Compte rendu – Kali Linux

LEDIEU Raphaël - DELCOURT Guillaume

Table des matières

Introduction

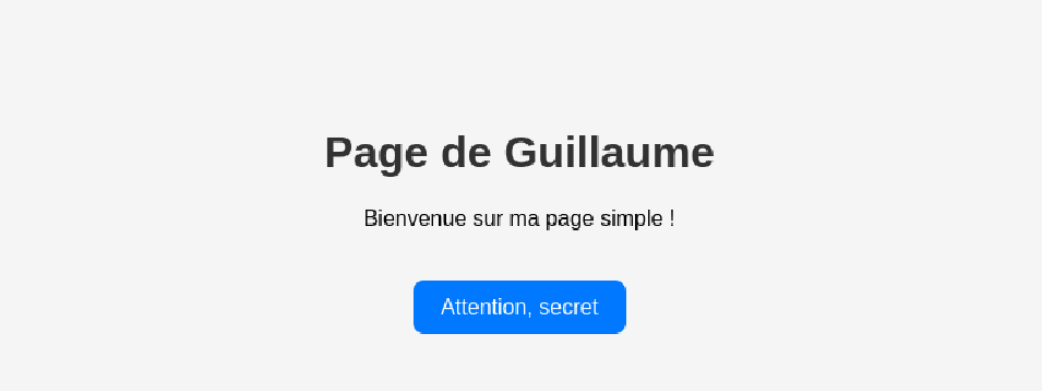
Dans le cadre de l'initiation à la cybersécurité offensive et à l'audit de vulnérabilités, ce projet de travaux pratiques visait à explorer en profondeur l'écosystème Kali Linux, la distribution de référence pour le test d'intrusion. Réalisé en binôme, ce travail a mobilisé notre participation active à toutes les étapes et a abouti à la rédaction de ce compte rendu unique.

L'environnement de travail s'est articulé autour d'une machine sous Kali Linux disposant de sa suite d'outils natifs, déployée au sein d'un laboratoire virtuel isolé. L'objectif principal était d'acquérir une maîtrise pratique de la chaîne d'attaque, allant de la reconnaissance initiale à l'exploitation, tout en comprenant les mécanismes de détection. Les manipulations ont intégré un large spectre d'activités techniques, incluant la découverte de l'environnement, l'utilisation d'outils de scan et d'attaque variés, ainsi que l'analyse minutieuse du trafic réseau. Un accent particulier a été mis sur l'utilisation de **Wireshark** pour capturer et décortiquer les paquets afin de visualiser les signatures des attaques, mais d'autres outils présents dans la distribution ont également été explorés.

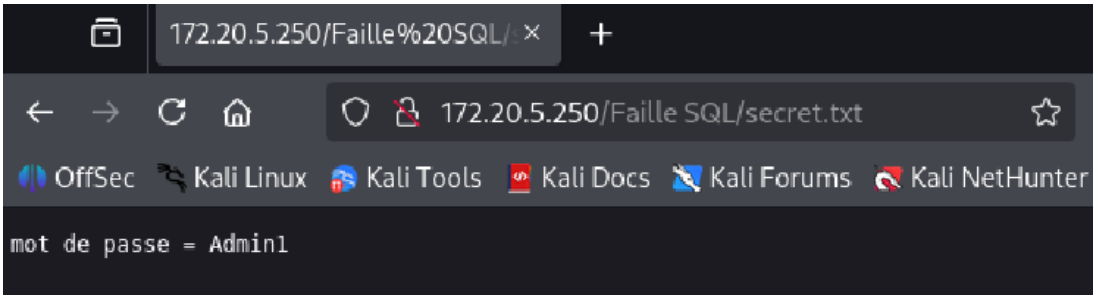
L'ensemble de ces expérimentations a permis de confronter la théorie à la pratique, offrant une vision globale des vecteurs de compromission et des techniques d'analyse forensique nécessaires à la compréhension des incidents de sécurité.

Interception de flux HTTP non sécurisé avec Wireshark

Pour débuter les manipulations, nous avons mis en place un scénario client-serveur afin de tester la confidentialité des échanges sur un réseau non sécurisé.
Guillaume a, en premier lieu, déployé un serveur web **Laragon** sur une machine Ubuntu et a créé une page HTML simple.

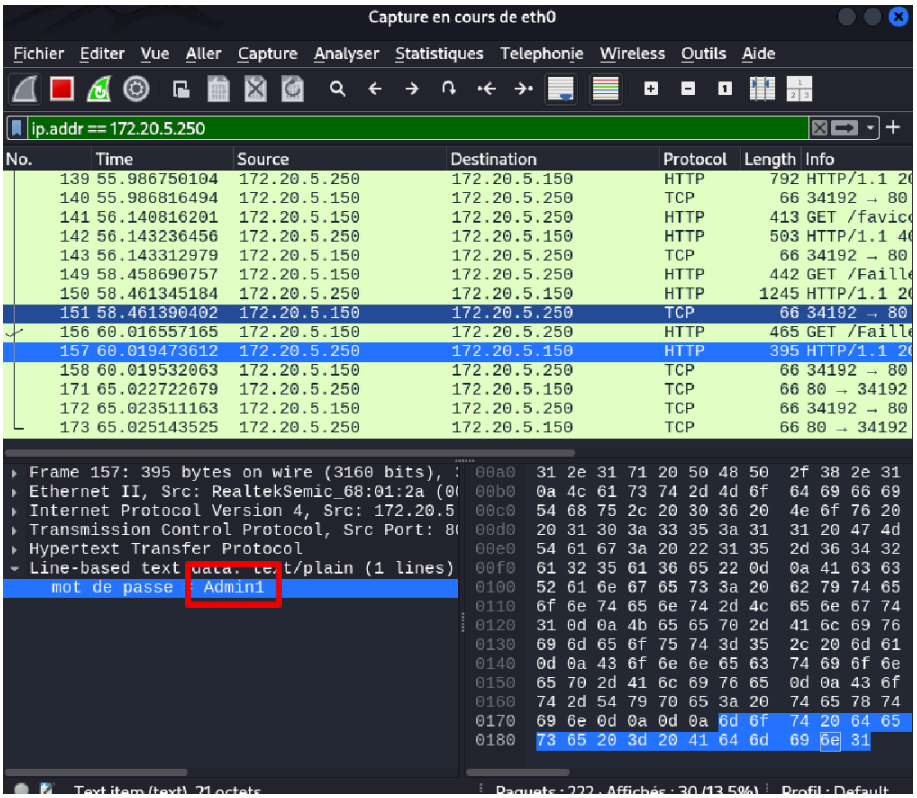


Cette page simple contenait un lien (**bouton bleu**) menant vers un fichier texte (**secret.txt**) contenant un mot de passe **(Admin1).**



Raphaël, depuis Kali Linux, a lancé l'outil d'analyse de paquets **Wireshark** et a alors démarré une capture de trame en filtrant les résultats sur l'adresse IP du serveur **(172.20.5.250)**. Il a pu isoler la requête HTTP au moment précis où l'accès au fichier secret était demandé.

L'analyse du flux TCP a révélé une faille critique, le protocole **HTTP** n'appliquant aucun chiffrement, il a pu lire le contenu de la réponse et récupérer le mot de passe directement en clair dans la trame réseau.



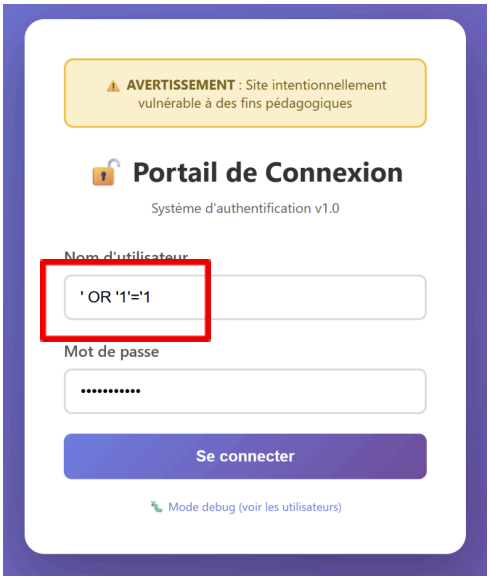
Sécurisation des échanges : Passage au HTTPS

L'analyse de cette capture démontre la dangerosité de l'utilisation du protocole HTTP pour le transfert de données sensibles. Pour pallier cette vulnérabilité, la mise en place du protocole **HTTPS** est indispensable. L'activation d'une couche de chiffrement via **SSL/TLS** garantit la confidentialité des échanges entre le client et le serveur.

