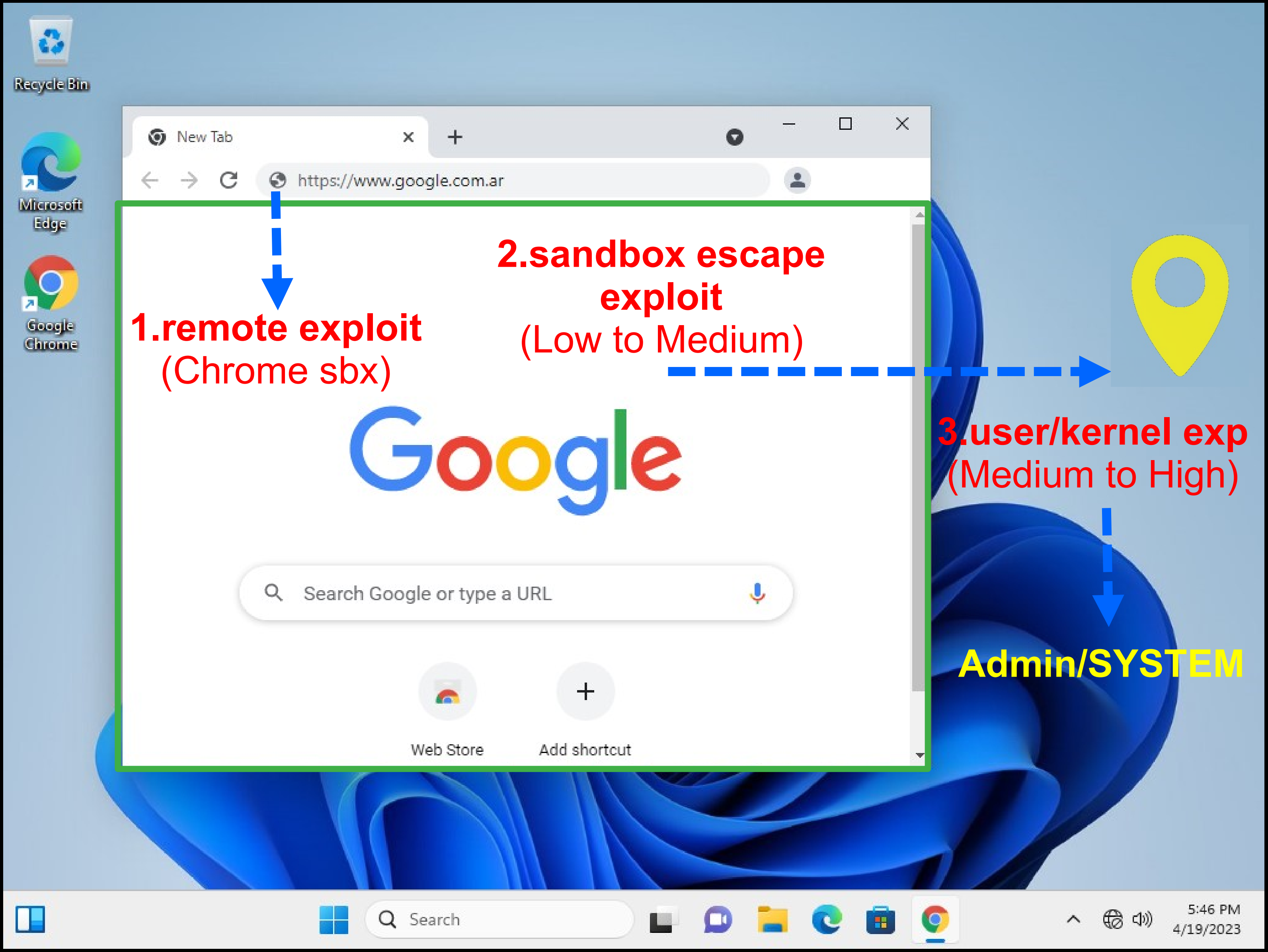
→ *ntdll!LdrpDrainWorkQueue*

→ *ntdll!LdrpProcessWork*

→ *ntdll!LdrpMapDllSearchPath*

→ *ntdll!LdrpMapDllNtFileName*

Real life attack scenario



1.remote exploit (Chrome sbx)

2.sandbox escape exploit (Low to Medium)

3.user/kernel exp (Medium to High)

Admin/SYSTEM

Exploitation
“part 1”

