TEST D’INTRUSION

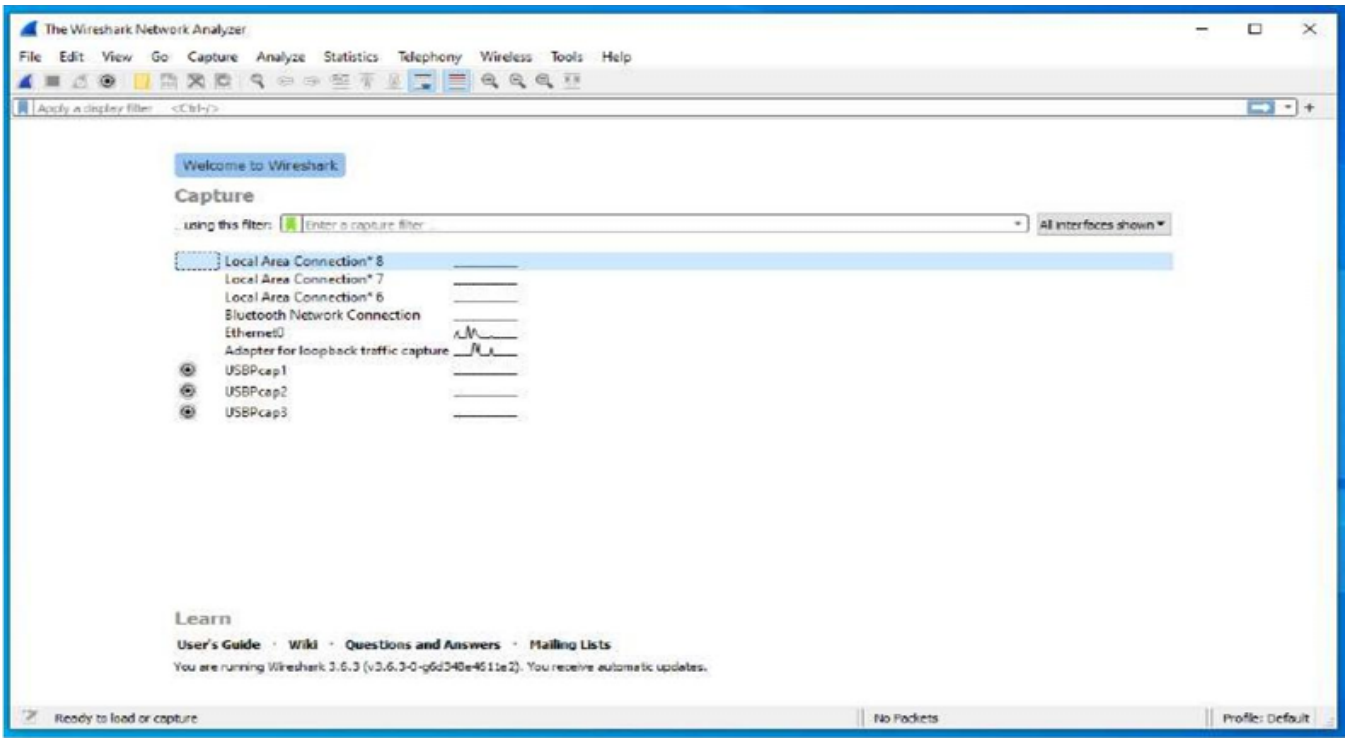
Ce document est toujours en cours de rédaction vu les mises à jour que j’ai effectuées depuis un bon moment et aussi les captures d’écran que je réunis pour les correspondances des tests. J’ai effectué ces tests, pour la plupart, dans des environnements virtuels.

Outils et attaques de réseau

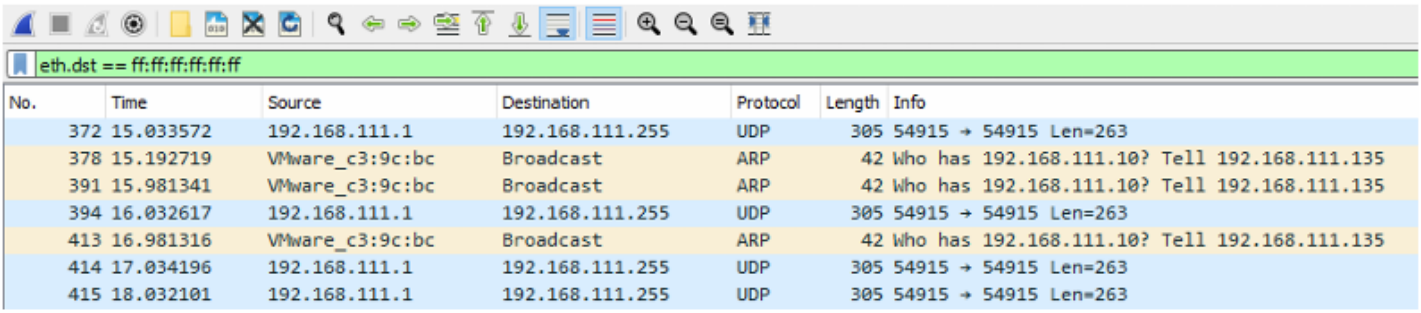
Ici, je vais commencer à couvrir certaines des attaques qui existent et les outils que je peux utiliser pour les réaliser. Comme le trafic traverse un réseau, je peux effectuer diverses attaques, telles que la capture du trafic et regarder ce qu'il contient, intercepter le trafic et le manipuler.

Capture de packets

La capture de paquets est également appelée reniflage. C'est le processus de capture des paquets quand ils traversent le réseau pour examiner l'intérieur et de découvrir toute information précieuse. En effectuant la capture de paquets, vous pouvez voir toutes sortes de trafic. Cela peut être du trafic à la fois protégé et non protégé. Divers outils qui peuvent effectuer la capture de paquets aujourd’hui existent. Je vais utiliser Wireshark ici.



Pour démarrer une capture de paquets avec Wireshark je vais sélectionner l'interface que je souhaite. Ensuite, je vais cliquer sur le bouton Démarrer pour commencer à capturer des paquets. Je verrai alors les résultats s'afficher dans la fenêtre d'affichage principale. La figure suivante est un exemple de requêtes ARP sur le réseau :



Lorsque j'effectue des captures de paquets, je dois filtrer la sortie pour rechercher mes résultats spécifiques. Le filtre d'affichage dans Wireshark fait exactement cela. Je peux le voir lorsque je regarde la capture ci-dessus - spécifiquement le texte dans la barre verte, indiquant eth.dst == ff:f:f:f:f:ff:f:ff. Lorsque je tape du texte dans le filtre d'affichage, Wireshark m'offre une liste de suggestions basées sur ce que j'ai tapé dedans. Il fournit également un moyen pour moi de voir si le filtre fonctionnera ou non - par exemple, si la barre devient jaune, cela signifie que le filtre d'affichage a été accepté, mais peut ne pas

fonctionner comme prévu. Si elle devient rouge, cela signifie que le filtre n'a pas été accepté et ne fonctionnera pas.

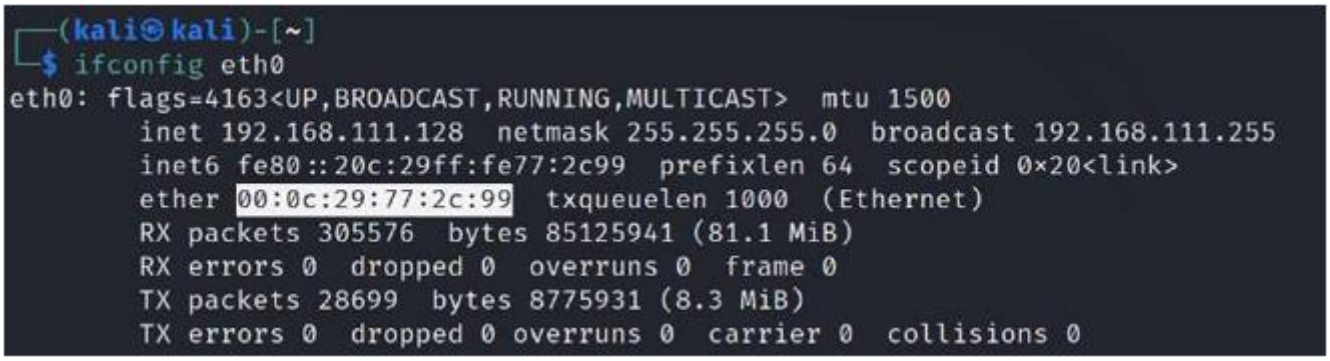
Il existe une liste exhaustive de filtres qui sont bien documentés sur la page de documentation de Wireshark. On peut trouver la liste complète ici: https://www.wireshark.org/docs/dfref/.

Usurpation d'adresse MAC

Chaque interface réseau a une adresse MAC unique. L'usurpation d'adresse MAC est un type de vol d'identité d'ordinateur qui implique de modifier l'adresse MAC sur la carte réseau. Les techniques d'usurpation d'adresse MAC sont couramment utilisées lors de la tentative de pénétrer dans un environnement LAN en supposant l'identité d'un ordinateur autorisé. Par exemple, certains réseaux peuvent blanchir des adresses MAC.