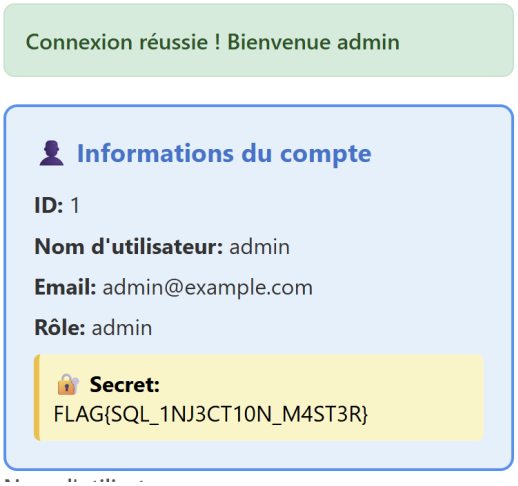
Concrètement, si cette même capture avait été réalisée sur une connexion **HTTPS**, le champ contenant le mot de passe **« Admin1 »** aurait été remplacé par une suite de caractères cryptés et totalement illisibles. Le chiffrement rend ainsi l'écoute passive du réseau **« sniffing »** inefficace pour le vol d'identifiants.

Exploitation de vulnérabilités Web : Injection SQL

Dans un second temps, nous nous sommes attaqués aux failles applicatives web. Pour ce scénario, Raphaël a développé et hébergé une page d'authentification PHP volontairement vulnérable, connectée à une base de données **« SQLite ».** Le formulaire présentait une faille critique de concaténation de variables, permettant de manipuler la requête SQL.



Après avoir vérifié manuellement la présence de la faille en contournant le login avec l'injection classique **« ' OR '1'='1 »,** on arrive à se connecter au compte administrateur.

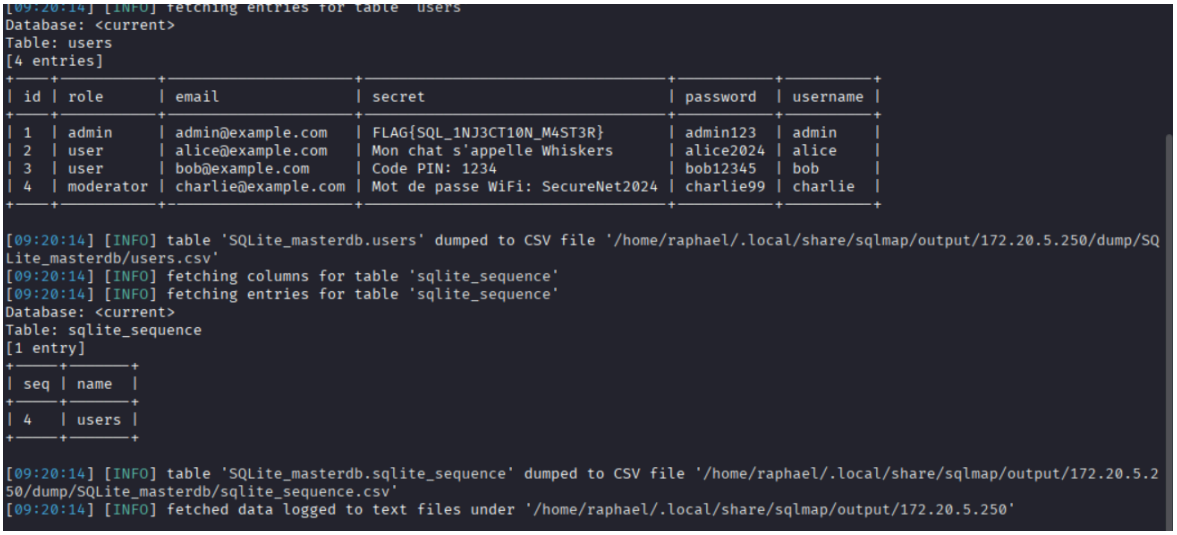


Explication de l’injection

L’injection « **' OR '1'='1** » fonctionne grâce au mot-clé **« OR »,** qui signifie **« ou »** en SQL. Quand on l’ajoute dans un champ de connexion vulnérable, on force la requête SQL à accepter une condition toujours vraie. Par exemple, **« 1 = 1 »** est une affirmation qui ne peut jamais être fausse. Donc même si le mot de passe ou le nom d’utilisateur saisi est incorrect, la partie **« OR '1'='1' »** rend la condition globale vraie. Le système croit alors que les informations de connexion sont valides et laisse entrer l’utilisateur. C’est pour cela que cette injection permet de contourner un login non sécurisé.

Raphaël a invité Guillaume à automatiser l'attaque pour démontrer l'ampleur des dégâts possibles. Depuis Kali Linux, Guillaume a utilisé l'outil **« SQLMap »** en ciblant l'URL du formulaire.

L'outil a injecté une série de requêtes malveillantes qui ont permis d'identifier le moteur de base de données et d'en exfiltrer le contenu. Comme l'attestent les captures d'écran, Guillaume a réussi à récupérer **« dump »** la totalité de la table **« users »** révélant ainsi les identifiants, les mots de passe en clair, ainsi que le drapeau secret **« FLAG{SQL_1NJ3CT10N_M4ST3R} »**. Cette expérience souligne l'importance vitale d'utiliser des requêtes préparées pour neutraliser ce type d'attaque.



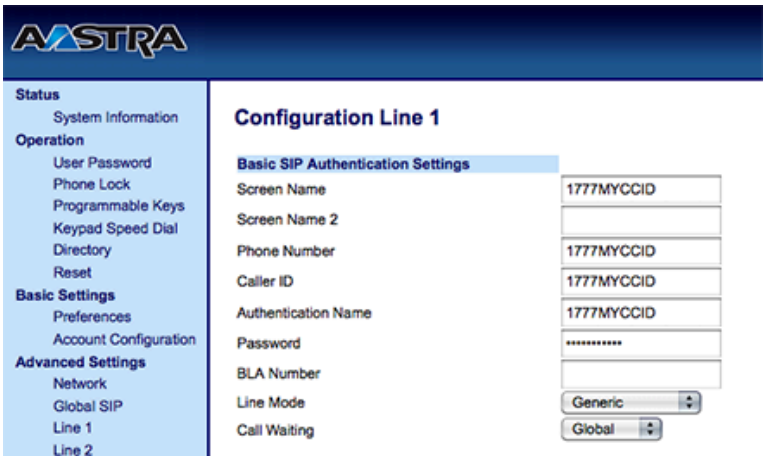
Éviter ce type de faille

Pour éviter ce type de faille, il aurait fallu sécuriser totalement le code dès la conception. La première mesure essentielle consiste à utiliser des **« requêtes préparées »**, qui empêchent l'injection de modifier la logique SQL. Avec ce mécanisme, les données entrées par l'utilisateur sont traitées comme de simples valeurs et ne peuvent plus être interprétées comme du code. Il est également indispensable de valider et filtrer les entrées, de ne jamais concatener directement des variables dans une requête, et de limiter les droits de l'utilisateur de la base de données au strict minimum. Si ces bonnes pratiques avaient été appliquées, ni l'injection manuelle, ni l'automatisation via SQLMap n'auraient pu fonctionner. Cette démonstration montre clairement que, sans une sécurisation correcte, une simple page de login peut compromettre l'intégralité d'un système.