Exploitation – part 1

- *Target process: “ctfmon.exe”*
- *Only a DLL is required ("MsCtfMonitor.dll")*
- *Only one exported function is required ("DoMsCtfMonitor")*

Exploitation – part 1

- *Create fake directory ".\windows\system32"*
- *Copy fake "MsCtfMonitor.dll" there*
- *Hooks "ShellExecute" function to intercept the process creation*

Exploitation – part 1 - steps

- *Execute "ctfmon.exe" via "ShellExecute"*
- *Remap "C:" in the hook (change the system directory when process is still suspended)*
- *Resume the process creation*
- *Code execution is achieved!*

Demo 1

Exploitation
“part 2 – the chain”

Exploitation – part 2

- *”””HIGH””” IL has been obtained*

- *The process still have restrictions (not real “Admin” privileges)*

- *It’s necessary to chain the attack to get full privileges*

Exploitation – part 2

- *The pwned process token has SECURITY_MANDATORY_HIGH_RID (0x3000)*

- *This privilege could be useful to elevate*

- *A possible attack vector is to register an ACTX for a EXE/DLL (aka cache poisoning!)*

Exploitation – part 2

- *At some patch level of “Win11” 22H2, an IL check was added for ACTX registering*

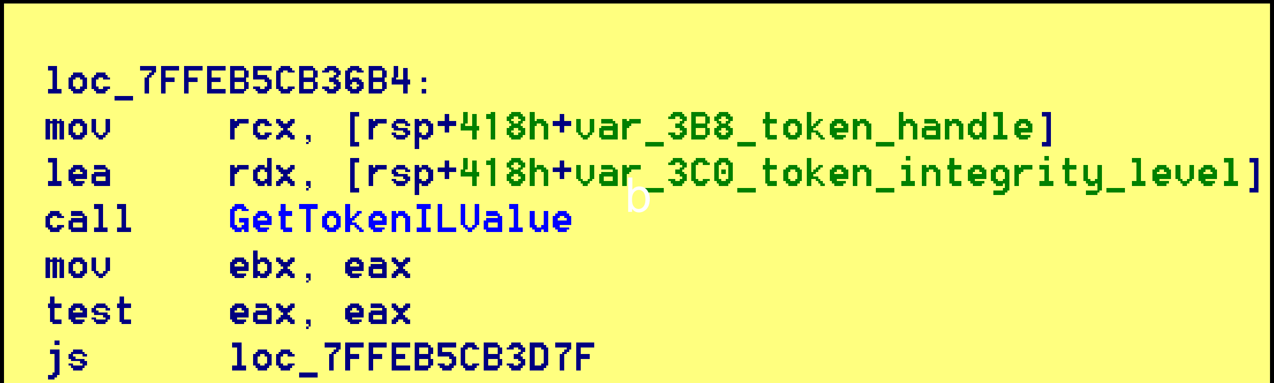
- *It was after a 0-day in the wild (CVEs mentioned on the ZDI blogpost – “Activation Context Cache Poisoning – CSRSS ...”)*