Voyons comment effectuer une usurpation d'adresse MAC. Pour cela, je vais utiliser Kali Linux et l'outil **macchanger**. À partir d'une fenêtre Terminal, j'execute la commande macchanger --help. Cela me montrera toutes les options qui sont disponibles à utiliser avec l'outil, comme indiqué dans la capture d'écran suivante:

Avant d'utiliser l'outil, je vais vérifier mon adresse MAC actuelle. Je peux le faire en utilisant la commande ifconfig, suivie par mon interface. Dans mon cas, c'est eth0, selon la figure suivante :



Actuellement, mon adresse MAC est 00:0c:29:77:2c:99. Maintenant, je vais le modifier à une certaine valeur aléatoire. Cela peut être fait en une seule étape facile:

Nous allons émettre la commande sudo macchanger-r eth0. J'utilise la commande sudo puisque mon utilisateur actuel n'a pas d'autorisations racine, -r est utilisé pour générer une adresse MAC aléatoire ; Je pourrais utiliser d'autres options si nécessaire. Par exemple, je pourrais définir une adresse MAC aléatoire du même type en utilisant le commutateur -a. Enfin, je vais définir mon interface, qui est eth0.

Les résultats présentés dans la figure suivante montrent que l'interface a maintenant une nouvelle adresse MAC de 06:1d:9f:2f:db:f6:

Si vous souhaitez définir votre propre adresse MAC, vous pouvez utiliser la commande suivante :

```
$ sudo macchanger  --mac XX:XX:XX:XX:XX:XX Set the MAC
XX:XX:XX:XX:XX:XX
```

Usurpation d’ARP

Les adresses MAC identifient qui vous êtes ; ce sont des identifications physiques. Les adresses IP sont utilisées pour identifier où vous êtes. Les tableaux ARP sont utilisés pour gérer la relation entre qui et où vous êtes.

L'ARP est utilisé pour découvrir l'adresse MAC liée à une adresse IP. Par exemple, si un routeur doit envoyer des données à un ordinateur qui détient l'adresse IP de 192.168.1. 20, il a besoin de connaître l'adresse MAC et de le découvrir, il enverra une requête ARP. Les requêtes ARP ne sont pas limitées aux routeurs ; d'autres périphériques, tels que les routeurs sans fil, les commutateurs et les ordinateurs, fonctionnent tous avec le protocole ARP.

Dans une attaque d'attaque d'ARP, l'attaquant envoie des fausses réponses ARP à une victime. Ces réponses indiquent essentiellement à la victime que l'adresse MAC de l'attaquant est mappée à autre chose, telle qu'une adresse IP du routeur. Cela signifie que la victime enverrait des paquets qui étaient initialement destinés au routeur à l'attaquant que l'adresse MAC du routeur serait remplacé par l'adresse MAC de l'attaquant. L'usurpation ARP est un exemple typique d'une attaque **man-in-the-middle** (MITM).

À partir de la machine Kali, j’ouvre un terminal et j’execute la commande suivante. Cela va commencer à usurper les paquets ARP vers la machine Windows 10:

```
$ sudo arpspoof -i eth0 -t 192.168.1.20 192.168.1.1
```

La commande continuera à s'exécuter indéfiniment jusqu'à ce que vous l'annuliez avec la séquence de touches *Ctrl + C*. La sortie se déroulera comme suit :

```
┌──(kali㉿kali)-[~]
└─$ sudo arpspoof -i eth0 -t 192.168.1.20 192.168.1.1
0:c:29:77:2c:99 0:c:29:ab:a3:28 0806 42: arp reply 192.168.1.1 is-at 0:c:29:77:2c:99
0:c:29:77:2c:99 0:c:29:ab:a3:28 0806 42: arp reply 192.168.1.1 is-at 0:c:29:77:2c:99
0:c:29:77:2c:99 0:c:29:ab:a3:28 0806 42: arp reply 192.168.1.1 is-at 0:c:29:77:2c:99
```

Puisque je veux intercepter le trafic et effectuer une attaque MITM, je vais mener une attaque arpspoof vers le routeur. Ici, je dis au routeur que tout le trafic destiné à l'adresse MAC appartenant à 192. 168.1.20 (machine Windows 10) est mon adresse MAC (Kali Machine). Kali effectuera ensuite le transfert de paquets entre le routeur et la machine Windows 10.

Dans une nouvelle fenêtre Terminal, exécutez la commande suivante. Cela va commencer à usurper les paquets ARP vers le routeur pfSense:

```
$ sudo arpspoof -i eth0 -t 192.168.1.1 192.168.1.20
```

La sortie se fera comme suit :

```
┌──(kali㉿kali)-[~]
└─$ sudo arpspoof -i eth0 -t 192.168.1.1 192.168.1.20
[sudo] password for kali:
0:c:29:77:2c:99 0:c:29:c:c:fe 0806 42: arp reply 192.168.1.20 is-at 0:c:29:77:2c:99
0:c:29:77:2c:99 0:c:29:c:c:fe 0806 42: arp reply 192.168.1.20 is-at 0:c:29:77:2c:99
0:c:29:77:2c:99 0:c:29:c:c:fe 0806 42: arp reply 192.168.1.20 is-at 0:c:29:77:2c:99
```

Comme les deux commandes s'exécutent, le routeur pfSense et la machine Windows 10 devrait maintenant avoir des entrées ARP poison. De là, je peux intercepter les paquets et voir toute la communication entre la machine Windows 10 et le routeur.

Un excellent outil pour visualiser rapidement le trafic web est appelé **URLSnarf**, qui fait partie de la suite dSniff. Je vais essayer ceci dans mon laboratoire en entrant la commande suivante :

```
$ sudo urlsnarf -i eth0
```

Cette commande inspectera tout le trafic Web et me fournira les URL à laquelle j'accède. Un exemple de ceci peut être vu dans la capture d'écran suivante :

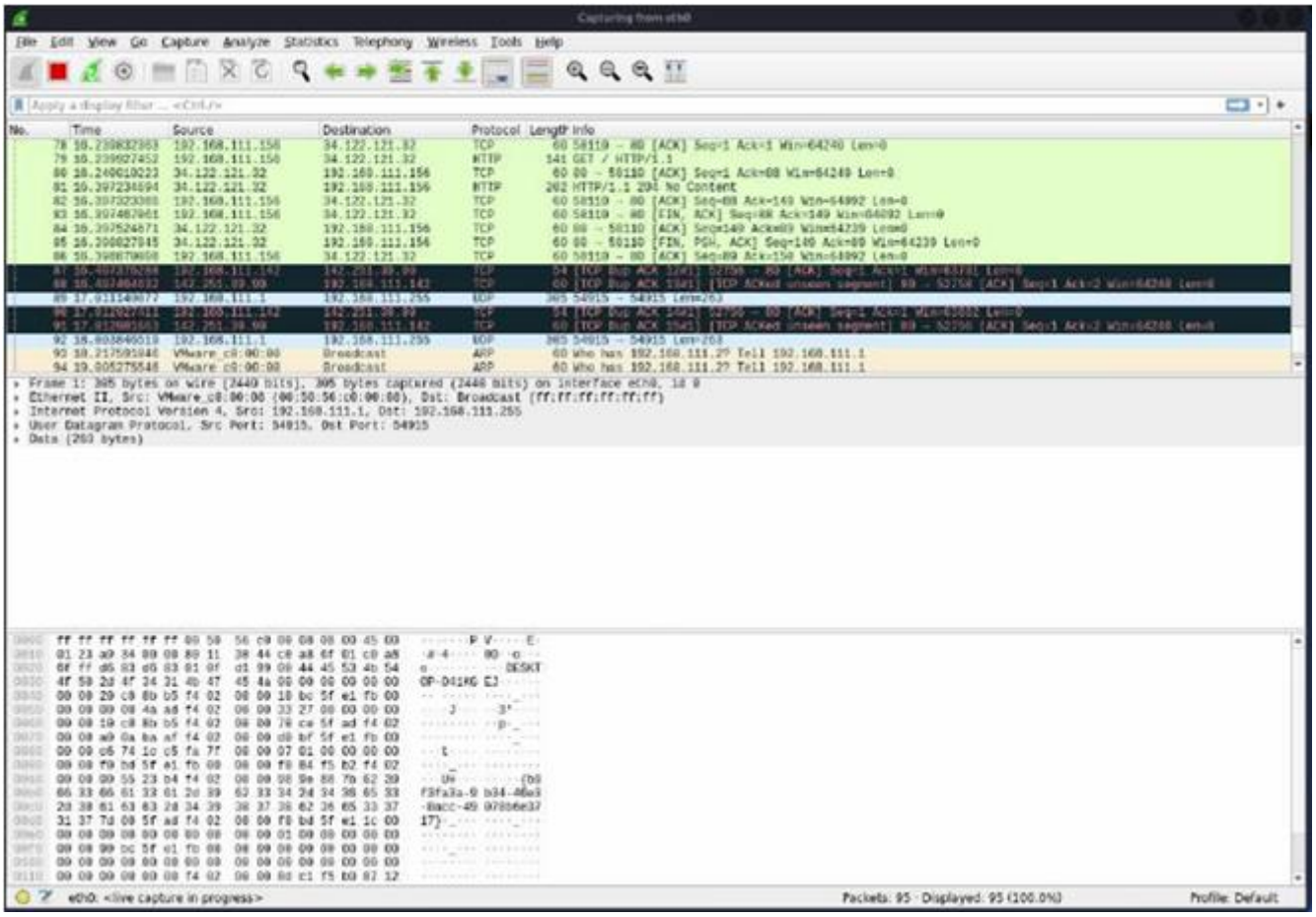
```
┌──(kali㉿kali)-[~]
└─$ sudo urlsnarf -i eth0
urlsnarf: listening on eth0 [tcp port 80 or port 8080 or port 3128]
192.168.1.20 - - [02/May/2022:07:25:29 -0400] "GET http://www.citi.com/ HTTP/1.1" -
 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/70.0.3538.102 Safari/537.36 Edge/18.19041"
192.168.1.20 - - [02/May/2022:07:25:29 -0400] "GET http://ocsp.digicert.com/MFEwTzB
NMEswSTAJBgUrDgMCGgUABBTfqhLjKLEJQZPin0KCzkdAQpVYowQUsT7DaQP4v0cB1JgmGggC72NkK8MCEA
x5qUSwjBGVIJJhX%2BJrHYM%3D HTTP/1.1" - - "-" "Microsoft-CryptoAPI/10.0"
192.168.1.20 - - [02/May/2022:07:25:31 -0400] "GET http://ocsp.comodoca.com/MFEwTzB
NMEswSTAJBgUrDgMCGgUABBRTtU9uFqgVGHhJwXZyWCNXmVR5ngQUoBEKIz6W8Qfs4q8p74Klf9AwpLQCED
lyRDr5IrdR19NsEN0xNZU%3D HTTP/1.1" - - "-" "Microsoft-CryptoAPI/10.0"
192.168.1.20 - - [02/May/2022:07:25:31 -0400] "GET http://ocsp.usertrust.com/MFEwTz
BNMEswSTAJBgUrDgMCGgUABBTNMNJMNDqCqx8FcBWK16EHdimS6QQUU3m%2FWqorSs9UgOHYm8Cd8rIDZss
CEH1bUSa0droR23QWC7xTDac%3D HTTP/1.1" - - "-" "Microsoft-CryptoAPI/10.0"
192.168.1.20 - - [02/May/2022:07:25:31 -0400] "GET http://ocsp.digicert.com/MFEwTzB
NMEswSTAJBgUrDgMCGgUABBTfqhLjKLEJQZPin0KCzkdAQpVYowQUsT7DaQP4v0cB1JgmGggC72NkK8MCEA
```