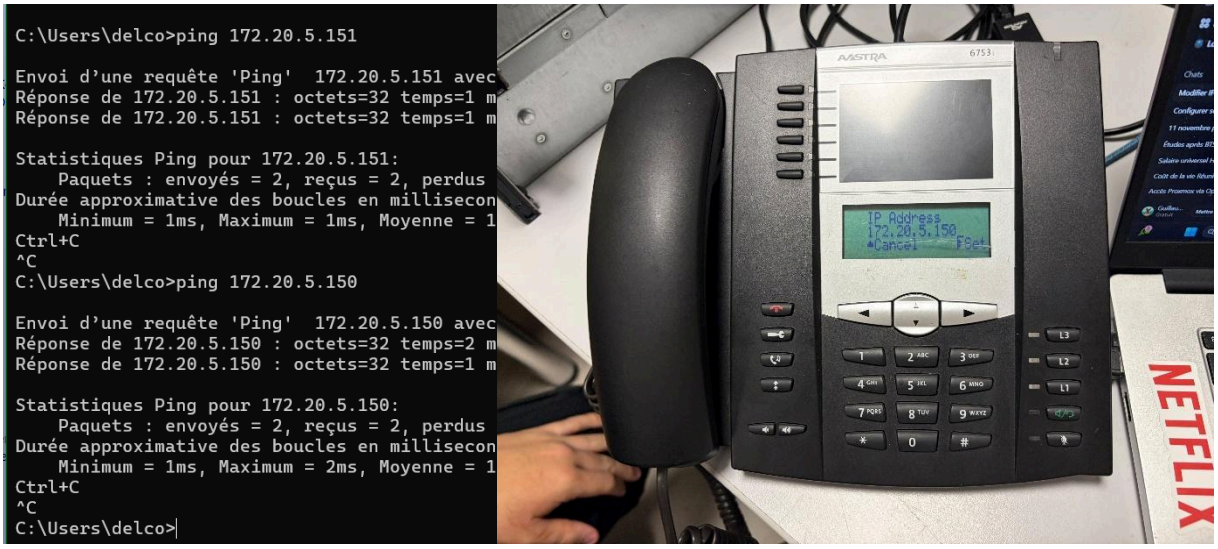
Interception et écoute de communications VoIP

Après avoir exploré les failles web, nous nous sommes tournés vers l'analyse de la voix sur le protocole internet (VoIP). L'objectif était de démontrer qu'une conversation téléphonique transitant sur un réseau local peut être interceptée et réécoutée si elle n'est pas chiffrée.

Dans un premier temps, nous avons connecté deux téléphones IP physiques sur un switch. Nous leur avons attribué des adresses IP statiques et des numéros de poste via leur interface de configuration web respective.



Pour valider la connectivité réseau entre les équipements avant de passer à la configuration du serveur, nous avons effectué des tests de ping.



Pour gérer les appels, Guillaume a installé un serveur **Asterisk** sur un serveur Ubuntu. Initialement, trouvant Asterisk vieillissant et complexe, Raphaël a tenté de rechercher et déployer des solutions plus modernes et plus simples. Cependant, face aux difficultés d'intégration rencontrées avec ces alternatives, nous avons pris la décision technique de rester sur Asterisk, qui demeure la référence industrielle stable.

Guillaume a, par la suite, configuré les fichiers clés pour déclarer nos utilisateurs (les téléphones) et le plan de numérotation :

```
 GNU nano 7.2                                    pjsip.conf
[global]
type=global
user_agent=AsteriskPBX

[transport-udp]
type=transport
protocol=udp
bind=0.0.0.0

; ---- téléphone 150 ----
[150]
type=endpoint
context=internal
disallow=all
allow=ulaw,alaw
auth=auth150
aors=150
callerid="Poste 150" <150>

[auth150]
type=auth
auth_type=userpass
username=150
password=150password
```

« pjsip.conf » : Déclaration des "endpoints" (les téléphones 150 et 151) et de leurs mots de passe. Ce fichier permet également d'identifier le téléphone sur le serveur asterisk.

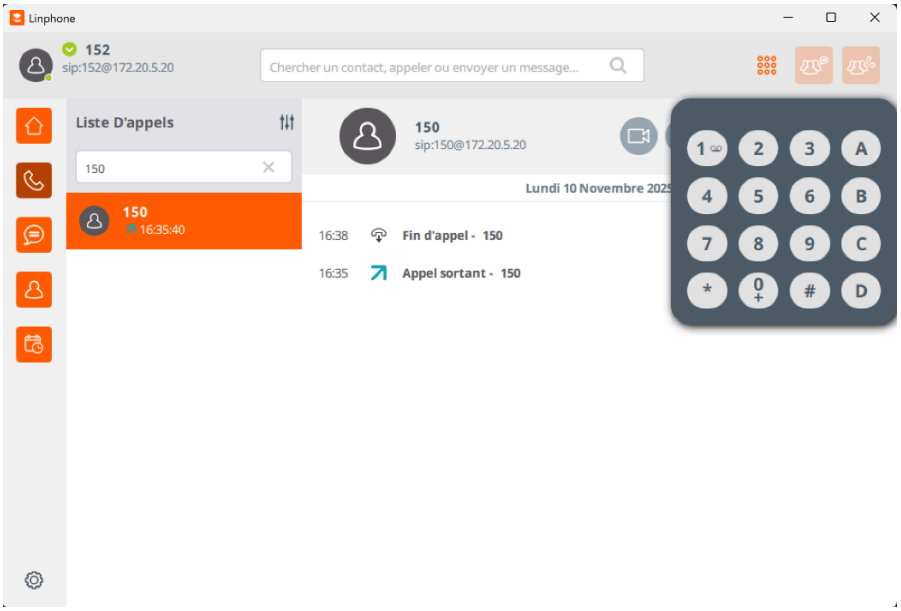
```
root@ubuntu-VirtualBox: /etc/asterisk
GNU nano 7.2                                    extensions.conf
       same => n,Playback(vm-from)
       same => n,SayDigits(${CALLERID(ani)})
       same => n,Wait(1.25)
       same => n,SayDigits(${CALLERID(ani)})   ; playback again in case of missed digit
       same => n,Return()

; For more information on applications, just type "core show applications" at your
; friendly Asterisk CLI prompt.
;
; "core show application <command>" will show details of how you
; use that particular application in this file, the dial plan.
; "core show functions" will list all dialplan functions
; "core show function <COMMAND>" will show you more information about
; one function. Remember that function names are UPPER CASE.

[internal]
exten => 150,1,Dial(PJSIP/150)
exten => 151,1,Dial(PJSIP/151)


exten => 6°°,1,Answer()
same => n,Playback(demo-echotest)
same => n,Echo()
same => n,Hangup()
```

« extensions.conf » : Configuration de la logique d'appel (qui peut appeler qui).

Une fois les fichiers de configuration sauvegardés et le service redémarré, nous avons validé le fonctionnement par un premier appel test entre les téléphones physiques.

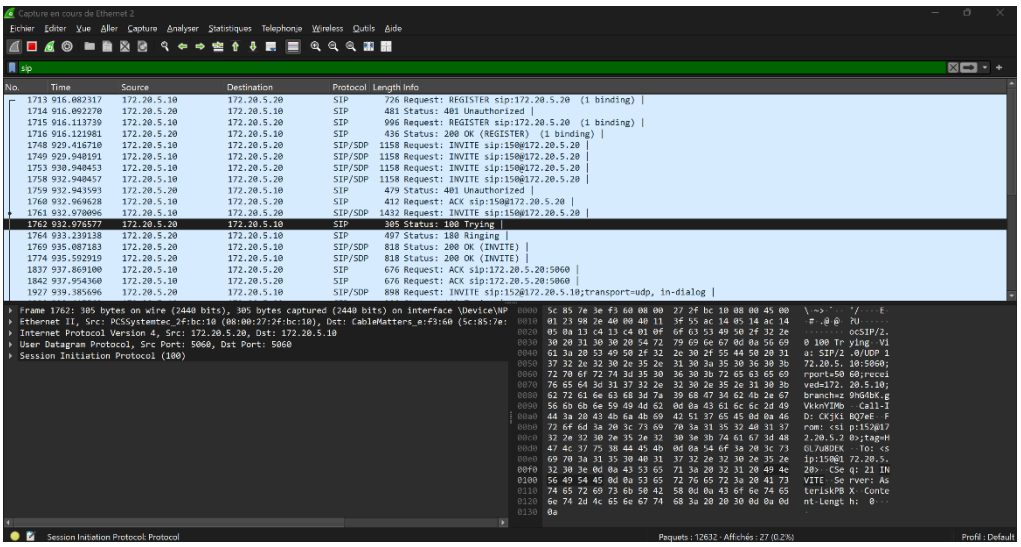
Pour réaliser l'interception du trafic, j'ai basculé sur un client logiciel nommé **« Linphone »,** installé sur ma machine Kali Linux. Guillaume a crée un nouveau numéro (152) pour que Raphaël puisse rejoindre le réseau.

Afin de simuler un environnement non sécurisé et garantir la réussite de l'interception, Raphaël a volontairement désactivé tous les paramètres de chiffrement **« SRTP »** et de sécurité réseau dans les options de Linphone. L'objectif était de forcer les données à transiter "en clair".

