

# Diving into MSSQL madness



**Cyberdefense**

whoami /all



Éditer le profil

**Aurélien Chalot**

@Defte\_



Hacker, sysadmin and security researcher @OrangeCyberdef 🖥️

Calisthenic enthusiast 💪 and wannabe philosopher 📖

🔥 Hide&Sec 🔥




# 1 / How it started ?

 Pennyw0rth / NetExec

<> Code • Issues 61 🔗 Pull requests 60 💬 Discussions ▶ Actions 📁 Projects 🛡 Security 📈 Insights

Implement channel binding check on mssql protocol #713



Open Feature



Signum21 opened on May 31

**Please Describe The Problem To Be Solved**  
It would be useful to have the channel binding check when using `nxc mssql <Target>` like in smb and ldap.

**(Optional): Suggest A Solution**  
An example can be found here <https://github.com/CompassSecurity/mssqlrelay>

  2

<https://github.com/Pennyw0rth/NetExec/issues/713>



cannatag / ldap3

Code



Issues

178



Pull requests



## Authentication #1087

`owers:add_ntlm_channel_binding` on

Contributor ...

idea is to bind the outer secure  
nel (NTLM here). This kind of  
LM authentication.

R [MS-NLMP 2.2.2.1](#) within the  
(ChannelBindings). The Value field  
e sha256 of the server's certificate

ects by [@skelsec](#).



## 2 / First we'll need a setup

TOP DOWNLOADS

# Get started with SQL Server on-premises or in the cloud



## SQL Server 2022 on-premises

Get the performance and security of SQL Server 2022—a scalable, hybrid data platform—now Azure-enabled.

[Download now](#)



## SQL Server on Azure

Run SQL Server on Azure SQL with built-in security and manageability.

[Learn more](#)



## SQL Server 2022 Developer

Get the full-featured free edition, licensed for use as a development and test database in a non-production environment.

[Download now](#)



## SQL Server 2022 Express

Get the free edition, ideal for development and production for desktop, web, and small server applications.

[Download now](#)

SQL Server Configuration Manager (Local)			
SQL Server Services			
SQL Server Network Configuration (32bit)			
SQL Native Client 11.0 Configuration (32bit)			
Azure Extension For SQL Server			
SQL Server Network Configuration			
SQL Native Client 11.0 Configuration			
Azure Extension For SQL Server			
Name	State	Start Mode	
SQL Server (SQLEXPRESS)	Running	Automatic	
SQL Server Agent (SQLEXPRESS)	Stopped	Automatic	
SQL Server Browser	Stopped	Other (Boot, System...)	

**A simple database server running as the Administrator (RID500) domain account because who cares right ?**

SQL Server (SQLEXPRESS) Properties

Always On Availability Groups | Startup Parameters | Advanced

Log On | Service | FILESTREAM

Log on as:

☐ Built-in account:

☐ This account:

Account Name: WHITEFLAG\Administrateur

Password: \*\*\*\*\*

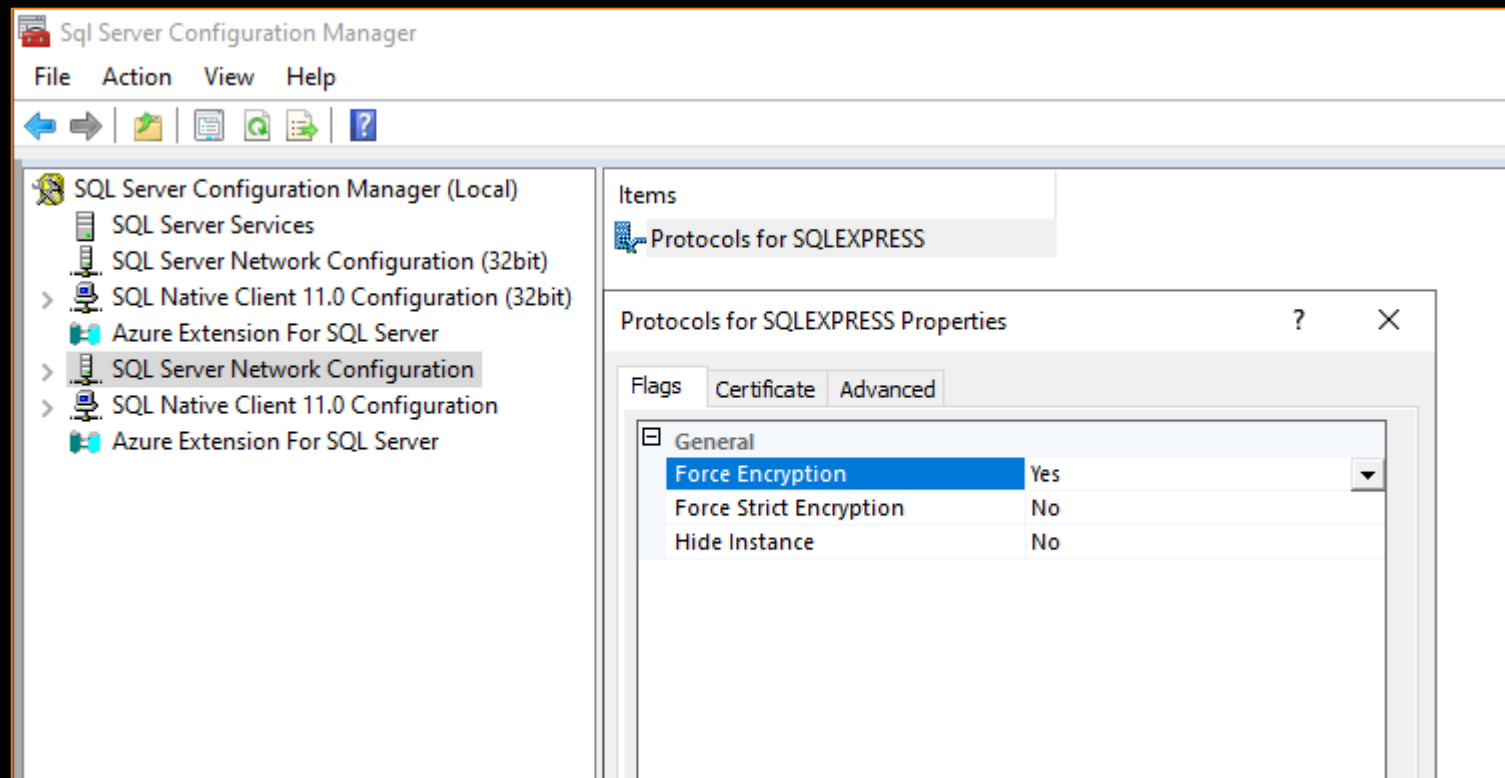
Confirm password: \*\*\*\*\*



## Let's try NXC

```
2:04 ] [ ach@blackpearl:~ ]  
$ nxc mssql 192.168.56.32 -u Administrateur -p Defte@WF  
SSQL      192.168.56.32    1433    SERVER      [*] Windows Server 2022 Build 20348 (name:SERVER) (dom  
SSQL      192.168.56.32    1433    SERVER      [+] whiteflag.local\Administrateur:Defte@WF (Pwn3d!)  
2:09 ] [ ach@blackpearl:~ ]  
$ █
```