- *Elevated processes only use now ACTXs registered with the same IL*

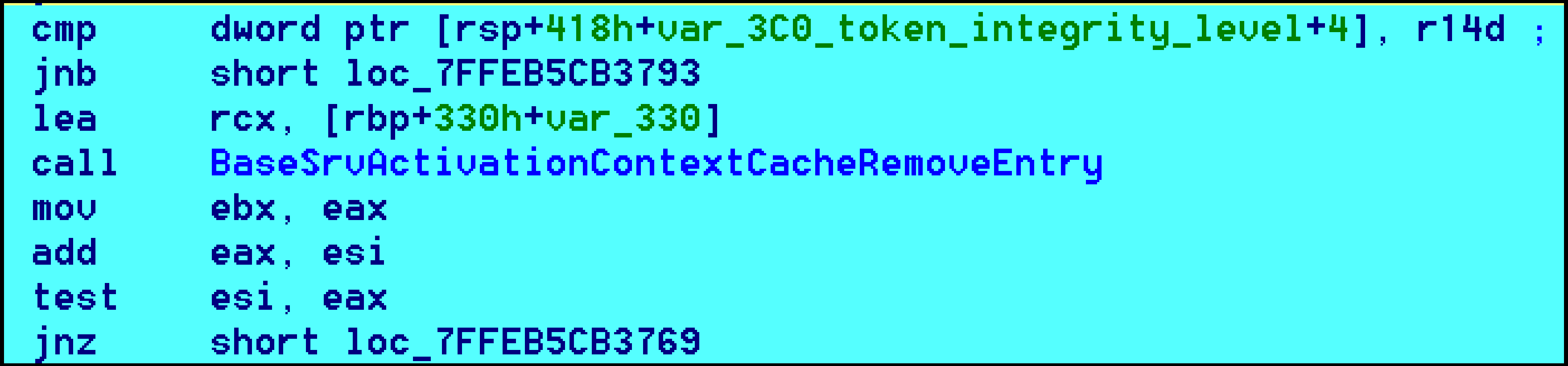
Exploitation – part 2

-sxssrv!BaseSrvSxsCreateActivationContextFromStructEx

(new code)



```
loc_7FFEB5CB36B4:
mov     rcx, [rsp+418h+var_3B8_token_handle]
lea     rdx, [rsp+418h+var_3C0_token_integrity_level]
call    GetTokenILValue
mov     ebx, eax
test    eax, eax
js      loc_7FFEB5CB3D7F
```



```
cmp     dword ptr [rsp+418h+var_3C0_token_integrity_level+4], r14d ;
jnb     short loc_7FFEB5CB3793
lea     rcx, [rbp+330h+var_330]
call    BaseSrvActivationContextCacheRemoveEntry
mov     ebx, eax
add     eax, esi
test    esi, eax
jnz     short loc_7FFEB5CB3769
```

Some RIDs

- *SECURITY_MANDATORY_LOW (0x1000)*
  - *Sandboxed processes (e.g: “Chrome” renderer)*

- *SECURITY_MANDATORY_MEDIUM (0x2000)*
  - *Most programs (e.g: “notepad.exe”, “cmd.exe”, etc)*

Some RIDs

- *SECURITY_MANDATORY_HIGH (0x3000)*
  - *“Run as Administrator”, some privileged programs*

- *SECURITY_MANDATORY_SYSTEM (0x4000)*
  - *Services*

Cache poisoning scenarios

- *E.g 1: if "chrome.exe" (renderer) registers an ACTX for "notepad.exe", it won't be used (0x1000 vs 0x2000)*

- *E.g 2: if "notepad.exe" registers an ACTX for "calc.exe", it'll be used (0x2000 vs 0x2000)*

Cache poisoning scenarios

- *E.g 3: if "notepad.exe" registers an ACTX for "tcmsetup.exe", it won't be used (0x2000 vs 0x3000)*

- *E.g 4: if "ctfmon.exe" registers an ACTX for "tcmsetup.exe", it'll be used (0x3000 vs 0x3000)*

Exploitation – part 2

- *Target process: "tcmsetup.exe" ('Telephony Client Setup Help')*

- *Run as real High IL (Administrator)*

- *Easy to get SYSTEM privileges from it (usually obtained by kernel exploits)*

Exploitation – part 2

- *Target DLL: “tapi32.dll” (register an ACTX)*
- *This DLL has an embedded manifest*
- *DLL hijacked: “imm32.dll” (where code execution is achieved)*

Steps

- *A real "windows\system32" subdirectory is required (for registering the ACTX)*

- *E.g: "c:\windows\system32\tasks" (because it's writable)*

- *Copy custom "tapi32.manifest" and fake "imm32.dll" there*

TAPI32 manifest

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Copyright (c) Microsoft Corporation →

<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
manifestVersion="1.0">
<assemblyIdentity
    version="1.0.0.0"
    name="TAPI32"
    processorArchitecture="amd64"
    type="win32"
    language="tasks"
/>