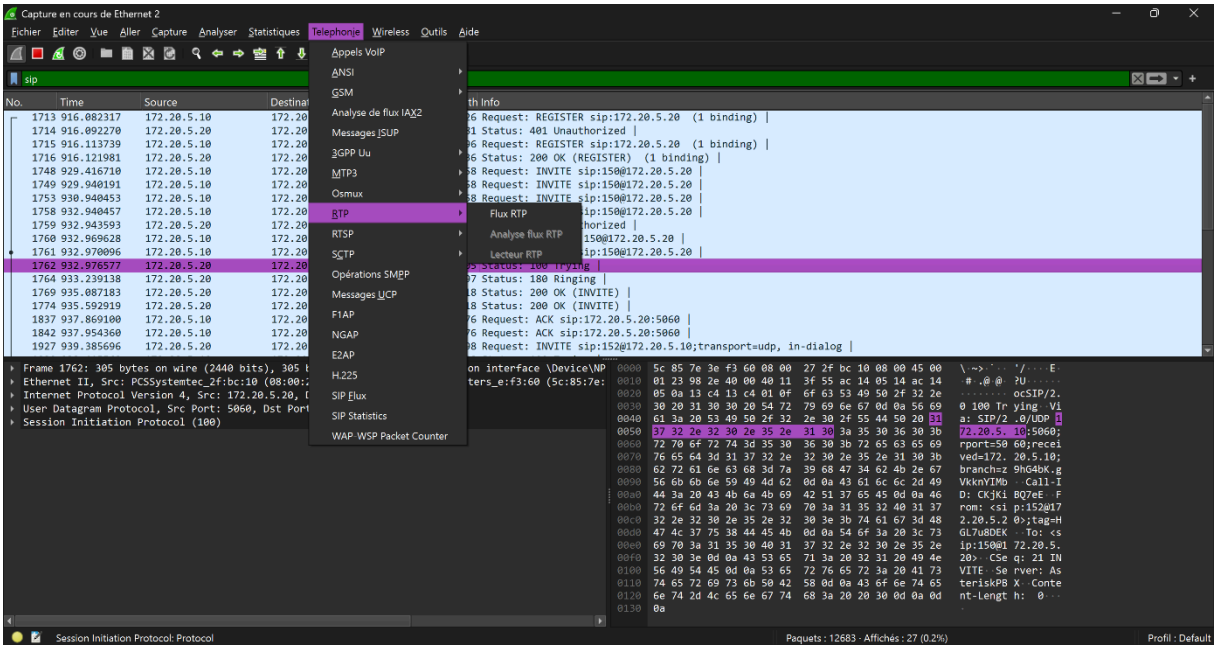


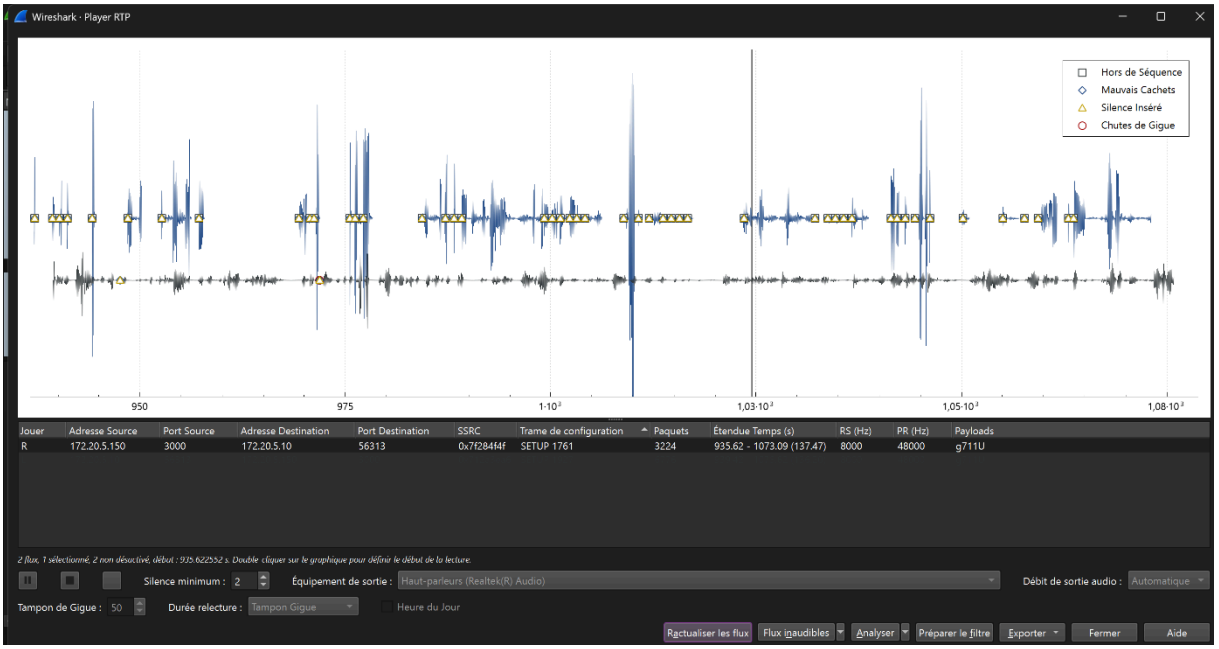
Pendant que nous réalisions un appel de test, Raphaël a lancé **Wireshark** pour capturer tout le trafic passant par mon interface réseau.

En filtrant sur le protocole sip, nous voyons clairement les étapes d'établissement de l'appel (INVITE, RINGING, OK, ACK). Toutes ces métadonnées (qui appelle qui, numéros, IP) sont lisibles en clair.



Le plus critique réside dans le protocole de transport de la voix (RTP). Wireshark dispose d'une fonctionnalité puissante **« Téléphonie > RTP > Analyse de flux »** qui permet non seulement de visualiser le graphique de la conversation, mais surtout de ré-écouter l'appel.





(La capture de trame est disponible en PJ)

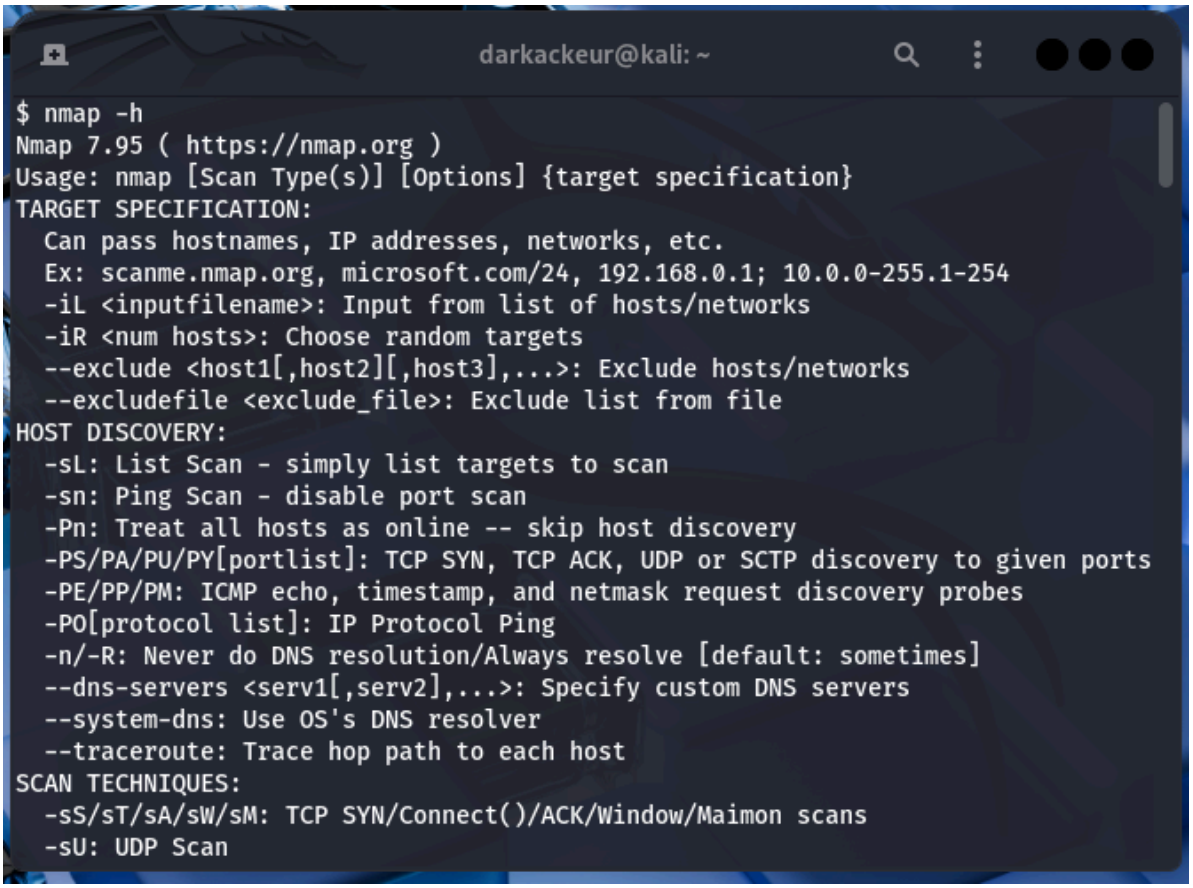
Cette manipulation démontre clairement qu'un réseau VoIP standard, dépourvu de mécanismes de sécurité, offre une transparence totale des communications. Toute personne connectée au même réseau peut intercepter et écouter les appels sans difficulté, ce qui constitue un risque majeur pour la confidentialité. Pour sécuriser efficacement une telle infrastructure, il est indispensable d'intégrer des protocoles dédiés **SIP-TLS**, afin de chiffrer toute la signalisation et masquer les informations d'appel, ainsi que **SRTP**, pour protéger le flux audio en le rendant inaudible lors d'une capture réseau. Avec ces mesures, l'interception deviendrait inutile, garantissant la confidentialité et l'intégrité des communications **VoIP**.