## Let's enable TLS over MSSQL



2

```
ug=True, no_p
eur'], passwo
one, kerberos
t_modules=Fal
force_ps32=F
```

```
sql.py
/mssql/databa
```

```
ipv6=False
```

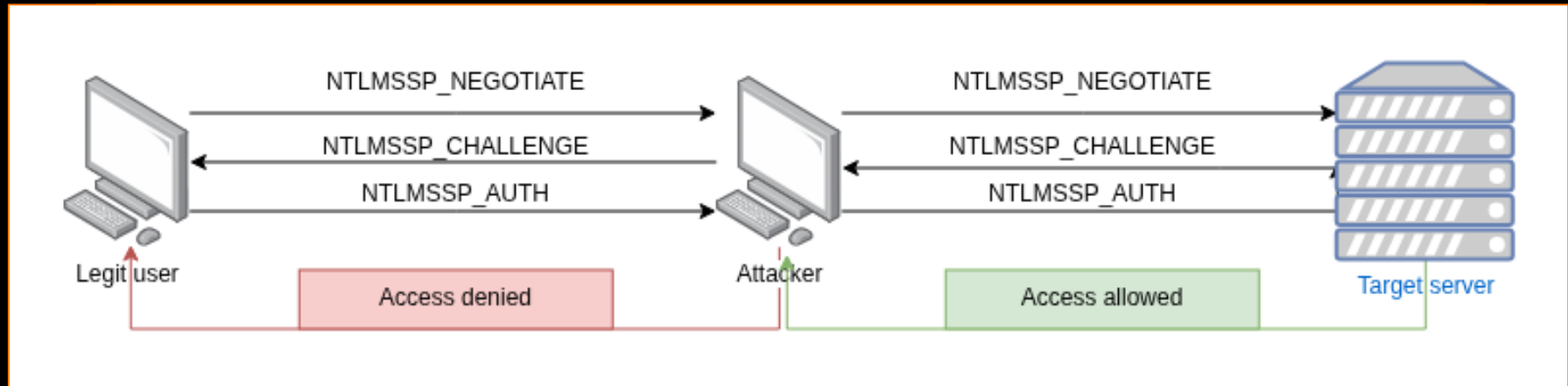
## Hanging.....

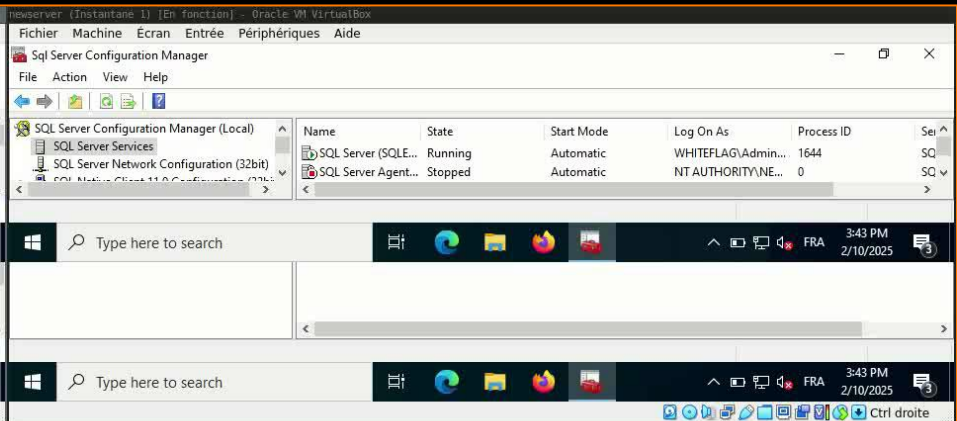
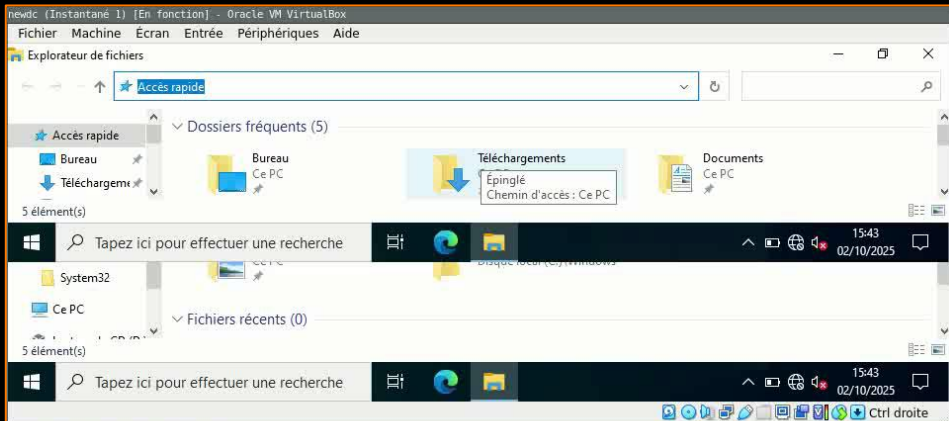


## 3 / Back to defaults

# Default MSSQL configuration is vulnerable to NTLM relay

- NTLMSSP\_NEGOTIATE: Hey man, I'm Deft, can I authenticate to you ?
- NTLMSSP\_CHALLENGE: Sure, can you encrypt that <string> with your password pls ?
- NTLMSSP\_AUTH: Here is the <encrypted\_string>





# The only way to prevent the relay is to enable Channel Binding:

```
[ 4:39 ] [ ach@blackpearl:~ ]
$ sudo ntlmrelayx.py -t mssql://192.168.56.32 -smb2support
Impacket v0.13.0.dev0+20250528.4535.5b338613 - Copyright Fortra, LLC and its affiliated companies

[*] Protocol Client SMB loaded..
> [*] Protocol Client WINRMS loaded..
> [*] Protocol Client SMTP loaded..
> [*] Protocol Client RPC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server on port 445
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server on port 9389
[*] Setting up RAW Server on port 6666
[*] Multirelay disabled

[*] Servers started, waiting for connections
[*] SMBD-Thread-5 (process_request_thread): Received connection from 192.168.56.31, attacking target mssql://192.168.56.32
[-] ERROR(server\SQLEXPRESS): Line 1: Échec de la connexion. La connexion provient d'un domaine non approuvé et ne peut pas
[-] Authenticating against mssql://192.168.56.32 as WHITEFLAG/ADMINISTRATEUR FAILED
[*] All targets processed!
```