Capture et analyse du trafic réseau

Capture et analyse du trafic non chiffré

Je vais commencer par utiliser DVWA. Une fois la capture démarrée, les résultats apparaitront sur le tableau de bord de Wireshark. Ensuite, je vais basculer vers mon

navigateur et sur la page de connexion, me connecter avec l'administrateur d'utilisateur et le mot de passe. Mon écran dans Wireshark ressemblera à la capture d'écran suivante :

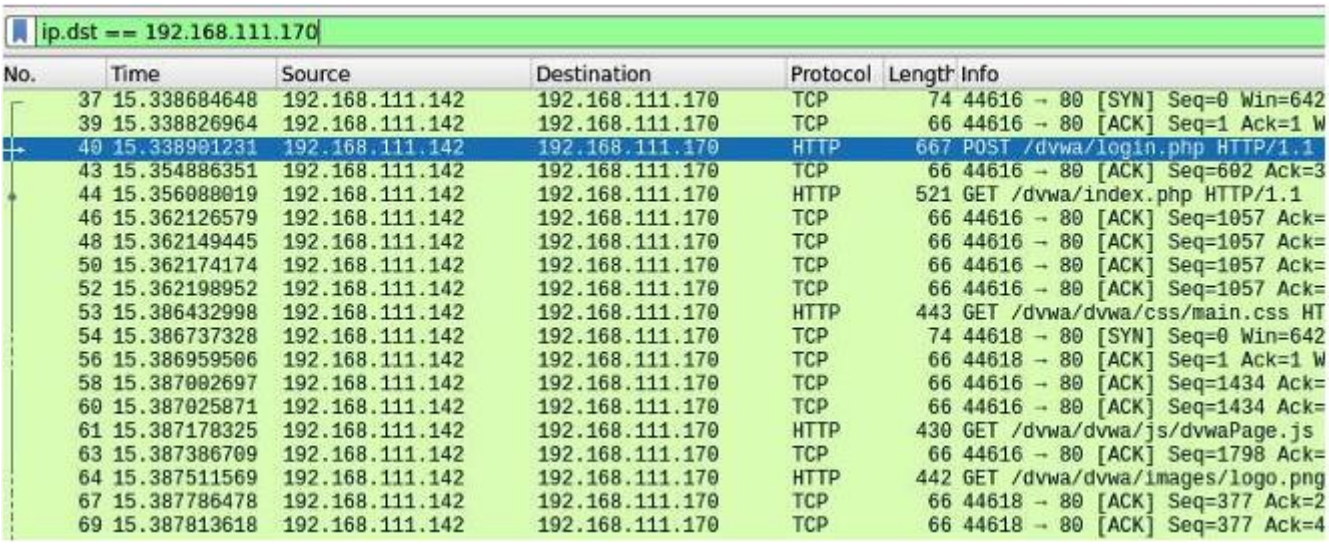


Maintenant que j'ai ma capture de paquets, je vais explorer ce qui peut être fait avec elle. Lorsque je me suis connecté à l'application DVWA, j'ai lancé un protocole d'**HyperText Transfer Protocol** (**HTTP**). HTTP est insécurisé, ce qui signifie que tout trafic transféré avec ce protocole est fait en texte clair. Je vais utiliser des filtres pour une bonne analyse de mes résultats. Le premier que je vais utiliser est le filtre de destination. Les filtres sur Wireshark sont basés sur la syntaxe :

```
[Protocol].[header/field] [operator: +,==,!=] [value]
```

Basé sur cette syntaxe, le filtre que je vais utiliser est ip.dst == 192.168.111.170. Ici, je définis le protocole IP et le champ dst. Je définis ensuite un opérateur de correspondance, qui est défini comme ==, puis la valeur, qui est 192.168.111.170.

Une fois que le filtre a été appliqué, je verrai tout le trafic qui est destiné à la machine virtuelle Metasploitable 2. Mes résultats ressembleront à la capture d'écran suivante :



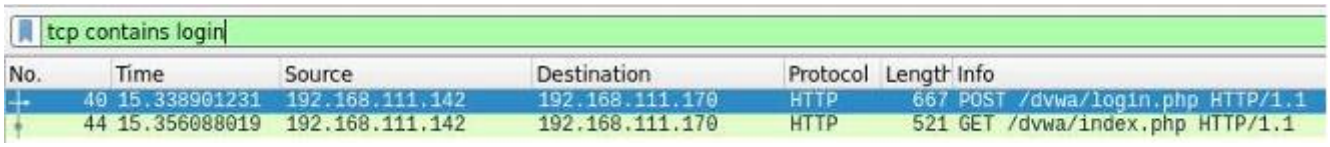
ip.dst == 192.168.111.170

No.	Time	Source	Destination	Protocol	Length	Info
37	15.338684648	192.168.111.142	192.168.111.170	TCP	74	44616 → 80 [SYN] Seq=0 Win=642
39	15.338826964	192.168.111.142	192.168.111.170	TCP	66	44616 → 80 [ACK] Seq=1 Ack=1 W
40	15.338901231	192.168.111.142	192.168.111.170	HTTP	667	POST /dvwa/login.php HTTP/1.1
43	15.354886351	192.168.111.142	192.168.111.170	TCP	66	44616 → 80 [ACK] Seq=602 Ack=3
44	15.356088019	192.168.111.142	192.168.111.170	HTTP	521	GET /dvwa/index.php HTTP/1.1
46	15.362126579	192.168.111.142	192.168.111.170	TCP	66	44616 → 80 [ACK] Seq=1057 Ack=
48	15.362149445	192.168.111.142	192.168.111.170	TCP	66	44616 → 80 [ACK] Seq=1057 Ack=
50	15.362174174	192.168.111.142	192.168.111.170	TCP	66	44616 → 80 [ACK] Seq=1057 Ack=
52	15.362198952	192.168.111.142	192.168.111.170	TCP	66	44616 → 80 [ACK] Seq=1057 Ack=
53	15.386432998	192.168.111.142	192.168.111.170	HTTP	443	GET /dvwa/dvwa/css/main.css HT
54	15.386737328	192.168.111.142	192.168.111.170	TCP	74	44618 → 80 [SYN] Seq=0 Win=642
56	15.386959506	192.168.111.142	192.168.111.170	TCP	66	44618 → 80 [ACK] Seq=1 Ack=1 W
58	15.387002697	192.168.111.142	192.168.111.170	TCP	66	44616 → 80 [ACK] Seq=1434 Ack=
60	15.387025871	192.168.111.142	192.168.111.170	TCP	66	44616 → 80 [ACK] Seq=1434 Ack=
61	15.387178325	192.168.111.142	192.168.111.170	HTTP	430	GET /dvwa/dvwa/js/dvwaPage.js
63	15.387386709	192.168.111.142	192.168.111.170	TCP	66	44616 → 80 [ACK] Seq=1798 Ack=
64	15.387511569	192.168.111.142	192.168.111.170	HTTP	442	GET /dvwa/dvwa/images/logo.png
67	15.387786478	192.168.111.142	192.168.111.170	TCP	66	44618 → 80 [ACK] Seq=377 Ack=2
69	15.387813618	192.168.111.142	192.168.111.170	TCP	66	44618 → 80 [ACK] Seq=377 Ack=4

Le filtre m’a permis de réduire les résultats afin que je puisse me concentrer sur tout le trafic vers ma destination spécifiée. Cependant, les résultats sont encore très étendus. Donc, je vais tirer profit d'un filtre différent qui cherche des connexions basées sur le TCP. Le filtre que je vais utiliser est le suivant :

```
tcp contains login
```