Découverte réseau grâce à NMAP

Pour poursuivre notre découverte des différents outils proposés dans Kali Linux, nous avons utilisé **NMAP** afin de comprendre son fonctionnement et d'observer comment il permet d'effectuer une découverte réseau de manière simple et structurée.

Nmap (Network Mapper) est un outil d'analyse réseau qui permet de détecter les machines actives, d'identifier les ports ouverts ainsi que les services associés. Il est principalement utilisé pour réaliser des diagnostics réseau et mieux comprendre la surface d'exposition d'un système.

Nmap s'accompagne d'une pléthore de commandes et d'options, ce qui lui permet de réaliser aussi bien des scans simples que des analyses réseau beaucoup plus détaillées selon les besoins.

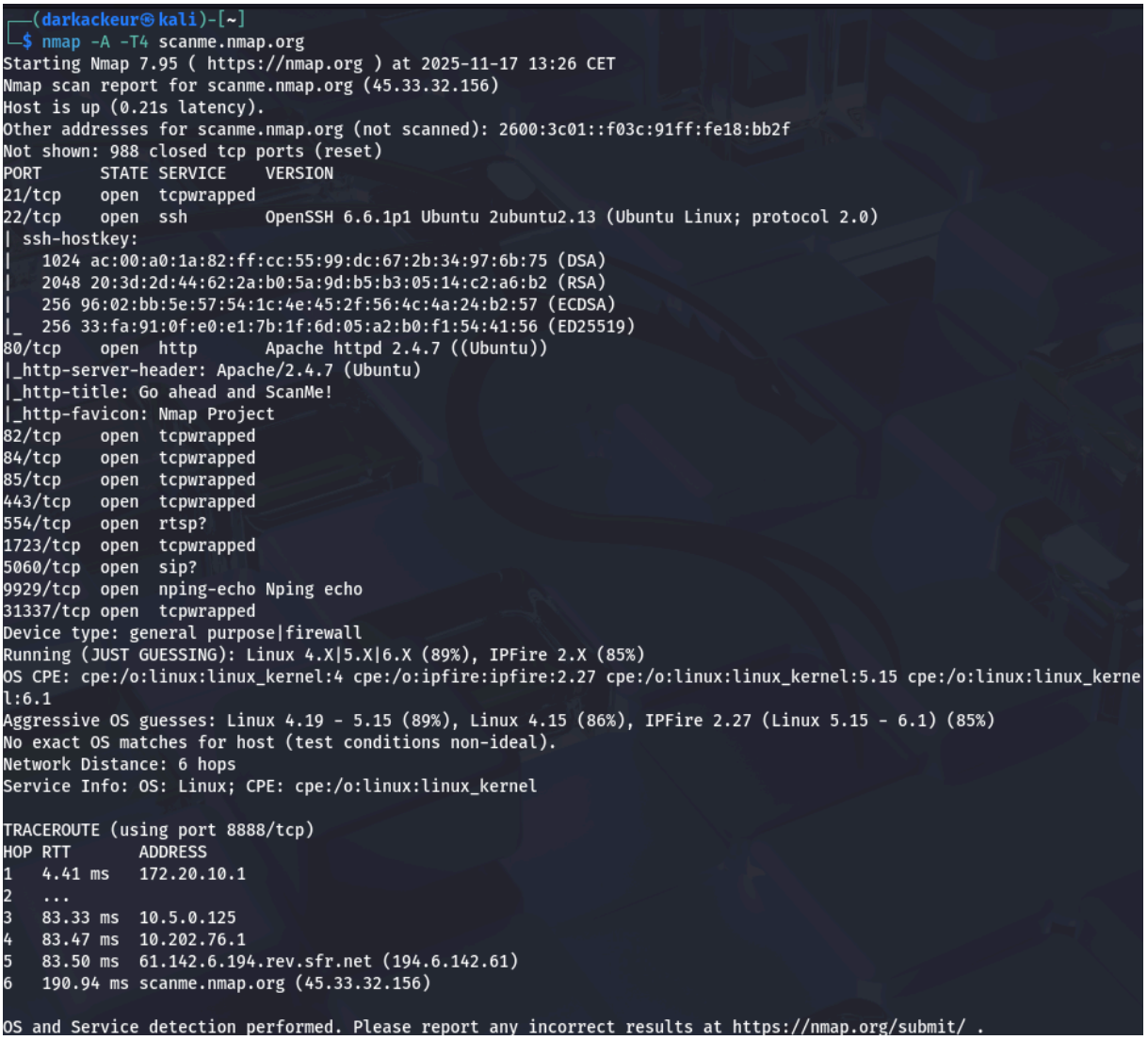


Dans notre exercice, Nmap nous mettait à disposition un domaine (ou une adresse cible) à analyser. Nous avons donc effectué un scan en utilisant les options -A et -T4.

Scan pédagogique

L'option -A permet d'activer une analyse avancée incluant la détection de services, la tentative d'identification du système d'exploitation et la recherche de scripts pertinents.

L'option -T4, quant à elle, ajuste la vitesse du scan sur un mode « agressif » mais toujours raisonnable, permettant d'accélérer la collecte d'informations tout en maintenant une bonne fiabilité.