# Looking at the logs

```
The Service Broker endpoint is in disabled or stopped state.  
The Database Mirroring endpoint is in disabled or stopped state.  
Service Broker manager has started.  
Recovery is complete. This is an informational message only. No user action is required.  
External governance manager initialized  
Attribute synchronization manager initialized  
Erreur : 17806, Gravité : 20, État : 46.  
SSPI handshake failed with error code 0x80090346, state 46 while establishing a connection with integrated security;  
the connection has been closed.  
Reason: Les liaisons de canaux de ce client sont manquantes ou ne correspondent pas au canal TLS (Transport Layer Security) établi.  
Il est possible que le service fasse l'objet d'une attaque, ou que le fournisseur de données ou système d'exploitation client nécessite  
une mise à niveau pour prendre en charge la protection étendue.  
Fermeture de la connexion. Les liaisons de chaînes SSPI fournies au client étaient incorrectes. [CLIENT : 192.168.56.1]  
Erreur : 18452, Gravité : 14, État : 1.  
Login failed. The login is from an untrusted domain and cannot be used with Integrated authentication. [CLIENT : 192.168.56.1]  
Erreur : 17806, Gravité : 20, État : 46.
```

The binding token for that client is missing or does not match the established TLS link.  
**It is possible that the MSSQL service is under attack or that the client does not support extended protection.**  
Closing the connection. The SSPI binding token sent by the client does not match.

**YOU DON'T SAY?**

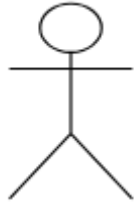






## 4 / About TDS and STARTTLS

# About TDS workflow



User



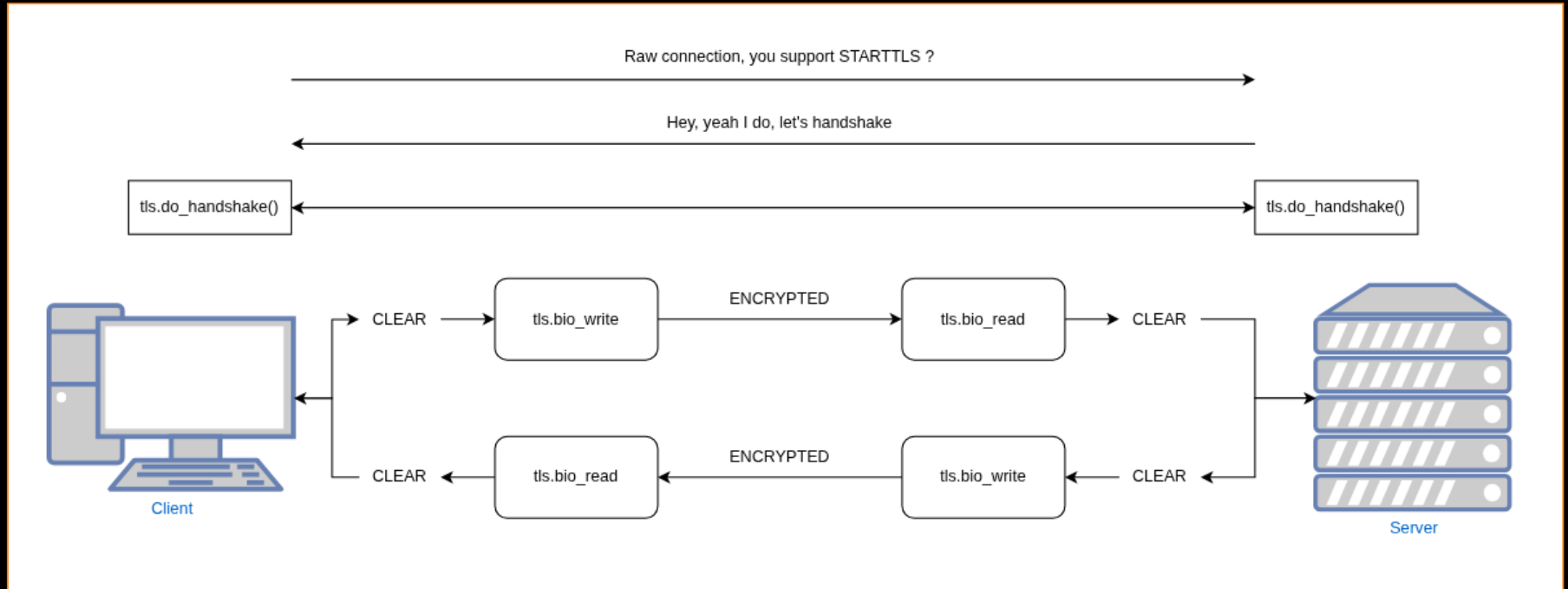
MSSQL database

```
class TDS_LOGIN(Structure):
    structure = (
        ('Length', '<L=0'),
        ('TDSVersion', '>L=0x71'),
        ('PacketSize', '<L=32764'),
        ('ClientProgVer', '>L=7'),
        ('ClientPID', '<L=0'),
        ('ConnectionID', '<L=0'),
        ('OptionFlags1', '<B=0xe0'),
        ('OptionFlags2', '<B'),
        ('TypeFlags', '<B=0'),
        ('OptionFlags3', '<B=0'),
        ('ClientTimeZone', '<L=0'),
        ('ClientLCID', '<L=0'),
        ('HostNameOffset', '<H'),
        ('HostNameLength', '<H=len(self["HostName"])/2'),
        ('UserNameOffset', '<H=0'),
        ('UserNameLength', '<H=len(self["UserName"])/2'),
        ('PasswordOffset', '<H=0'),
        ('PasswordLength', '<H=len(self["Password"])/2'),
        ('AppNameOffset', '<H'),
        ('AppNameLength', '<H=len(self["AppName"])/2'),
        ('ServerNameOffset', '<H'),
        ('ServerNameLength', '<H=len(self["ServerName"])/2'),
        ('UnusedOffset', '<H=0'),
        ('UnusedLength', '<H=0'),
        ('CltIntNameOffset', '<H'),
        ('CltIntNameLength', '<H=len(self["CltIntName"])/2'),
        ('LanguageOffset', '<H=0'),
        ('LanguageLength', '<H=0'),
        ('DatabaseOffset', '<H=0'),
        ('DatabaseLength', '<H=len(self["Database"])/2'),
        ('ClientID', '0s=b"\x01\x02\x03\x04\x05\x06"'),
        ('SSPIOffset', '<H'),
        ('SSPILength', '<H=len(self["SSPI"])'),
        ('AtchDBFileOffset', '<H'),
        ('AtchDBFileLength', '<H=len(self["AtchDBFile"])/2'),
        ('HostName', ':'),
        ('UserName', ':'),
        ('Password', ':'),
        ('AppName', ':'),
        ('ServerName', ':'),
        ('CltIntName', ':'),
        ('Database', ':'),
        ('SSPI', ':'),
        ('AtchDBFile', ':'),
    )
```