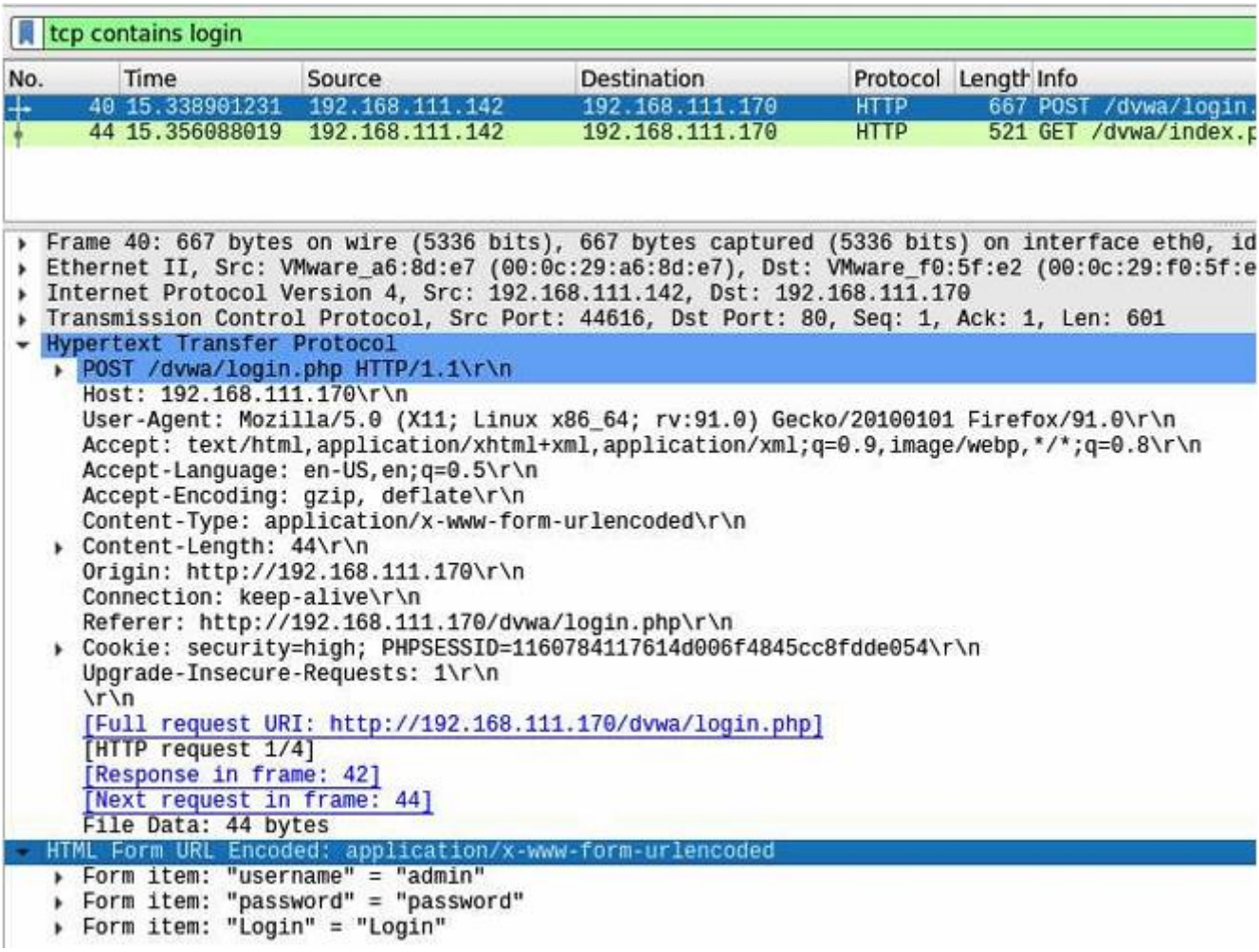
Une fois appliqué, mes résultats ressemblent à ce qui suit :



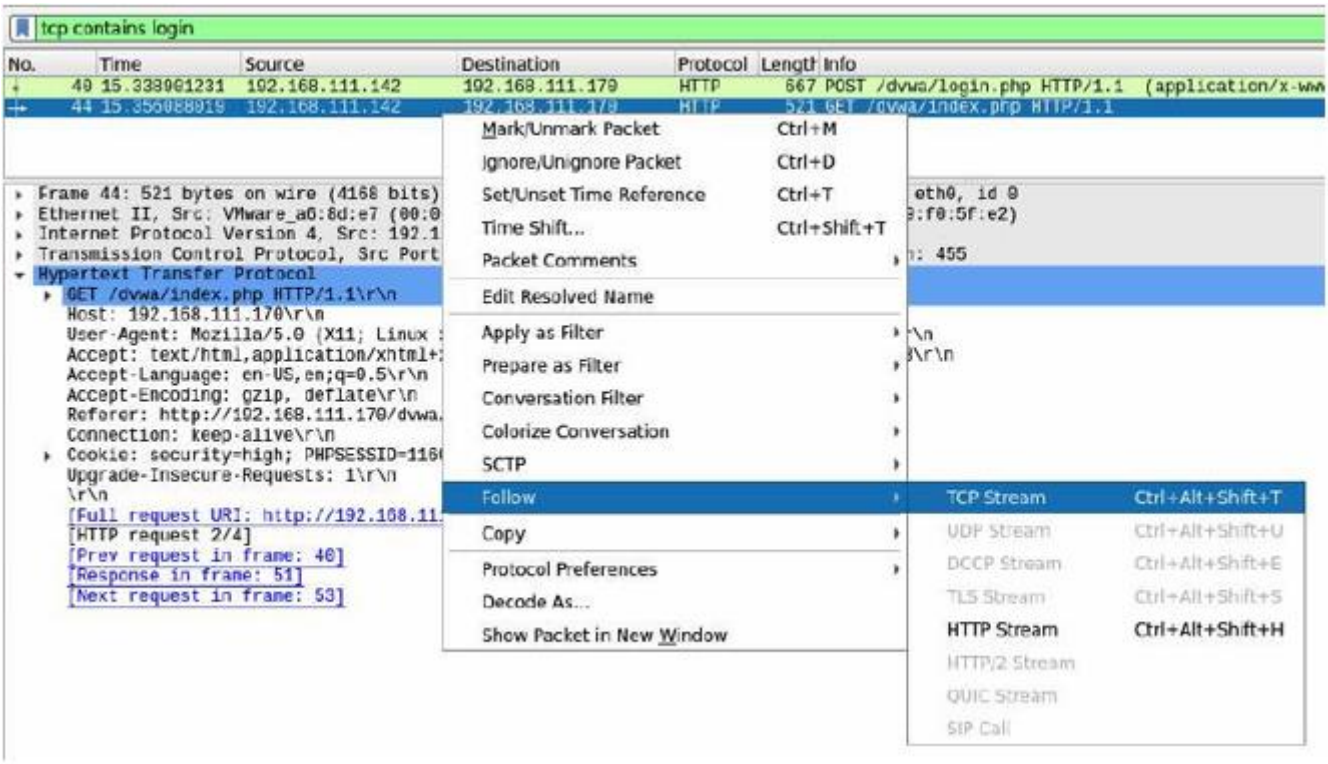
tcp contains login

No.	Time	Source	Destination	Protocol	Length	Info
40	15.338901231	192.168.111.142	192.168.111.170	HTTP	667	POST /dvwa/login.php HTTP/1.1
44	15.356088019	192.168.111.142	192.168.111.170	HTTP	521	GET /dvwa/index.php HTTP/1.1

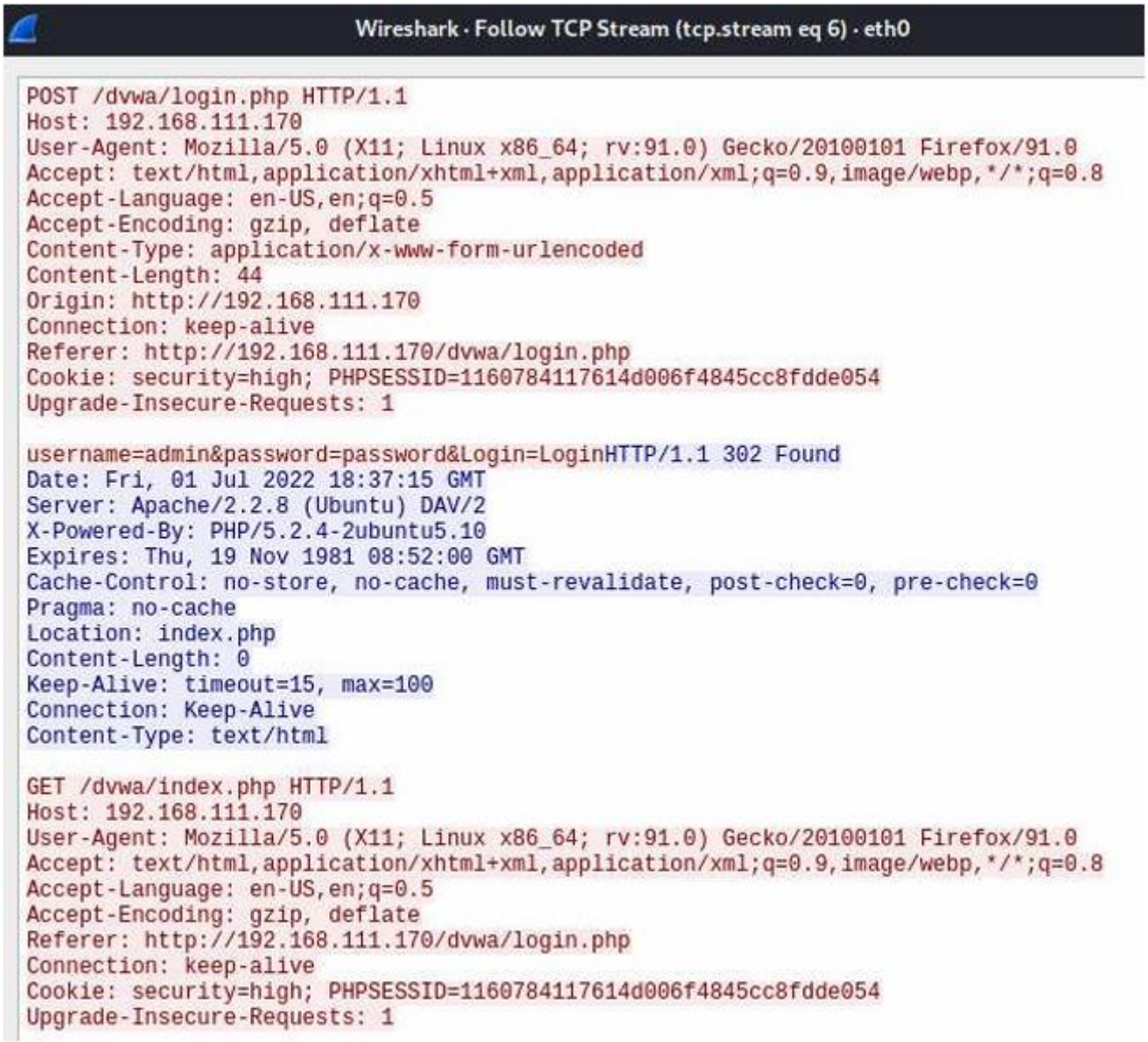
Maintenant, mes résultats sont beaucoup plus précis. Examinons les données dans ces résultats. Développer les différentes sections me fournira des informations telles que la source et la destination MAC adresses (**Ethernet II**). Je serai en mesure de voir les adresses IP source et de destination (**Internet Protocol Version 4**) et les ports TCP source et destination (**Protocole de contrôle de transmission**). Enfin, j’ai la section du protocole de transfert d'hypertexte, qui me fournit des informations sur la demande web réelle. Lorsque je développe ces champs, je remarque le nom d'utilisateur et le mot de passe en texte brut, comme indiqué dans la capture d'écran suivante :