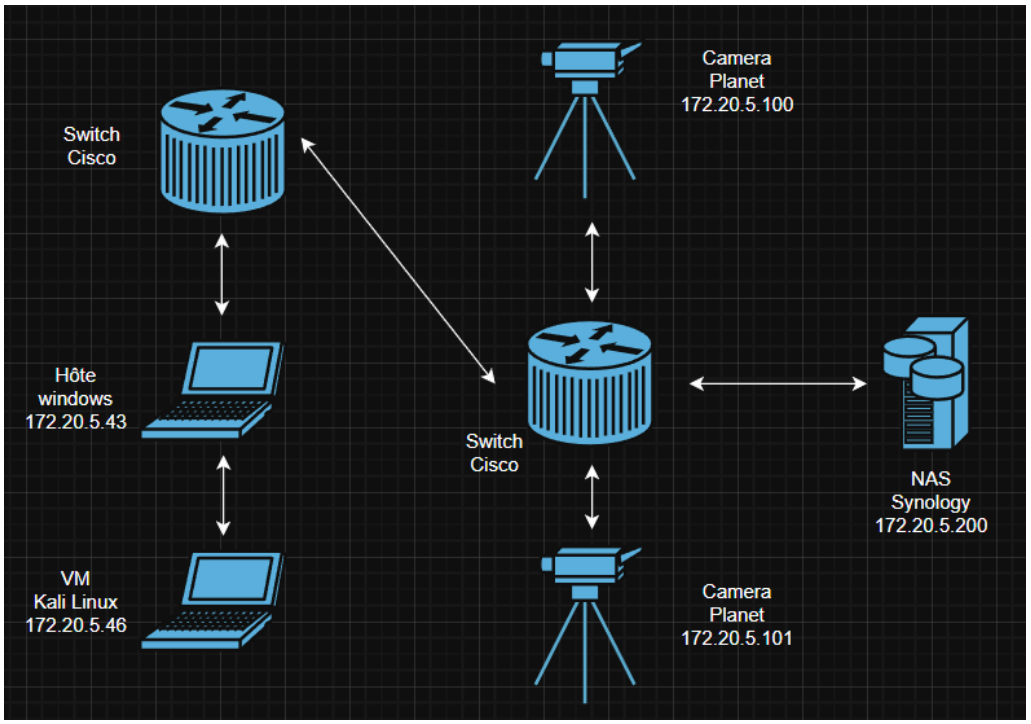


```
┌──(darkackeur㉿kali)-[~]
└─$ nmap -A -T4 scanme.nmap.org
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-17 13:26 CET
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.21s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 988 closed tcp ports (reset)
PORT      STATE SERVICE    VERSION
21/tcp    open  tcpwrapped
22/tcp    open  ssh        OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 ac:00:a0:1a:82:ff:cc:55:99:dc:67:2b:34:97:6b:75 (DSA)
|   2048 20:3d:2d:44:62:2a:b0:5a:9d:b5:b3:05:14:c2:a6:b2 (RSA)
|   256 96:02:bb:5e:57:54:1c:4e:45:2f:56:4c:4a:24:b2:57 (ECDSA)
|_  256 33:fa:91:0f:e0:e1:7b:1f:6d:05:a2:b0:f1:54:41:56 (ED25519)
80/tcp    open  http       Apache httpd 2.4.7 ((Ubuntu))
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Go ahead and ScanMe!
|_http-favicon: Nmap Project
82/tcp    open  tcpwrapped
84/tcp    open  tcpwrapped
85/tcp    open  tcpwrapped
443/tcp   open  tcpwrapped
554/tcp   open  rtsp?
1723/tcp  open  tcpwrapped
5060/tcp  open  sip?
9929/tcp  open  nping-echo Nping echo
31337/tcp open  tcpwrapped
Device type: general purpose|firewall
Running (JUST GUESSING): Linux 4.X|5.X|6.X (89%), IPFire 2.X (85%)
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:ipfire:ipfire:2.27 cpe:/o:linux:linux_kernel:5.15 cpe:/o:linux:linux_kerne
l:6.1
Aggressive OS guesses: Linux 4.19 - 5.15 (89%), Linux 4.15 (86%), IPFire 2.27 (Linux 5.15 - 6.1) (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 6 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 8888/tcp)
HOP RTT       ADDRESS
1   4.41 ms   172.20.10.1
2   ...
3   83.33 ms  10.5.0.125
4   83.47 ms  10.202.76.1
5   83.50 ms  61.142.6.194.rev.sfr.net (194.6.142.61)
6   190.94 ms scanme.nmap.org (45.33.32.156)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
```

Grâce à ce scan, on peut voir que l'hôte est actif, les différents ports qui sont ouverts et les services qui leurs sont associés, le système d'exploitation de la machine, la route qu'a prise le NMAP. Ce scan de test est mis en place par NMAP afin d'éclairer le mode de fonctionnement de l'outil.

Scan du réseau local (LAN)

Voici un schéma de notre réseau local :



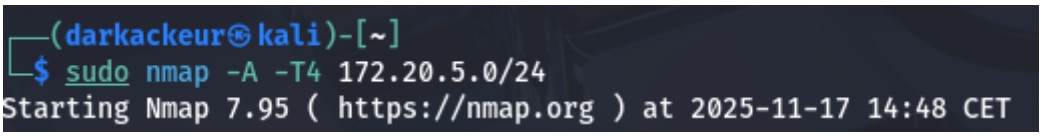
Une fois l'outil pris en main une première fois, nous avons pris la décision de scanner l'entièreté de notre réseau local. Nous avons donc procédé à un scan simple de tout notre réseau, c'est-à-dire **172.20.5.0/24**.

```
┌──(darkackeur㉿kali)-[~]
└─$ nmap -Pn 172.20.5.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-17 14:44 CET
Nmap scan report for 172.20.5.43
Host is up (0.00060s latency).
All 1000 scanned ports on 172.20.5.43 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 00:E0:4C:68:01:2A (Realtek Semiconductor)

Nmap scan report for 172.20.5.100
Host is up (0.0012s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
554/tcp  open  rtsp
6000/tcp open  X11
MAC Address: 00:30:4F:BC:FC:6A (Planet Technology)

Nmap scan report for 172.20.5.101
Host is up (0.0015s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
554/tcp  open  rtsp
6000/tcp open  X11
MAC Address: 00:30:4F:BC:FC:55 (Planet Technology)
```

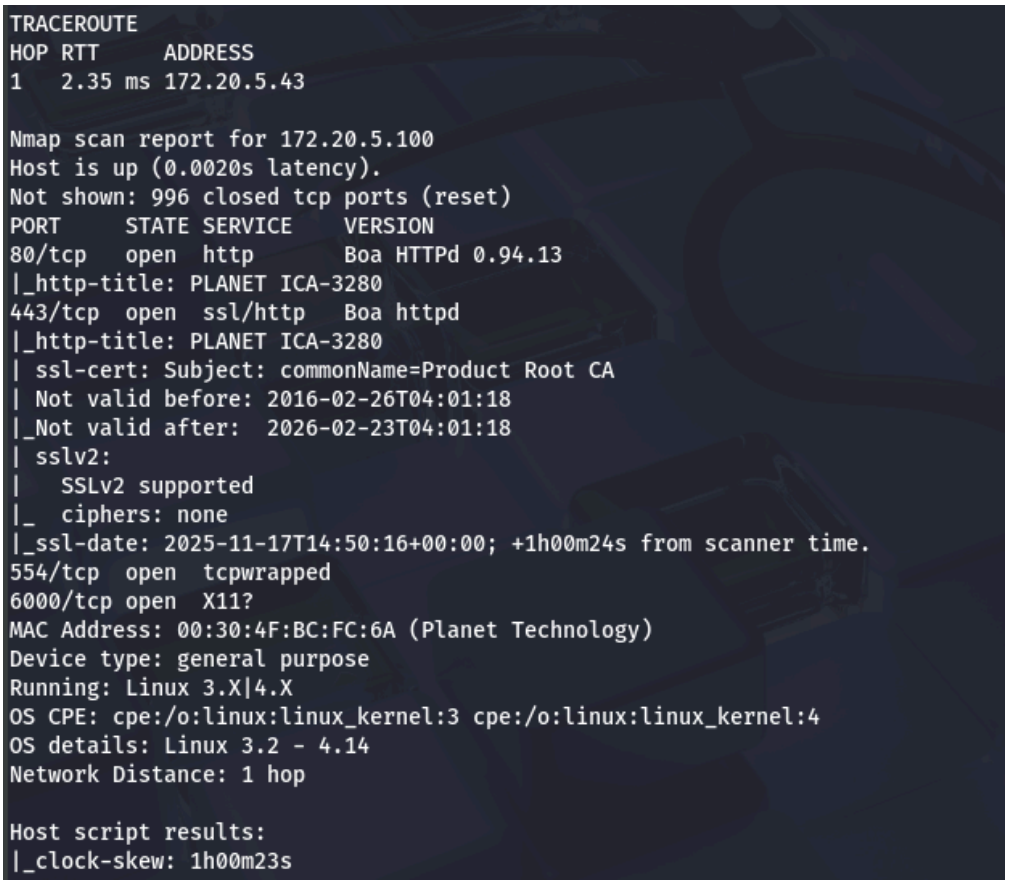
Cela a permis à NMAP de scanner l'entièreté des machines connectées à notre réseau. Le scan lancé était un scan simple, c'est-à-dire, un scan qui considère l'hôte comme actif peu importe si celui-ci répond aux pings. On a donc pu scanner les 3 machines présentes sur le réseau : 2 caméras de surveillance et 1 NAS (172.20.5.200).

Ce scan est donc simple et très rapide, il a pris 32 secondes à NMAP de recueillir toutes les informations présentes sur la capture ci-dessus.

Afin de recueillir le plus d'informations sur les appareils de notre réseau, nous avons lancé un scan du réseau en -A, donc agressif avec la commande suivante :



Les résultats obtenus après cette commande sont bien plus complets que le scan précédent.



Voici un exemple de scan agressif sur l'une des caméras de notre réseau. On voit donc encore une fois les ports ouverts, l'OS ou encore la route.

Réduire les risques d'être scanné par NMAP

Nmap apparaît donc comme un outil particulièrement puissant, capable de révéler des informations détaillées sur une machine ou un réseau, et de mettre en évidence de potentielles failles liées à des ports ouverts, des services mal configurés ou une surface d'exposition trop large.