# Authentication workflow and STARTTLS

```
7 def login(self, database, username, password='', domain='', hashes = None, useWindowsAuth = False):
8     if hashes is not None:
9         lmhash, nthash = hashes.split(':')
10        lmhash = binascii.a2b_hex(lmhash)
11        nthash = binascii.a2b_hex(nthash)
12    else:
13        lmhash = ''
14        nthash = ''
15
16    resp = self.preLogin()
17    # Test this!
18    if resp['Encryption'] == TDS_ENCRYPT_REQ or resp['Encryption'] == TDS_ENCRYPT_OFF:
19        LOG.info("Encryption required, switching to TLS")
20
21        # Switching to TLS now
22        ctx = SSL.Context(SSL.TLS_METHOD)
23        ctx.set_cipher_list('ALL:@SECLEVEL=0'.encode('utf-8'))
24        tls = SSL.Connection(ctx, None)
25        tls.set_connect_state()
26        while True:
27            try:
28                tls.do_handshake()
29            except SSL.WantReadError:
30                data = tls.bio_read(4096)
31                self.sendTDS(TDS_PRE_LOGIN, data, 0)
32                tds = self.recvTDS()
33                tls.bio_write(tds['Data'])
34            else:
35                break
```

# STARTTLS schema



## LDAP3 channel binding code

```
if self.channel_binding == TLS_CHANNEL_BINDING:
    # To perform channel binding during NTLM authentication, we need to add a new AV_PAIR (MS-NLMP 2.2.2.1)
    # within the AUTHENTICATE_MESSAGE (MS-NLMP 2.2.1.3). This new AV_PAIR has AvId 0x000A (MsvAvChannelBindings).
    # The Value field contains an MD5 hash of a gss_channel_bindings_struct.
    # The logic here is heavily inspired by "msldap", "minikerberos" and "asyssocks" projects by @skelsec.
    from hashlib import sha256, md5
    ntlm_client.tls_channel_binding = True
    peer_certificate_sha256 = sha256(self.server.tls.peer_certificate).digest()
    # https://datatracker.ietf.org/doc/html/rfc2744#section-3.11
    channel_binding_struct = bytes()
    initiator_address = b'\x00'*8
    acceptor_address = b'\x00'*8
    # https://datatracker.ietf.org/doc/html/rfc5929#section-4
    application_data_raw = b'tls-server-end-point:' + peer_certificate_sha256
    len_application_data = len(application_data_raw).to_bytes(4, byteorder='little', signed = False)
    application_data = len_application_data
    application_data += application_data_raw
    channel_binding_struct += initiator_address
    channel_binding_struct += acceptor_address
    channel_binding_struct += application_data
    # https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-nlmp/83f5e789-660d-4781-8491-5f8c6641f75e
    # "The Value field contains an MD5 hash of a gss_channel_bindings_struct"
    ntlm_client.client_av_channel_bindings = md5(channel_binding_struct).digest()
```

## Copy paste and...

```
[ 11:55 ] [ ach@blackpearl:~ ]
```

```
$ mssqlclient.py WHITEFL
```

```
/home/ach/.local/pipx/ven
```

```
kg_resources is deprecate
```

```
kg_resources package is s
```

```
n to Setuptools<81.
```

```
import pkg_resources
```

```
Impacket v0.13.0.dev0+202
```

```
[*] Encryption required,
```

```
[-] ERROR(adcs): Line 1:
```

```
ut pas être utilisée avec
```

```
[ 11:55 ] [ ach@blackpearl:~ ]
```

```
$
```



Defte 5  
hours into  
the project

```
-windows-auth
```

```
on.py:12: UserWarning: p
```

```
pkg_resources.html. The p
```

```
using this package or pi
```

```
affiliated companies
```


```
line non approuvé et ne pe
```

## Looking at the authentication PCAP...

No.	Time	Source	Destination	Protocol	Length	Info
16	4.911001	192.168.56.11	192.168.56.12	TLSv1.2	775	Application Data
17	4.911100	192.168.56.12	192.168.56.11	TCP	54	1433 → 32816 [ACK] Seq=1294 Ack=2904 Win=2097920 Len=0
18	4.915733	192.168.56.12	192.168.56.11	TLSv1.2	315	Application Data
19	4.920803	192.168.56.12	192.168.56.11	TLSv1.2	603	Application Data
20	4.921191	192.168.56.11	192.168.56.12	TCP	60	32816 → 1433 [ACK] Seq=2904 Ack=2104 Win=262656 Len=0

Frame 18: 315 bytes on wire (2520 bytes captured) on interface 0: Ethernet II, Src: PCSSystemtec..., Dst: 192.168.56.11, Internet Protocol Version 4, Src: 192.168.56.12, Destination: 192.168.56.11, Transmission Control Protocol, Src Port: 49157, Destination Port: 49157, Transport Layer Security, Version: TLS 1.2 (0x0303), Content Type: Application Data, Length: 256, Encrypted Application Data