Wireshark permet de remonter le flux de paquets d'une connexion. Cette fonctionnalité permet de visualiser le flux de données complet, y compris les données qui ont été envoyées et reçues. Je vais essayer cela sur l'ensemble de paquets actuel que j’ai appliqué avec le filtre de connexion.



Maintenant, je serai en mesure de voir le flux complet de données liées à ce paquet. Lorsque je fais défiler la sortie, je verrai les requêtes POST et GET entre mon client et l'application DVWA de Metasploitable 2, y compris les informations d'identification du texte clair, comme illustré dans la figure suivante :

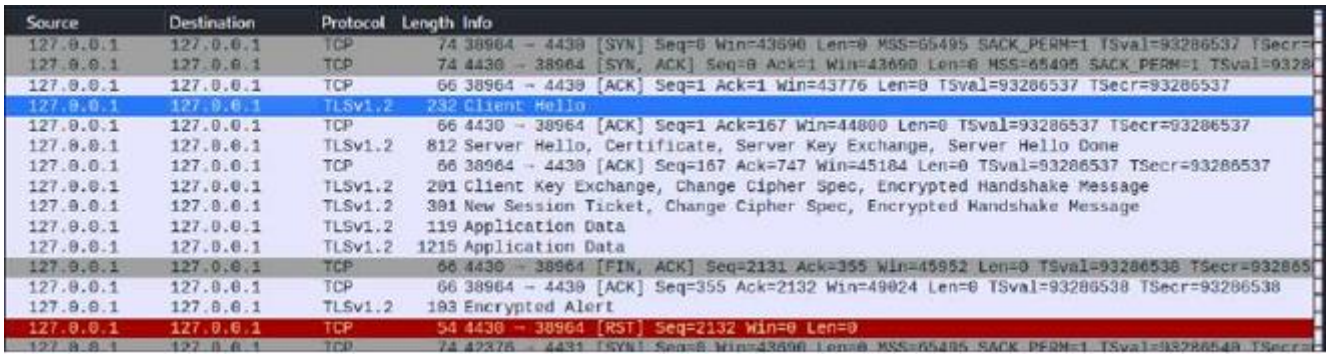


Ainsi, la capture du trafic HTTP est simple et peut récolter des récompenses en ce qui concerne l'obtention d'informations texte clair.

Capture et analyse du trafic chiffré

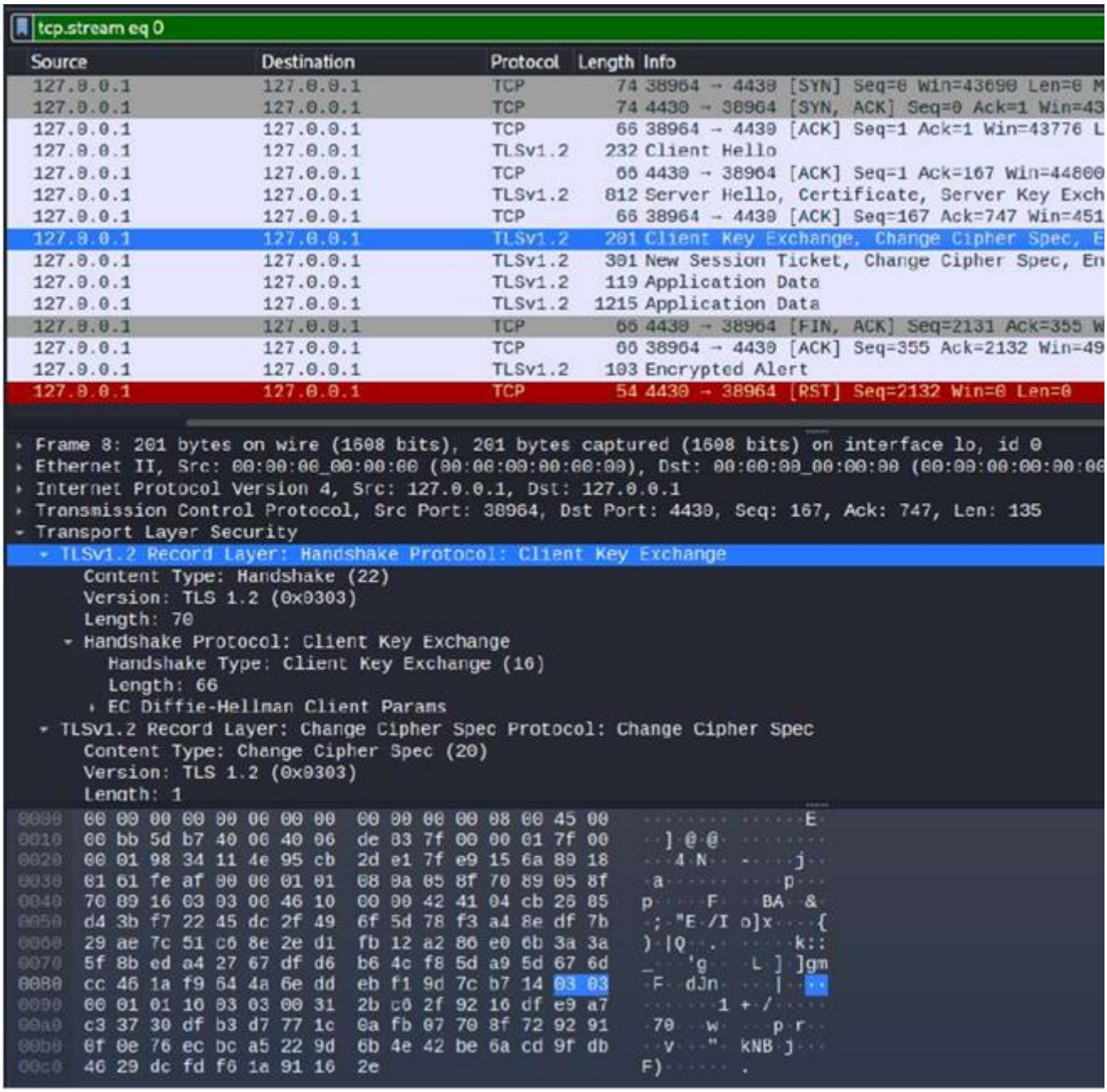
Wireshark héberge un certain nombre de captures de paquets que je peux utiliser pour performer mes compétences d'analyse. Je vais me concentrer sur celui qui concerne TLS v1.2. La **capture de paquets** (**pcap**) se trouve à l'adresse URL suivante: https://bugs.wireshark.org/bugzilla/attachet.cgi?id=11612. J'aurai besoin de la clé de premaster afin de décrypter le trafic. Il se trouve à l'adresse https://bugs.wireshark.org/bugzilla/attachet.cgi?id=11616. Je vais téléchargez ces deux fichiers et travailler avec eux sur Wireshark sur ma machine Kali Linux.