<file name="imm32.dll"/>
</assembly>
```

Registering ACTXs

- *The “CreateActCtx()” function is used to register ACTXs (better use low level ;-))*

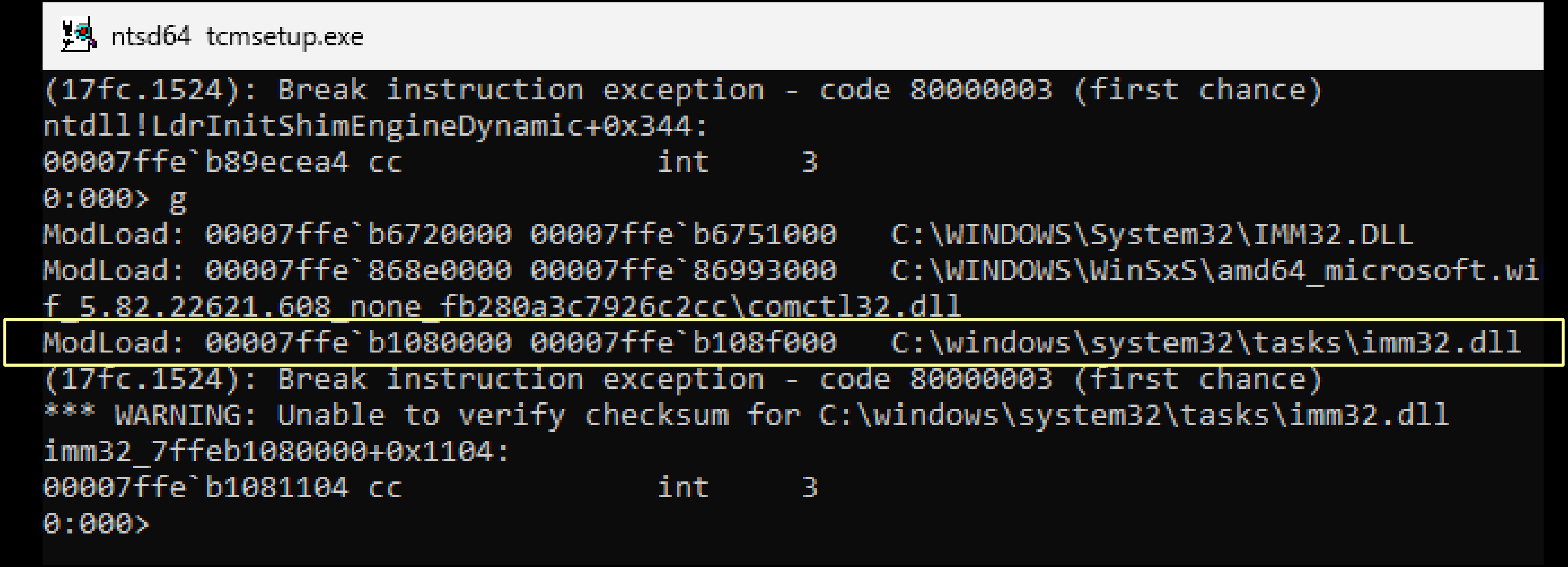
```
”””””””
ACTCTXA actx = {0};
actx.cbSize = sizeof (actx);
actx.lpSource = "test.manifest";

CreateActCtxA (&actx);
”””””””
```

Final steps

- *Register the ACTX for "tapi32.dll" (from "ctfmon.exe")*
- *Execute "tcmsetup.exe"*
- *"tcmsetup.exe" → "tapi32.dll" → "imm32.dll"*
- *Code execution achieved!*

Bug found



```
(17fc.1524): Break instruction exception - code 80000003 (first chance)
ntdll!LdrInitShimEngineDynamic+0x344:
00007ffe`b89ecea4 cc              int     3
0:000> g
ModLoad: 00007ffe`b6720000 00007ffe`b6751000   C:\WINDOWS\System32\IMM32.DLL
ModLoad: 00007ffe`868e0000 00007ffe`86993000   C:\WINDOWS\WinSxS\amd64_microsoft.wi
f_5.82.22621.608_none_fb280a3c7926c2cc\comctl32.dll
ModLoad: 00007ffe`b1080000 00007ffe`b108f000   C:\windows\system32\tasks\imm32.dll
(17fc.1524): Break instruction exception - code 80000003 (first chance)
*** WARNING: Unable to verify checksum for C:\windows\system32\tasks\imm32.dll
imm32_7ffeb1080000+0x1104:
00007ffe`b1081104 cc              int     3
0:000>
```

Demo 2

Bug limitations

Bug limitations

- *If the current user is member of the "Administrators" group (the default one)*
  - *SECURITY_MANDATORY_HIGH_RID (0x3000) is obtained*

- *If the current user is member of the "Users" group*
  - *SECURITY_MANDATORY_MEDIUM_RID+ (0x2010) is obtained*

Conclusions

Conclusions

- *All Windows versions are vulnerable!*
- *It could be thought of like a UAC bypass*
- *System drives shouldn’t be remapped from MIL*

Thanks!

@NicoEconomou

BFS LABS
BLUE FROST SECURITY

EKOPARTY
SINCE 2001