Pour réduire les risques liés aux scans réseau, il est possible d'appliquer deux mesures simples mais efficaces. Premièrement, on peut bloquer ou filtrer **l'ICMP**, ce qui empêche les réponses au ping. Comme NMAP utilise souvent cette étape pour vérifier si une machine est active, le fait de ne plus répondre rend la détection initiale beaucoup plus difficile. Cependant, comme on l'a vu, certaines options (dont **-Pn**) permettent de ne pas tenir compte du ping et de scanner tout de même les ports.

Deuxièmement, on peut configurer un pare-feu pour fermer l'ensemble des ports non nécessaires et placer les règles en mode DROP. Dans ce cas, la machine ne renvoie aucune information aux paquets entrants, ce qui crée une absence totale de réponse. Pour un outil comme NMAP, il devient alors compliqué de distinguer un port fermé d'une machine inexistante, ce qui réduit significativement les informations qu'un scan peut récupérer. Ces deux approches n'éliminent pas totalement la détection, mais elles diminuent fortement l'exposition et la quantité de données accessibles à un observateur externe.

BruteForce de mot de passe de borne WiFi

Dans l'optique d'une attaque informatique, la compromission d'un réseau Wi-Fi peut représenter une faille critique de sécurité. Un mot de passe faible, un protocole mal configuré ou une mauvaise hygiène de sécurité peuvent permettre à un attaquant d'obtenir un accès réseau non autorisé. Cet accès ouvre potentiellement la porte à des interceptions de trafic, à des attaques internes ou à une compromission plus large du système.

Nous avions pour objectif de mettre en place une borne Wi-Fi de test dans un environnement isolé et contrôlé, puis d'en évaluer la sécurité en tentant de retrouver son mot de passe à l'aide de la suite **Aircrack-ng**.

Nous avons alors commencé par mettre en place une borne WiFi netgear. Nous avons configuré son interface réseau.

LAN Information - IPv4

IP Address	172.20.5.90
Subnet Mask	255.255.255.0
Gateway	172.20.5.254
Primary DNS	8.8.8.8
Secondary DNS	1.1.1.1
DHCP Client	Disable
Spanning Tree Protocol(STP)	Disable

Puis, nous avons ajouté un mot de passe, relativement simple. En effet, celui-ci n'est pas censé être confidentiel.

Wireless Security

Security Mode	WPA2-Personal	
Encryption	AES	
Passphrase	Azerty08!	
Group Key Update Interval	3600	(30~3600; 0:Disable)

Au cours du TP, nous avons rencontré une contrainte matérielle majeure : la carte Wi-Fi utilisée n'était pas compatible avec le mode "monitor", parfois également appelé mode moniteur. Ce mode spécifique est indispensable pour les opérations de découverte et de capture de trames réseau, étapes préalables à toute analyse de sécurité d'un réseau Wi-Fi.

En l'absence de support du mode moniteur, il est impossible :

- D'écouter le trafic environnant
- De détecter correctement les points d'accès et les clients associés
- et surtout de capturer les échanges nécessaires à l'évaluation du mécanisme d'authentification (comme un handshake WPA/WPA2).

En conséquence, nous avons dû renoncer à la phase de récupération et d'analyse du mot de passe via Aircrack-ng, puisque l'environnement matériel ne permettait pas de mener à bien l'audit prévu.

Cette contrainte met en évidence l'importance du choix du matériel dans les opérations de test de sécurité Wi-Fi, et notamment la nécessité d'utiliser des cartes compatibles avec le mode moniteur et l'injection de paquets lors d'exercices de ce type.

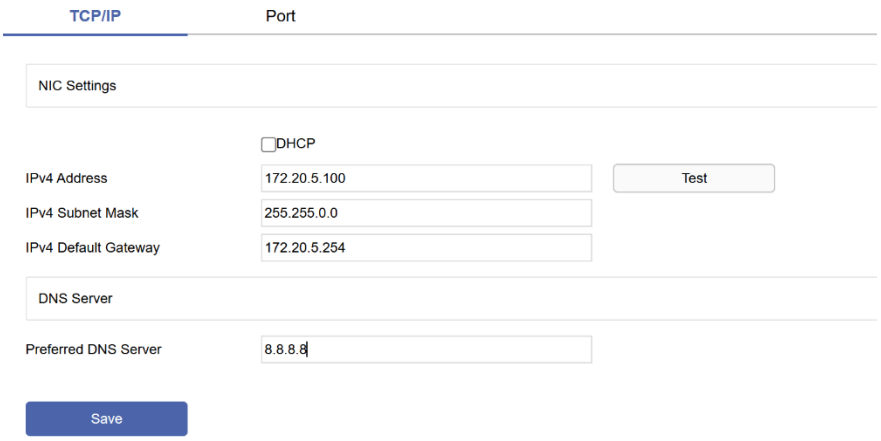
Configuration et audit de sécurité de caméras IP

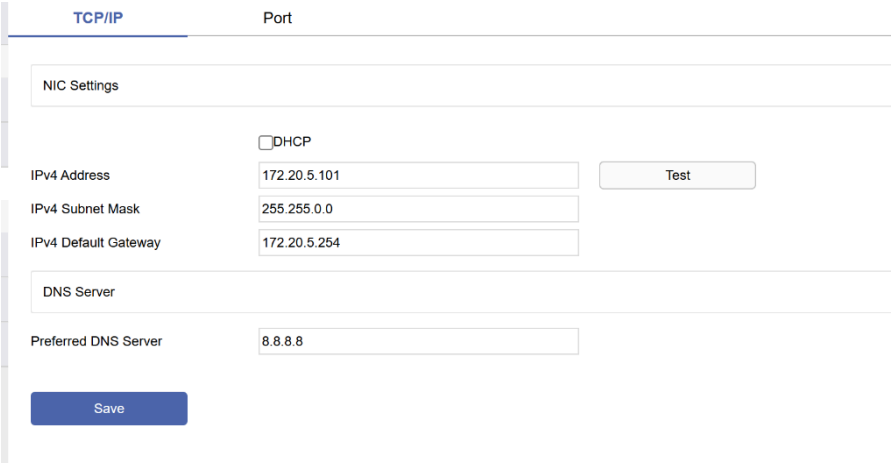
Pour la dernière phase de ce laboratoire, nous nous sommes intéressés à la sécurité physique et à l'Internet des Objets **« IoT »**. Nous avons redéployé deux caméras de surveillance dans la salle afin de les configurer et de tester leur résistance face à une tentative d'intrusion.

Nous avons connecté les deux caméras sur le switch du réseau local. Afin de faciliter leur gestion, nous avons opté pour une configuration en adressage statique. Nous nous sommes réparti la tâche :

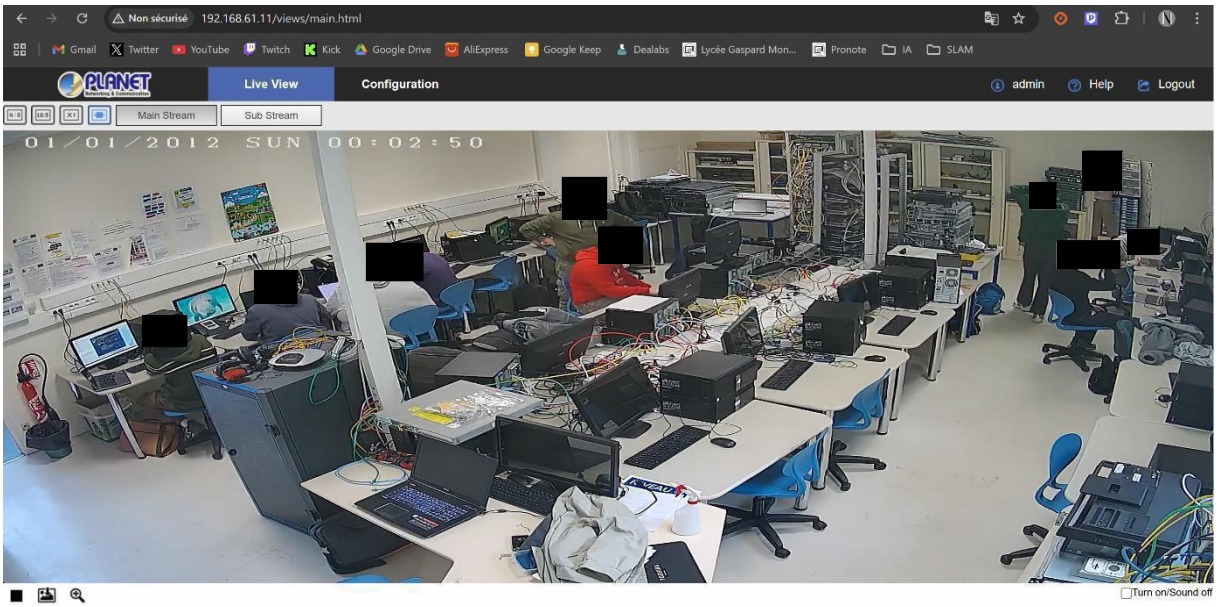
Caméra 1 (Configurée par Guillaume) : IP attribuée 172.20.5.100 / Masque 255.255.0.0.