Une fois ma capture ouverte, je remarque que le protocole utilisé est TLS v1.2, comme le montre la capture d'écran suivante:



Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	TCP	74	38964 → 4430 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=93286537 TSecr=
127.0.0.1	127.0.0.1	TCP	74	4430 → 38964 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=9328
127.0.0.1	127.0.0.1	TCP	66	38964 → 4430 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=93286537 TSecr=93286537
127.0.0.1	127.0.0.1	TLSv1.2	232	Client Hello
127.0.0.1	127.0.0.1	TCP	66	4430 → 38964 [ACK] Seq=1 Ack=167 Win=44800 Len=0 TSval=93286537 TSecr=93286537
127.0.0.1	127.0.0.1	TLSv1.2	812	Server Hello, Certificate, Server Key Exchange, Server Hello Done
127.0.0.1	127.0.0.1	TCP	66	38964 → 4430 [ACK] Seq=167 Ack=747 Win=45184 Len=0 TSval=93286537 TSecr=93286537
127.0.0.1	127.0.0.1	TLSv1.2	201	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
127.0.0.1	127.0.0.1	TLSv1.2	301	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
127.0.0.1	127.0.0.1	TLSv1.2	119	Application Data
127.0.0.1	127.0.0.1	TLSv1.2	1215	Application Data
127.0.0.1	127.0.0.1	TCP	66	4430 → 38964 [FIN, ACK] Seq=2131 Ack=355 Win=45952 Len=0 TSval=93286538 TSecr=932865
127.0.0.1	127.0.0.1	TCP	66	38964 → 4430 [ACK] Seq=355 Ack=2132 Win=49024 Len=0 TSval=93286538 TSecr=93286538
127.0.0.1	127.0.0.1	TLSv1.2	103	Encrypted Alert
127.0.0.1	127.0.0.1	TCP	54	4430 → 38964 [RST] Seq=2132 Win=0 Len=0
127.0.0.1	127.0.0.1	TCP	74	42378 → 4431 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=93286540 TSecr=

Lorsque je sélectionne l'un des paquets du protocole TLS v1.2, je vois que le texte brut est brouillé. Par exemple, dans la capture d'écran suivante, j'ai sélectionné **Client Key Exchange, Change Cipher Spec, Encrypted Message Handshake** et je suis incapable de voir quoi que ce soit dans le texte clair:

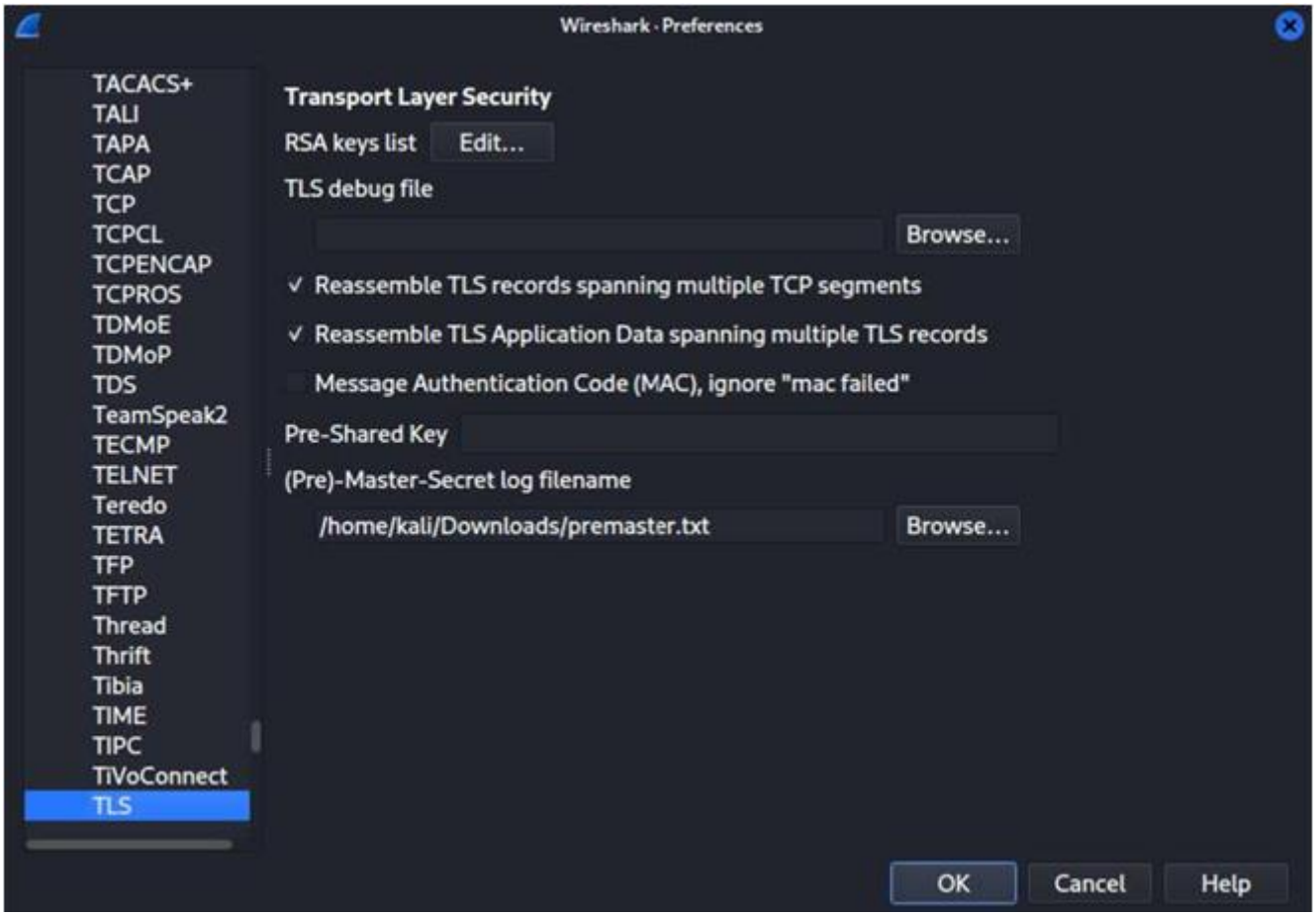


Pour afficher ce flux TLS, j'aurais besoin d'utiliser les touches de la session de cryptage. Ces clés peuvent être capturées par l'intermédiaire d'une attaque man-in-the-middle. Un échantillon des clés peut être vu dans la capture d'écran suivante :

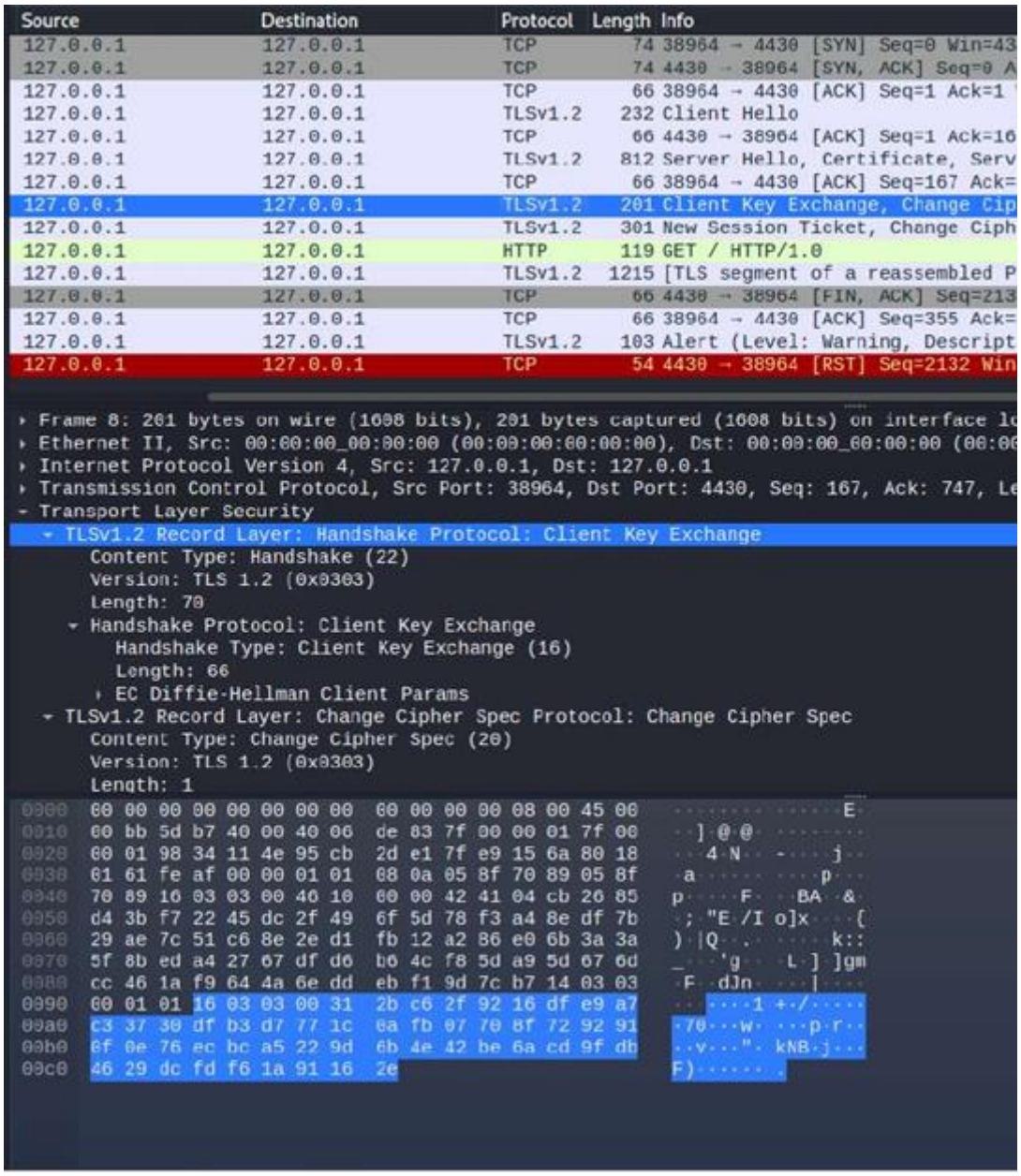
```
1 # Cipher Suite ECDHE-RSA-AES256-GCM-SHA384
2 CLIENT_RANDOM 52362c10a2665e323a2adb4b9da0c10d4a8823719272f8b4c97af24f92784812
  9F9A0F19A02BDDBE1A05926597D622CCA06D2AF416A28AD9C03163B87FF1B0C67824BBDB595B32D80
3 CLIENT_RANDOM 52362c1012cf23628256e745e903cea696e9f62a60ba0ae8311d70dea5e41949
  9F9A0F19A02BDDBE1A05926597D622CCA06D2AF416A28AD9C03163B87FF1B0C67824BBDB595B32D80
4 # Cipher Suite ECDHE-ECDSA-AES256-GCM-SHA384
5 CLIENT_RANDOM 52362c10dc11d7fda79a49a516e8b725ee3f453b74307c9a84d0bd557a0ed992
  C8614BBBC645CE250080C506527596A7FAAC5158E7DF4C5630210520855F6DB501EA396830F1409AC
6 CLIENT_RANDOM 52362c10a4530e7bcee0704c9dce137017972c1bbeaa8143cea2f17b7881a837
  C8614BBBC645CE250080C506527596A7FAAC5158E7DF4C5630210520855F6DB501EA396830F1409AC
7 # Cipher Suite ECDHE-RSA-AES256-SHA384
8 CLIENT_RANDOM 52362c108f76c8c0dcce66f5d575a72c5c33900bd3159b803757aaa528599297
  7C76985C7E4961A89AF1771A17BB7129AA8E43341F1A1CE23B71F9A1837D8DFFF6404E2B87AE52B28
9 CLIENT_RANDOM 52362c101f4ff95c15cb6d385a1d4e432f5c6bbfaeadd37ae192c298675e4b10
  7C76985C7E4961A89AF1771A17BB7129AA8E43341F1A1CE23B71F9A1837D8DFFF6404E2B87AE52B28
10 # Cipher Suite ECDHE-ECDSA-AES256-SHA384
11 CLIENT_RANDOM 52362c1047ce0b34c18471f06cbc00c0fc8b0569a2db68aeb08fcff5bb86ff35
   EECB64AE71CFAB29AF82D053664C4255F2598B5BD7CB5386DA8460273E82F0DF5EF13914EF99AE899
12 CLIENT_RANDOM 52362c10b2dea601cfa46be974650eefd0ff91aefcd9dbd6fcf66f2582b62484
   EECB64AE71CFAB29AF82D053664C4255F2598B5BD7CB5386DA8460273E82F0DF5EF13914EF99AE899
13 # Cipher Suite ECDHE-RSA-AES256-SHA
14 CLIENT_RANDOM 52362c10be8774a91228070fe82326ef3378ab509ccf45d5d1a8b823a08d6d57
```