Defte 10  
hours into  
the project

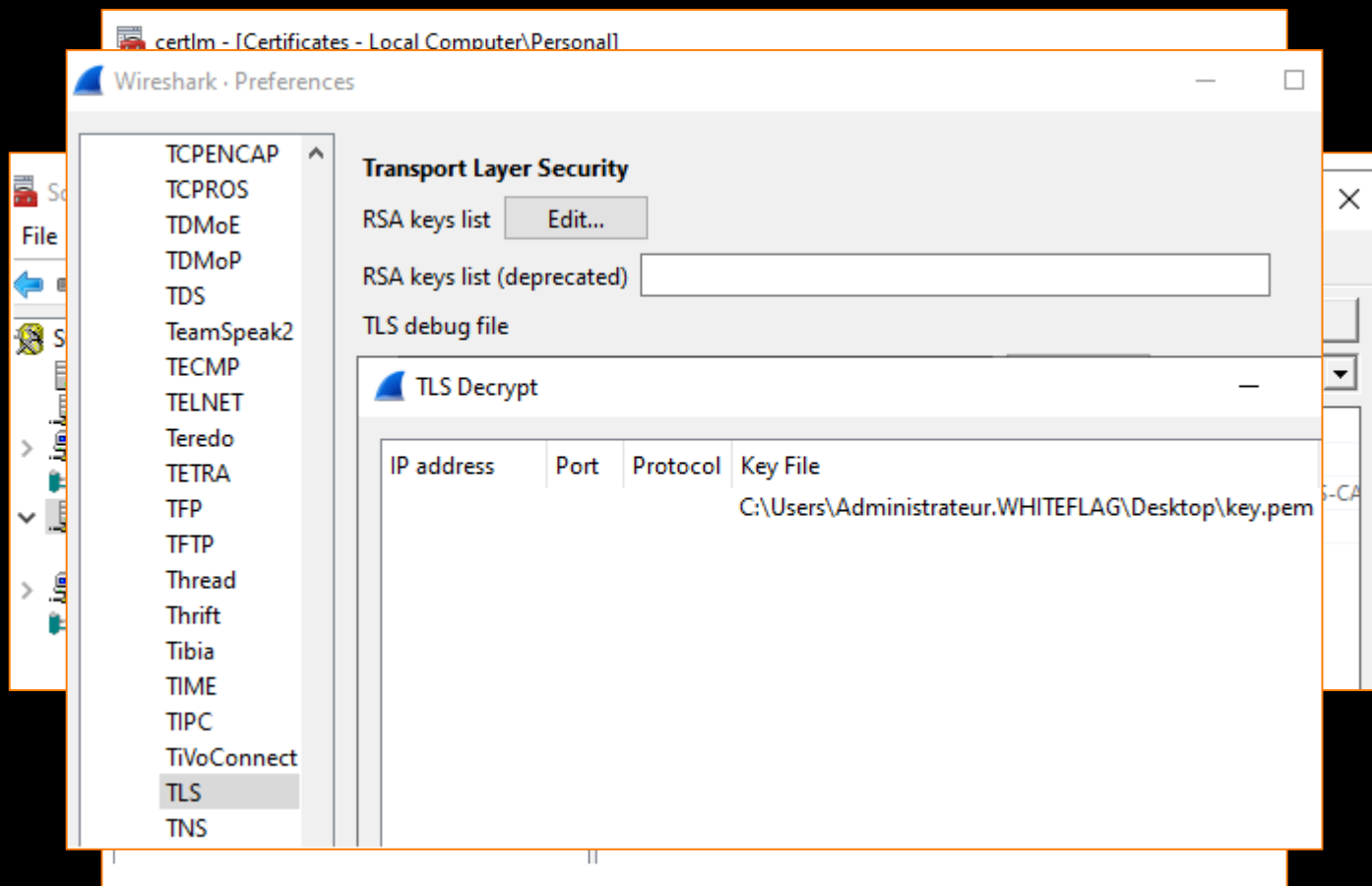
Offset	Hex	ASCII
00d0	35 e8 83 83 5b bf 60 79	6f 44 37 5f 58 21 4a 32
00e0	55 0c 7b 1a ee 75 b7 b2	62 d8 d8 93 41 ce 3f cd
00f0	d0 1c be de 67 71 ef a3	7a 28 6f 44 da 57 0e 59
0100	fa 53 c6 96 c3 19 ef 68	13 f1 ec a1 6b 87 d4 52
0110	9d 7e c8 68 48 ec 74 49	55 dc 75 8a f2 d4 02 69
0120	c8 95 05 fd 3b 04 94 42	74 5b ab a8 29 d2 02 aa
0130	06 91 d9 22 94 36 cd e5	18 10 19



## 5 / Downgrading TLS



## First we need a certificate and its private key



# Then we need to downgrade TLS standards

The screenshot shows the Local Group Policy Editor window. The left pane displays the tree structure under 'Local Computer Policy' > 'Computer Configuration' > 'Administrative Templates' > 'Network' > 'SSL Configuration Settings'. The right pane shows the 'SSL Cipher Suite Order' policy setting. A table at the top lists the current state of the policy:

Setting	State	Comment
SSL Cipher Suite Order	Not configured	No
ECC Curve Order	Not configured	No

Below the table, the 'Description' section states: 'This policy setting determines the cipher suites used by the Secure Socket Layer (SSL).'. The 'Requirements' section states: 'At least Windows Vista'. The 'Options' section shows 'Not Configured' selected. The 'Supported on' section states: 'At least Windows Vista'. The 'Help' section provides a link for all the cipher suites: <http://go.microsoft.com/fwlink/?Linkid=517265>.

Overlaid on the screenshot is a red text box containing the following text:


**TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256, TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256**

**RSA is mandatory because anything else uses Elliptic-Curves which is a pain to intercept1**

# And finally....

No.	Time	Source	Destination	Protocol	Length	Info
4	4.513243	192.168.56.11	192.168.56.1	TCP	74	1433 → 52762 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
5	4.513424	192.168.56.1	192.168.56.11	TCP	66	52762 → 1433 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval
6	4.513628	192.168.56.1	192.168.56.11	TDS	118	TDS7 pre-login message
7	4.513731	192.168.56.11	192.168.56.1	TDS	103	Response
8	4.513938	192.168.56.1	192.168.56.11	TCP	66	52762 → 1433 [ACK] Seq=53 Ack=38 Win=64256 Len=0 TSv
9	4.515458	192.168.56.1	192.168.56.11	TLSv1.2	505	Client Hello
10	4.515647	192.168.56.11	192.168.56.1	TLSv1.2	1538	Server Hello, Certificate, Server Hello Done
11	4.515898	192.168.56.1	192.168.56.11	TCP	66	52762 → 1433 [ACK] Seq=492 Ack=1510 Win=67072 Len=0
12	4.516318	192.168.56.1	192.168.56.11	TLSv1.2	432	Client Key Exchange, Change Cipher Spec, Finished
13	4.51763					
14	4.51889					
15	4.51909					
16	4.52218					
17	4.52264					
18	4.52308					

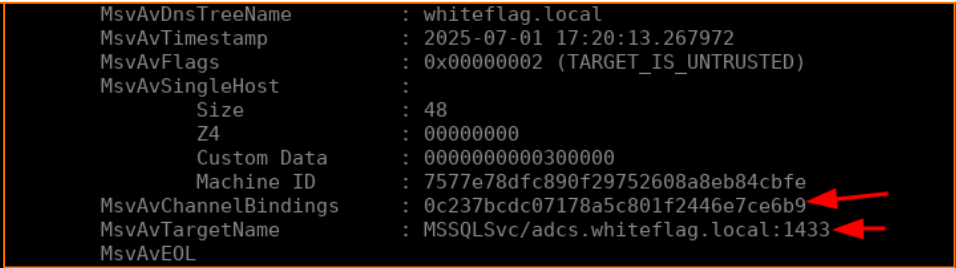
Frame 16: 535	Ethernet II, S	Internet Prot	Transmission	Transport Lay	Data (402 byt
					