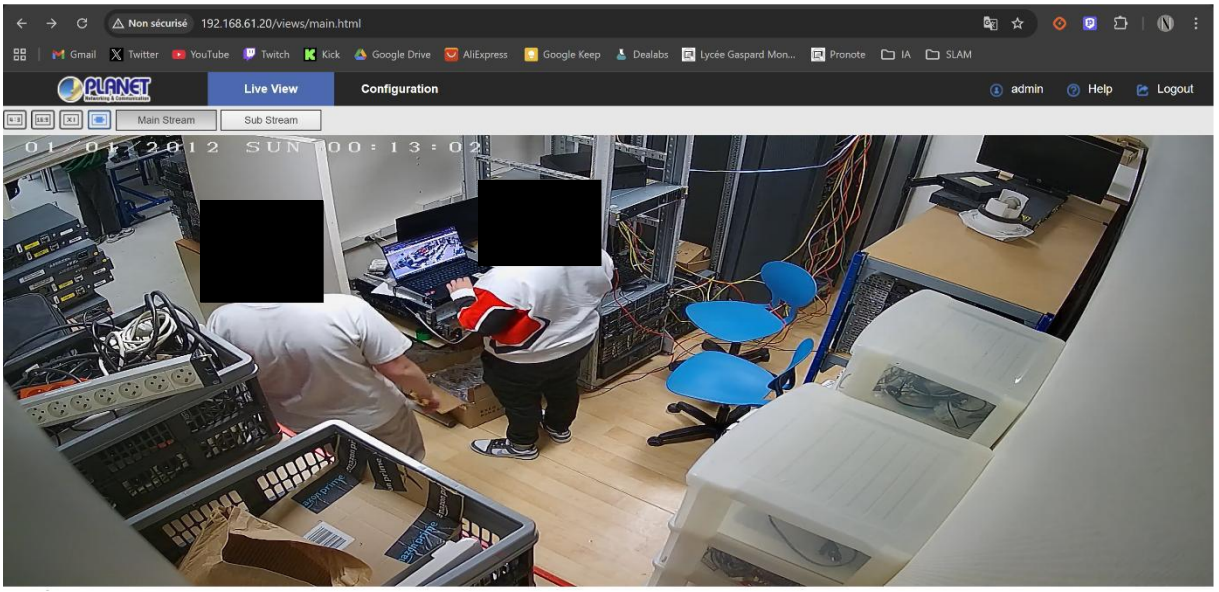
Caméra 2 (Configurée par Raphaël) : IP attribuée 172.20.5.101 / Masque 255.255.0.0.





Nous avons vérifié l'accès aux interfaces d'administration web et visualisé les flux vidéo en direct pour confirmer le bon fonctionnement.





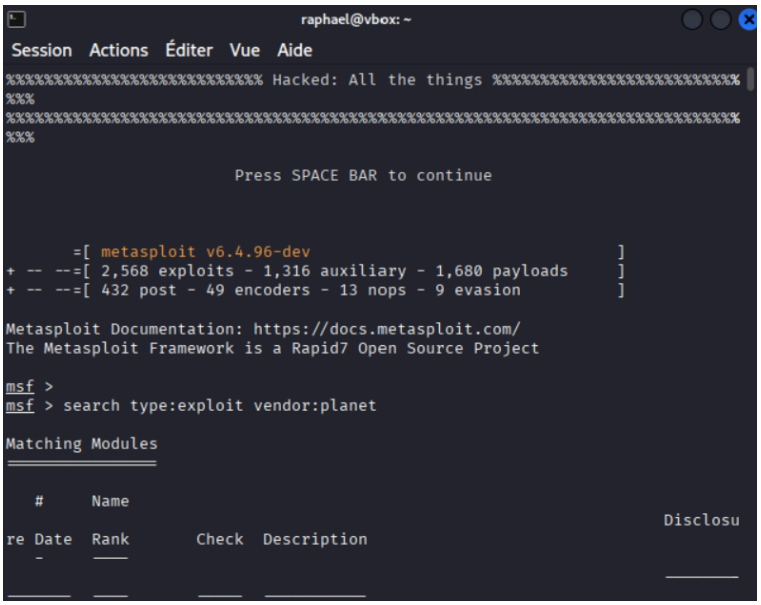
Une fois les caméras opérationnelles, nous avons tenté de compromettre leur sécurité depuis notre station Kali Linux. L'objectif était de voir s'il était possible d'accéder au flux vidéo ou à l'administration sans posséder les identifiants légitimes.

Nous avons utilisé plusieurs outils standards de l'arsenal offensif :

« Nmap » : Pour scanner les ports ouverts et tenter d'identifier les versions des services (HTTP, RTSP, ONVIF).

```
┌──(raphael㉿vbox)-[~]
└─$ nmap -sV -O -p- 172.20.5.100
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-20 12:28 CET
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabl
ed. Try using --system-dns or specify valid servers with --dns-servers
Note: Host seems down. If it is really up, but blocking our ping probes, try
-Pn
Nmap done: 1 IP address (0 hosts up) scanned in 1.82 seconds

┌──(raphael㉿vbox)-[~]
└─$
```

« Metasploit » : Pour rechercher des exploits connus liés au modèle de la caméra.



Contrairement aux exercices précédents (Web et VoIP), toutes nos tentatives d'intrusion sur le système des caméras ont échoué. Nous n'avons pas réussi à pénétrer l'infrastructure ni à intercepter le flux vidéo sans authentification.

Plusieurs éléments expliquent cette résistance :

Nmap n'a révélé aucune ouverture exploitable.
Lors du scan, seuls quelques services standards ont été détectés, et aucun ne permettait un accès direct au flux ou une tentative d'exploitation. Les ports exposés ne présentaient aucune vulnérabilité évidente.

Metasploit (MSF) n'a rien trouvé non plus.
Les modules testés ne correspondaient qu'à d'autres modèles ou d'autres appareils du réseau, mais aucun n'était applicable à cette caméra. Aucune CVE connue ne ciblait spécifiquement la version du firmware détecté.

Mot de passe fort obligatoire dès l'installation.
Contrairement aux anciens objets connectés (admin/admin), la caméra force la création d'un mot de passe robuste, ce qui élimine immédiatement une des attaques les plus courantes.

Cette étape montre qu'un matériel assez récent, correctement configuré et protégé par des mots de passe robustes, offre un niveau de sécurité par défaut largement suffisant contre les attaques basiques. Malgré nos scans (Nmap) et nos tests d'exploitation (MSF), aucune faiblesse exploitable n'a été trouvée.

Conclusion :

Ce laboratoire nous a permis d'explorer un large panel d'outils et de techniques de cybersécurité, couvrant aussi bien l'analyse réseau que l'exploitation de failles applicatives, l'interception de communications VoIP et l'audit de périphériques IoT. L'objectif était de toucher un peu à tout, et c'est exactement ce que nous avons réussi à faire : sniffing réseau, attaques web, utilisation avancée de Wireshark, exploitation SQL avec SQLMap, configuration d'un serveur VoIP Asterisk, et tests d'intrusion avec Nmap et Metasploit.

En complément de ces manipulations réelles, nous avons également travaillé sur la plateforme Root-Me, un environnement reconnu et très complet pour l'apprentissage des techniques d'intrusion. Bien que certains challenges soient particulièrement complexes, cette plateforme nous a permis de développer davantage nos capacités d'analyse, de raisonnement et de compréhension des mécanismes d'exploitation. Elle représente un excellent entraînement pour progresser dans le domaine.

Finalement, ces expériences montrent clairement la différence entre un système mal configuré facile à compromettre et du matériel récent correctement sécurisé, comme les caméras IP utilisées, qui n'ont présenté aucune faille exploitable. Ce projet nous a offert une vision concrète du fonctionnement d'un test d'intrusion et a renforcé notre compréhension des bonnes pratiques nécessaires pour sécuriser efficacement un environnement informatique.