Heureusement, j'ai déjà ce fichier journal à portée de main, comme je l'ai téléchargé plus tôt. Pour le charger dans Wireshark, j'ai besoin de naviguer vers le menu principal et cliquez sur **Edit | Preferences**. Dans **Preferences**, je sélectionne **Protocols** sur le côté gauche de la fenêtre et je fais défiler vers le bas jusqu'à **TLS**.

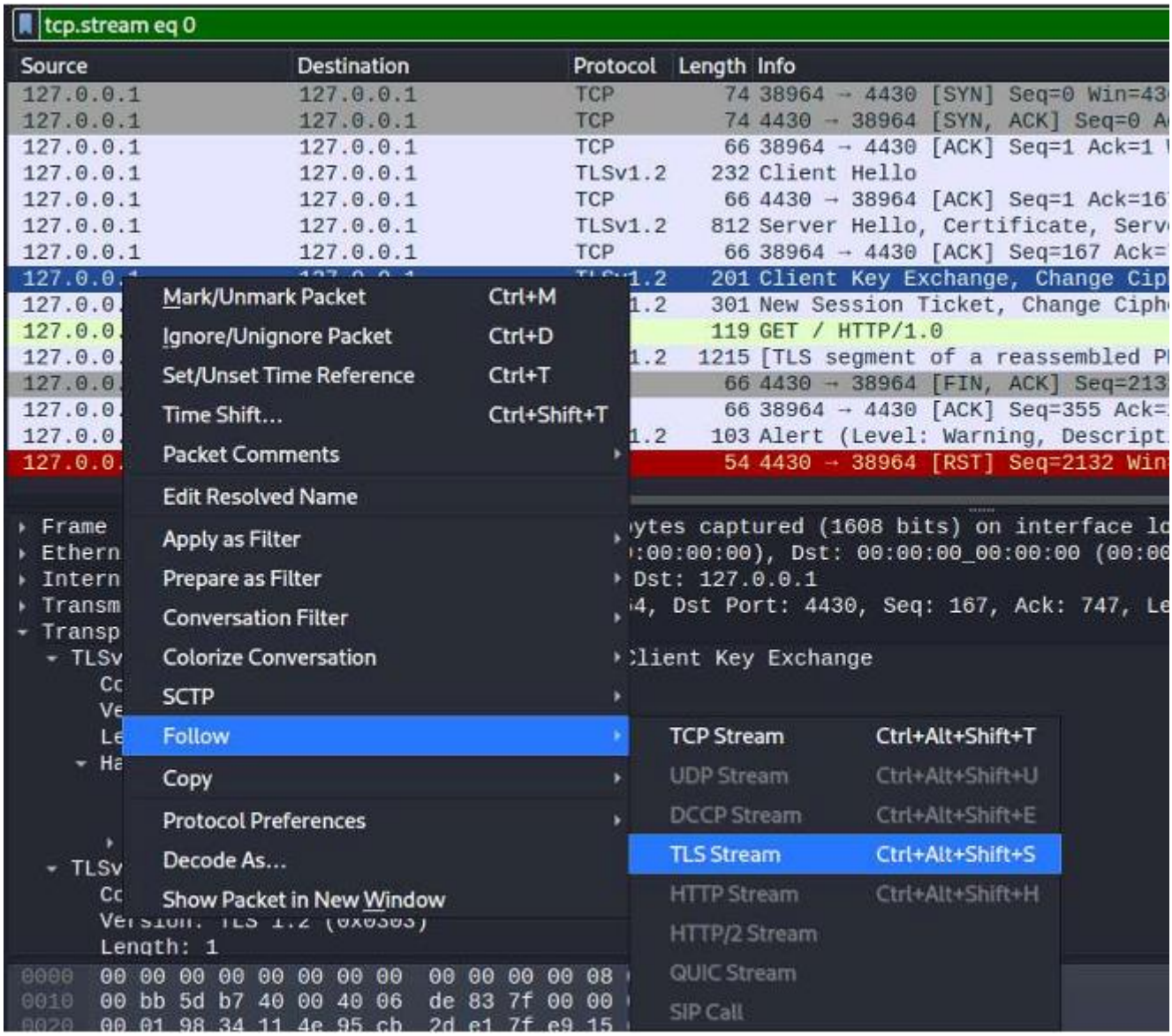
Une fois **TLS** sélectionné, je vois un champ appelé **(Pre)-Master-Secret nom de filename**. Ici, je sélectionne le fichier premaster.txt que j'ai téléchargé et je clique sur **OK**.



Une fois que vous avez chargé le fichier premaster.txt, vous verrez que Wireshark vous montre un onglet supplémentaire au bas de la fenêtre intitulée **Decrypted TLS**, comme illustré dans la capture d'écran suivante:



Maintenant que je peux déchiffrer la communication, je peux faire un clique avec le bouton droit sur le paquet et sélectionnez **Follow | TLS Stream**.



Maintenant, je suis en mesure de voir l'échange de clé de chiffre entre les deux points de terminaison dans ce flux de communication TLS. La première partie de la communication montre les chiffres qui sont pris en charge par le serveur. Ceux-ci peuvent être vus dans la capture d'écran suivante:

```
GET / HTTP/1.0

HTTP/1.0 200 ok
Content-type: text/html

<HTML><BODY BGCOLOR="#ffffff">
<pre>

s_server -accept 4430 -cert /tmp/test-certs/server.crt -key /tmp/test-certs/server.pem -www
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1/SSLv3:ECDHE-RSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDHE-ECDSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDHE-RSA-AES256-SHA384  TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA384
TLSv1/SSLv3:ECDHE-RSA-AES256-SHA     TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA
TLSv1/SSLv3:SRP-DSS-AES-256-CBC-SHA  TLSv1/SSLv3:SRP-RSA-AES-256-CBC-SHA
TLSv1/SSLv3:DHE-DSS-AES256-GCM-SHA384TLSv1/SSLv3:DHE-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:DHE-RSA-AES256-SHA256    TLSv1/SSLv3:DHE-DSS-AES256-SHA256
TLSv1/SSLv3:DHE-RSA-AES256-SHA       TLSv1/SSLv3:DHE-DSS-AES256-SHA
TLSv1/SSLv3:DHE-RSA-CAMELLIA256-SHA  TLSv1/SSLv3:DHE-DSS-CAMELLIA256-SHA
TLSv1/SSLv3:ECDH-RSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDH-ECDSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDH-RSA-AES256-SHA384   TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA384
TLSv1/SSLv3:ECDH-RSA-AES256-SHA      TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA
TLSv1/SSLv3:AES256-GCM-SHA384        TLSv1/SSLv3:AES256-SHA256
TLSv1/SSLv3:AES256-SHA               TLSv1/SSLv3:CAMELLIA256-SHA
TLSv1/SSLv3:PSK-AES256-CBC-SHA       TLSv1/SSLv3:ECDHE-RSA-DES-CBC3-SHA
TLSv1/SSLv3:ECDHE-ECDSA-DES-CBC3-SHA TLSv1/SSLv3:SRP-DSS-3DES-EDE-CBC-SHA
TLSv1/SSLv3:SRP-RSA-3DES-EDE-CBC-SHA TLSv1/SSLv3:EDH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:EDH-DSS-DES-CBC3-SHA     TLSv1/SSLv3:ECDH-RSA-DES-CBC3-SHA
```

Lorsque je fais défiler vers le bas, je vois le code qui a été sélectionné:

```
---
Ciphers common between both SSL end points:
ECDHE-RSA-AES256-GCM-SHA384
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID:
    Session-ID-ctx: 01000000
    Master-Key: 9F9A0F19A02BDDBE1A05926597D622CCA06D2AF416A2
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Compression: 1 (zlib compression)
    Start Time: 1379281936
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
```

La capacité de décrypter les communications chiffrées dépend en grande partie d'avoir les fichiers clés de log correctes.