<h2>Defte 1 week into the project</h2>					

Offset	Hex	ASCII
0100	28 00 61 00 64 00 63 00 73 00 2e 00 77 00 68 00	( a d c s . w h
0110	69 00 74 00 65 00 66 00 6c 00 61 00 67 00 2e 00	i t e f l a g .
0120	6c 00 6f 00 63 00 61 00 6c 00 05 00 1e 00 77 00	l o c a l . . . w
0130	68 00 69 00 74 00 65 00 66 00 6c 00 61 00 67 00	h i t e f l a g
0140	2e 00 6c 00 6f 00 63 00 61 00 6c 00 07 00 08 00	. l o c a l . . .
0150	7e 8a f1 3a 03 d7 db 01 09 00 32 00 63 00 69 00	~ : . . . . 2 c i
0160	66 00 73 00 2f 00 61 00 64 00 63 00 73 00 2e 00	f s / a d c s .
0170	77 00 68 00 69 00 74 00 65 00 66 00 6c 00 61 00	w h i t e f l a
0180	67 00 2e 00 6c 00 6f 00 63 00 61 00 6c 00 00 00	g . l o c a l . .
0190	00 00	



## 6 / Implementing CBT

[illegible]

# The fuck ??

```
rs from NTLMv2 response:
MsvAvNbDomainName      : WHITEFLAG
MsvAvNbComputerName    : ADCS
MsvAvDnsDomainName     : whiteflag.local
MsvAvDnsComputerName   : adcs.whiteflag.local
MsvAvDnsTreeName       : whiteflag.local
MsvAvTimestamp         : 2025-06-25 15:25:03.887978
MsvAvFlags              : 0x00000002 (TARGET_IS_UNTRUSTED)
MsvAvSingleHost        :
  Size                  : 48
  Z4                    : 00000000
  Custom Data           : 0000000000300000
  Machine ID            : fd97e010703133b6a0a9e9042bf74eed
MsvAvChannelBindings    : 0d0aeb3d968741e55aa2529829bc5a10
MsvAvTargetName         : MSSQLSvc/adcs.whiteflag.local:1433
MsvAvEOL
```

```
rs from NTLMv2 response:
MsvAvNbDomainName      : WHITEFLAG
MsvAvNbComputerName    : ADCS
MsvAvDnsDomainName     : whiteflag.local
MsvAvDnsComputerName   : adcs.whiteflag.local
MsvAvDnsTreeName       : whiteflag.local
MsvAvTimestamp         : 2025-06-25 15:26:25.035212
MsvAvFlags              : 0x00000002 (TARGET_IS_UNTRUSTED)
MsvAvSingleHost        :
  Size                  : 48
  Z4                    : 00000000
  Custom Data           : 0000000000300000
  Machine ID            : fd97e010703133b6a0a9e9042bf74eed
MsvAvChannelBindings    : 61195c99f19cfae2647e719a15d40cf3
MsvAvTargetName         : MSSQLSvc/adcs.whiteflag.local:1433
MsvAvEOL
```



Did you know there are multiple ways you can compute a Channel Binding Token ?

# TLS Server End Point (the one used by LDAPS)

99	201.493288	192.168.56.11	192.168.56.1	TLSv1.2	1538 Server Hello, Certificate	Server Hello Done
100	201.493450	192.168.56.1	192.168.56.11	TCP	66 48488 + 1433 [ACK] Seq=191 Ack=1510 Win=67072 Len=0 TSval=4235207394 TSecr=1882519	
<						
>	Frame 10: 1538 bytes on wire (12304 bits), 1538 bytes captured (12304 bits) on interface \Device\NPF_{0...}					
>	Ethernet II, Src: PCSSystemtec_b7:2a:9a (08:00:27:b7:2a:9a), Dst: 0a:00:27:00:00:00 (0a:00:27:00:00:00)					
>	Internet Protocol Version 4, Src: 192.168.56.11, Dst: 192.168.56.1					
>	Transmission Control Protocol, Src Port: 1433, Dst Port: 35032, Seq: 38, Ack: 191, Len: 1472					
>	Tabular Data Stream					
>	Type: TD57 pre-login message (18)					
>	Status: 0x01, End of message					
>	Length: 1472					
>	Channel: 0					
>	Packet Number: 0					
>	Window: 0					
>	Pre-Login Message					
>	Transport Layer Security					
>	TLSv1.2 Record Layer: Handshake Protocol: Multiple Handshake Messages					
>	Content Type: Handshake (22)					
>	Version: TLS 1.2 (0x0303)					
>	Length: 1459					
>	Handshake Protocol: Server Hello					
>	Handshake Protocol: Certificate					
>	Handshake Type: Certificate (11)					
>	Length: 1366					
>	Certificates Length: 1363					
>	Certificates (1363 bytes)					
>	Certificate Length: 1360					
>	Certificate [-]: 3082054c30820434a00302010202134a0000000a6bfd89db5b388a0e0000000000a300d					
>	Handshake Protocol: Server Hello Done					



# TLS Unique (spoiler: the one used by MSSQL)

No.	Time	Source	Destination	Protocol	Length	Info
16	7.716138	192.168.56.1	192.168.56.11	TLSv1.2	204	Client Hello
17	7.716308	192.168.56.11	192.168.56.1	TLSv1.2	1538	Server Hello, Certificate, Server Hello Done
19	7.717434	192.168.56.1	192.168.56.11	TLSv1.2	432	Client Key Exchange, Change Cipher Spec, Finished
20	7.718322	192.168.56.11	192.168.56.1	TLSv1.2	165	Change Cipher Spec, Finished

> Frame 20: 165 bytes on wire (1320 bits), 165 bytes captured (1320 bits) on interface \Device\NPF- > Ethernet II, Src: PCSSystemtec_b7:2a:9a (08:00:27:b7:2a:9a), Dst: 0a:00:27:00:00:00 (0a:00:27:00:00:00) > Internet Protocol Version 4, Src: 192.168.56.11, Dst: 192.168.56.1 > Transmission Control Protocol, Src Port: 1433, Dst Port: 43574, Seq: 1510, Ack: 557, Len: 99 ▼ Tabular Data Stream Type: TDS7 pre-login message (18) > Status: 0x01, End of message Length: 99 Channel: 0 Packet Number: 0 Window: 0 Pre-Login Message ▼ Transport Layer Security > TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec ▼ TLSv1.2 Record Layer: Handshake Protocol: Finished Content Type: Handshake (22) Version: TLS 1.2 (0x0303) Length: 80 ▼ Handshake Protocol: Finished Handshake Type: Finished (20) Length: 12 Verify Data	0000 0a 00 27 00 00 00 08 00 27 b7 2a 9a 08 00 45 00 ..'. .... '*...E. 0010 00 97 53 eb 40 00 80 06 00 00 c0 a8 38 0b c0 a8 ..S.@... ..B... 0020 38 01 05 99 aa 36 f8 d9 fe c4 12 1e ca 72 80 18 8....6... ..r... 0030 20 01 f1 e6 00 00 01 01 08 0a 00 16 81 61 5b 38 .....a[8 0040 70 a3 12 01 00 63 00 00 00 00 14 03 03 00 01 01 p...c... .. 0050 16 03 03 00 50 25 68 21 e8 ec dd e8 fd cb 88 10 ....Pkh! ..... 0060 6d e8 da 23 94 f7 09 eb 81 78 5a 07 90 e0 e4 8d m...#....xZ.... 0070 81 54 d4 2f a9 46 24 ec dd 18 25 9f 9c 06 4f 2d .T./f\$. .%..O.. 0080 0a 21 81 3e 13 78 d2 3e e6 e7 d7 da fb 9d 50 47 .!>.>x> .....PG 0090 04 12 86 fc 20 c4 19 9b df 04 e2 85 10 b1 f6 ba .... .. 00a0 10 2a 0b 85 39 ..*..g
--	--