Intrusion par effraction

Reconnaissance

La reconnaissance peut être définie comme une enquête qui est effectuée pour obtenir autant d'informations que possible sur une cible. Elle vise à vous permettre d'avoir un pied dans votre environnement cible.

Les deux principales catégories de collecte d'informations sont les suivantes :

• Collecte d'informations passives

• Collecte d'informations actives

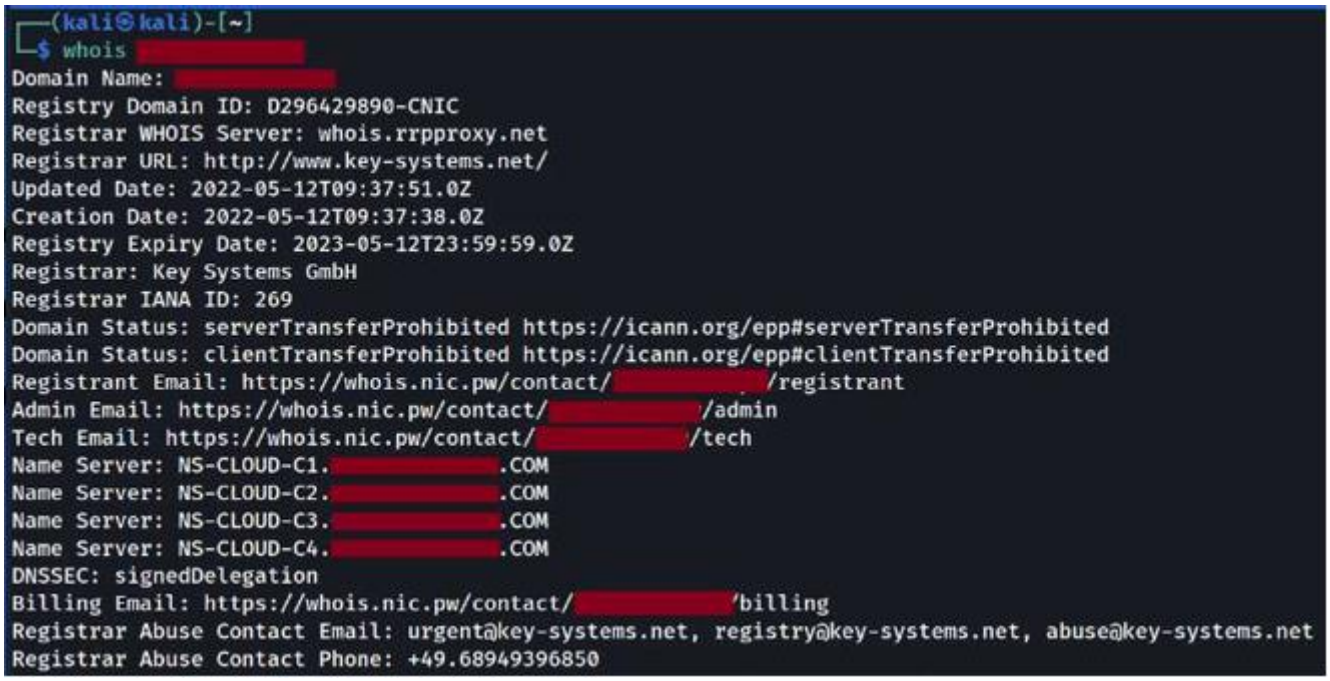
Collecte d'informations passives

La collecte d'informations passives implique collecte d'informations où vous n'interagissez pas directement avec votre cible. Elle est souvent réalisée en exploitant des ressources accessibles au public. Cela peut être comparé à la collecte d'informations sur votre cible à distance. Prenons un exemple d'utilisation de l'outil WHOIS dans Kali Linux. Pour exécuter une requête WHOIS, je vais effectuer les opérations suivantes :

1. Ouvrir un Terminal.

2. Commande whois, suivie par le nom de domaine.

La capture d'écran suivante montre la sortie d'une requête whois de Kali Linux. Dans le cas de ce domaine spécifique, notez qu'une grande partie des données est masquée en raison de problèmes de confidentialité. De nombreux registres de domaine permettent un contrôle de sécurité qui restreint les informations personnelles identifiables (PII) d'être affiché dans les requêtes whois. Par exemple, dans la section Email Admin : vous disposez d'un lien hypertexte au lieu d'une adresse de messagerie typique :



Il est possible d'afficher une liste complète des différentes options WHOIS en exécutant la commande whois -h.

Collecte d'informations DNS

La collecte d'informations DNS est le processus de localisation des serveurs DNS d'une cible et de leurs enregistrements associés. Un enregistrement DNS contient des informations sur les types de ressources qui existent dans le domaine d'une organisation. Ceux-ci peuvent être comparés à des enregistrements de bases de données qui sont stockés dans les fichiers de zone du domaine.

Je vais commencer par un qui est intégré dans la plupart des systèmes d'exploitation modernes: **nslookup**.

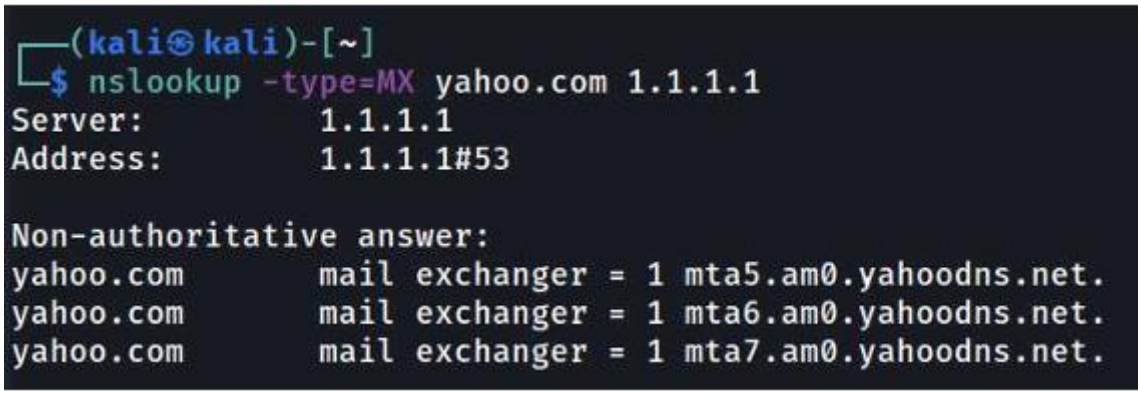
nslookup est un outil qui permet d'obtenir des informations sur un domaine. Cet outil est également trouvé nativement dans les systèmes d'exploitation modernes, et son utilisation est simple. Pour utiliser nslookup, utiliser la syntaxe suivante :

```
nslookup [OPTIONS] [DOMAIN NAME] [NAME SERVER]
```

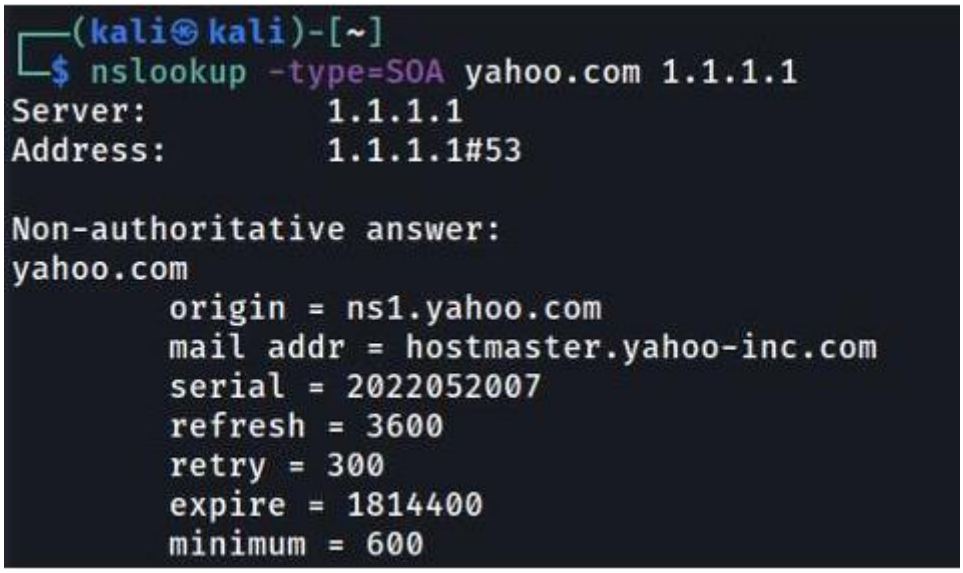
OPTIONS permet de spécifier le type d'enregistrement que je recherche – par exemple, type=A interrogera le domaine pour tous les enregistrements A. DOMAIN NAME est l'endroit où spécifier le nom de domaine cible. Enfin, NAME SERVER permet de spécifier un serveur de noms spécifique que je souhaite interroger. Ceux-ci peuvent varier de serveurs DNS publics tels que 1.1.1.1, 9.9.9.9, et ainsi de suite. Par exemple :

```
$ nslookup -type=MX yahoo.com 1.1.1.1
```

Cela va retourner tous les enregistrements MX pour yahoo.com, comme sur la capture d'écran suivante:



Exécution de la même commande, mais cette fois à la recherche d'enregistrements SOA :



Lorsque j’explore les options de nslookup, je constate que je peux obtenir des informations précieuses sur mon domaine cible.

Des outils tels que sublist3r permettent de découvrir facilement les sous-domaines. Par exemple, en utilisant la commande suivante :

```
$ sublist3r -d example.com
```

Plusieurs autres outils tels que **DNSDumpster, Shodan, Recon-ng** (et bien d’autres) nous permettent d’aller plus loin. Concentrons-nous maintenant sur la collecte d'informations actives.

Collecte d'informations actives