# TLS Exporter

tls exporter computation



Demander Tous Images Actualités Vidéos Recherche personnelle

The computation of exported keying material in TLS is defined by RFC 5705 for TLS 1.2 and earlier, and by RFC 8446 for TLS 1.3. For TLS 1.2, the exporter computes a pseudorandom bit string using the TLS Pseudorandom Function (PRF) based on the master secret, a disambiguating label, and optionally a context value. The formula is:

$$\text{PRF}(\text{SecurityParameters.master\_secret}, \text{label}, \text{SecurityParameters.client\_random} + \text{SecurityParameters.server\_random} + \text{context\_value\_length} + \text{context\_value})[\text{length}]$$

If no context is provided, the computation simplifies to:

$$\text{PRF}(\text{SecurityParameters.master\_secret}, \text{label}, \text{SecurityParameters.client\_random} + \text{SecurityParameters.server\_random})[\text{length}]$$

For TLS 1.3, the PRF is replaced with HKDF, and the exporter computation is defined as:

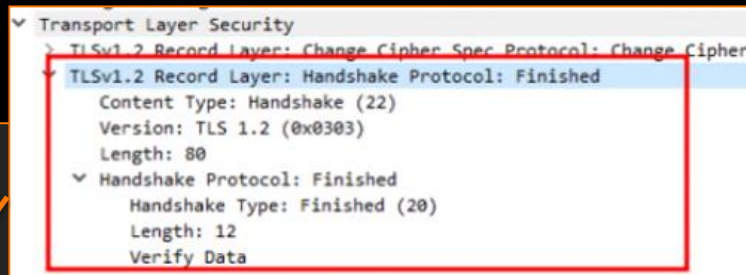
$$\text{TLS-Exporter}(\text{label}, \text{context\_value}, \text{key\_length}) = \text{HKDF-Expand-Label}(\text{Derive-Secret}(\text{Secret}, \text{label}, ""), \text{"exporter"}, \text{Hash}(\text{context\_value}), \text{key\_length})$$


**And MSSQL does not support it yet so....**



# Final computing

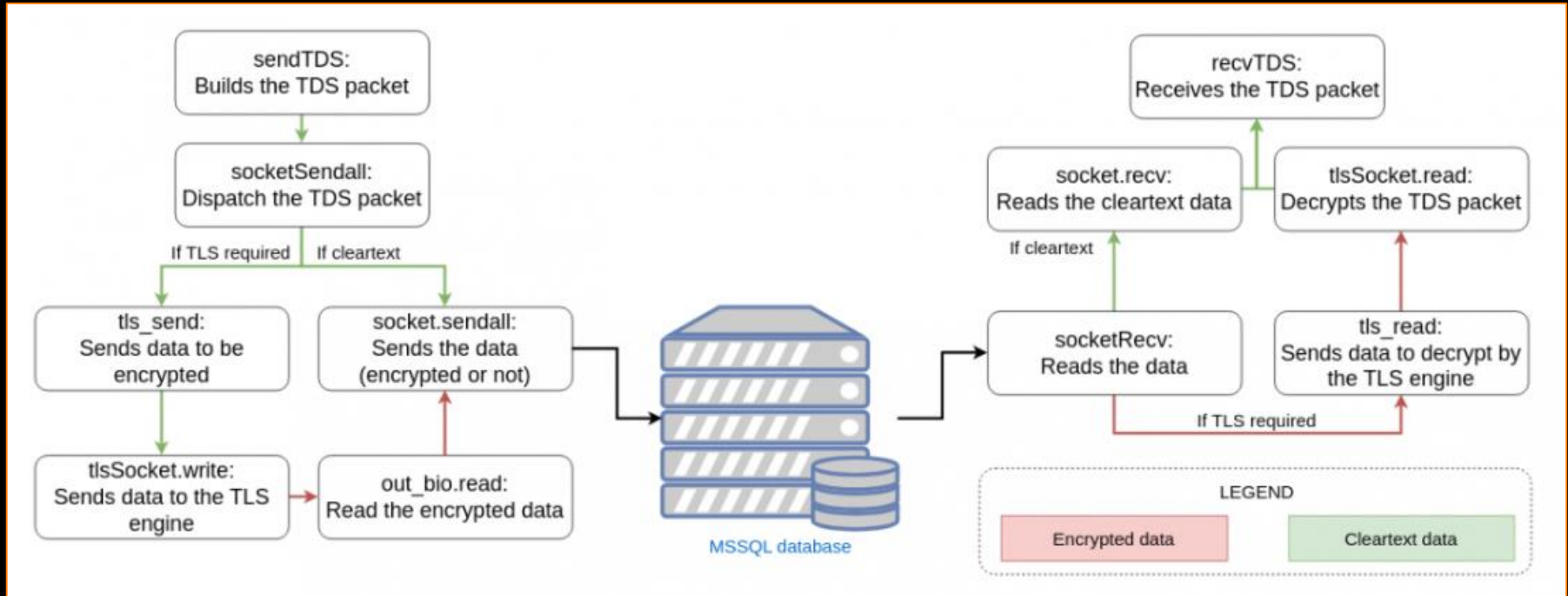
```
def generate_cbt_from_tls_unique(self) -> bytes:
    channel_binding_struct = b""
    initiator_address = b"\x00" * 8
    acceptor_address = b"\x00" * 8
    application_data_raw = b"tls-unique:" + self.tls_unique # This was tls-server-end-point before
    len_application_data = len(application_data_raw).to_bytes(4, byteorder="little", signed=False)
    application_data = len_application_data
    application_data += application_data_raw
    channel_binding_struct += initiator_address
    channel_binding_struct += acceptor_address
    channel_binding_struct += application_data
    cbt_token = md5(channel_binding_struct).digest()
    LOG.debug(f"Computed tls-unique CBT token: {cbt_token.hex()}")
    return cbt_token
```



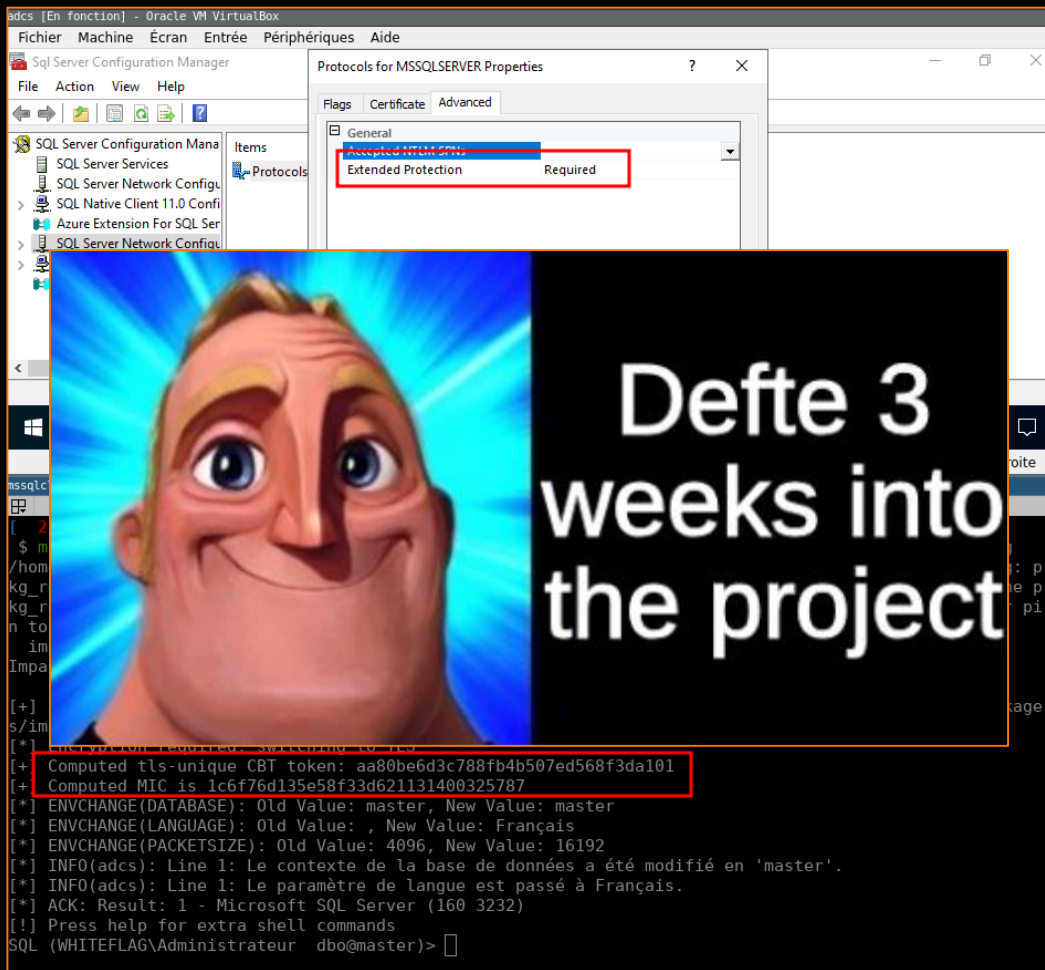
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x17\x00\x00\x00  
tls-unique;;\xbfx87\_\x85(\x94\xd562}]'

# BUT...

MSSQL supports both encryption via STARTTLS and plain text communications, so:



And...



The screenshot displays a Windows Virtual Machine environment. In the background, the 'SQL Server Configuration Manager' is open, showing the 'Protocols for MSSQLSERVER Properties' dialog box with the 'Extended Protection' checkbox checked and set to 'Required'. Overlaid on the terminal window is a large image of Mr. Incredible with the text 'Defte 3 weeks into the project'.

The terminal window shows the following commands and output:

```
sqlcmd -S . -u sa -P 'StrongPassword123!' -i C:\Users\WHITEFLAG\Documents\script.sql
[+] Computed tls-unique CBT token: aa80be6d3c788fb4b507ed568f3da101
[+] Computed MIC is 1c6f76d135e58f33d621131400325787
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: Français
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(adcs): Line 1: Le contexte de la base de données a été modifié en 'master'.
[*] INFO(adcs): Line 1: Le paramètre de langue est passé à Français.
[*] ACK: Result: 1 - Microsoft SQL Server (160 3232)
[!] Press help for extra shell commands
SQL (WHITEFLAG\Administrateur dbo@master)>
```

# Finally...

The screenshot displays a Windows VirtualBox environment with two windows. The left window, titled 'newadcs (Instantané 1) [En fonction] - Oracle VM VirtualBox', shows a menu bar with 'Fichier', 'Machine', 'Écran', 'Entrée', 'Périphériques', and 'Aide'. The right window, titled 'newserver (Instantané 1) [En fonction] - Oracle VM VirtualBox', shows a 'Propriétés de : Protocoles pour SQLEXPRESS' dialog box with the 'Général' tab selected. The 'Protection étendue' is set to 'Obligatoire' and 'SPN NTLM acceptés' is checked. A central overlay features a cartoon image of Mr. Incredible and the text 'Defte a month into the project'. The terminal at the bottom shows the following commands and output:

```
[ 5:48 ] [ ach@blackpearl:/opt/tools/ad/NetExec(mssql_cbt_reworkx) ]  
$ poetry run pxc mssql server.whiteflag.local -u Administrateur -p Defte@WF  
MSSQL 192.168.56.32 1433 SERVER [*] Windows Server 2022 Build 20348 (name:SERVER) (domain:whiteflag.local)  
MSSQL 192.168.56.32 1433 SERVER [+] whiteflag.local\Administrateur:Defte@WF (Pwn3d!)  
[ 5:48 ] [ ach@blackpearl:/opt/tools/ad/NetExec(mssql_cbt_reworkx) ]
```



## 7 / Sum up & conclusion

## Sum up

- **MSSQL databases are vulnerable to NTLMrelay attacks by default**
- **To prevent that, MS developed the Channel Binding token that is used to bind an authentication to a TLS tunnel**
- **To work, a TLS tunnel is mandatory (e.g CBT doesn't work on unencrypted channels)**
- **Channel Binding Tokens are built using three methods:**
  - **TLS Server End Point (used by LDAPS) : computed via the sha256 hash of the certificate**
  - **TLS Unique (used by MSSQL) : computed via the sha256 of the TLS Finished message**
  - **TLS Exporter (used by ???) : computed via ..... ?**
- **Computing the CBT value and adding it to the NTLMSSP\_AUTHENTICATE message grants us access to the most hardened MSSQL databases**

# Questions ?



**Mail:** [aurelien.chalot@orangecyberdefense.com](mailto:aurelien.chalot@orangecyberdefense.com)

**Twitter:** [https://x.com/Defte\\_](https://x.com/Defte_)

**Discord:** [deft\\_](#)

**Related blogpost :**

<https://blog.whiteflag.io/blog/a-journey-about-mssql/>

**Slides:** <https://github.com/Orange-Cyberdefense/conferences>