Vulnérabilité des objets connectés

Encadrant : **S. Tixeuil,**

Etudiants : **C. Boisson, H. Chibani, E. Giang, L. Merlin**

Table des matières

1 Cahier des charges

Durant ces dernières années, le monde a connu une importante révolution numérique grâce au développement de l'informatique et du réseau internet. Ces innovations, omniprésentes de nos jours, ont permis le développement de l'Internet of Things (IoT). En effet, le nombre de personnes achetant des objets connectés ne cesse de croître. Ces objets sont de plus en plus variés (lampes, thermomètres, enceintes), et permettent de bénéficier de nombreux services. Les particuliers qui achètent ces objets veulent en effet pouvoir être informés et pouvoir contrôler leur maison à distance à n'importe quel moment, peu importe le lieu où ils se trouvent.

Néanmoins, une question se pose sur la fiabilité de ces objets connectés. En effet, la sécurité lors de l'utilisation de ces objets est l'un des enjeux primordiaux, car ceux-ci font l'objet d'attaques régulières qui peuvent occasionner, par exemple, des fuites de données personnelles. Il serait donc important pour les clients, de pouvoir prendre connaissance des vulnérabilités de leurs objets connectés. C'est pourquoi, le but de notre projet est de réaliser une sonde qui écoutera le réseau domestique dans lequel elle est installée. Cette sonde, à partir du trafic capturé, va se charger de répertorier les objets identifiés du réseau domestique et va, à partir de failles connues, identifier les différentes vulnérabilités de ces objets. Par conséquent, le but du projet n'est pas de contrer ces vulnérabilités trouvées, mais uniquement de les présenter au client.

Ainsi, un client pourra accéder à la liste des objets connectés de son foyer et connaître leurs vulnérabilités face à de possibles attaques. Les clients seront donc bien informés et pourront agir en conséquence.

Nous disposons pour ce projet d'un délai d'environ cinq mois, puisqu'il est à terminer pour fin mai 2021.

2 Plan de développement

Ce projet devra être réalisé sur cinq mois, avec un rendu de rapport prévu pour le 24 mai 2021 et une soutenance du projet prévu pour le 28 mai. La première phase du projet a consisté à prendre connaissance de l'objectif et des contraintes de celui-ci. Après cela, nous avons discuté et réfléchi à la manière dont nous allions réaliser ce projet et des problématiques que nous rencontrerions sûrement. En découle un temps d'appropriation et de compréhension du matériel nécessaire afin de le réaliser. Ce temps de réflexion commune nous a occupé durant les trois premières semaines de janvier.

Par la suite, nous aurons à configurer notre switch afin de l'intégrer à notre réseau local et pouvoir configurer ses ports, grâce à cela nous pourrons récupérer le trafic de notre réseau via le switch. Nous pensons réaliser l'ensemble de ce travail dans un délai de un mois. Après la capture de notre trafic, il nous faudra pouvoir le lire et ainsi l'analyser afin de pouvoir faire ressortir nos appareils connectés présents sur le réseau domestique écouté. Cette analyse devra nous permettre d'extraire et de stocker toutes les informations utiles sur nos objets connectés. Cette identification sera réalisée sur une période de quatre semaines. Pour exploiter ces données, nous testerons les appareils via un outil d'intrusion afin de déceler les premières failles que nous pouvons lister. Puis, nous chercherons dans une base de données stockant les failles connues pour de nombreux appareils, si, nos appareils s'y trouvent et possèdent des failles supplémentaires. Nous stockerons les failles que nous avons pu relever et cela se clôturera en une période de vingt jours.

Nous aurons à notre disposition les informations souhaitées et nous devrons donc travailler l'interface d'une application afin de lancer ce projet et afficher le résultat de cette exécution. Le développement de cette application se fera sur dix jours Il nous restera alors pour finir ce projet une à deux semaines afin de peaufiner notre programme et rédiger/-structurer notre rapport. Pour finir, nous aurons à réaliser notre soutenance prévue pour fin mai.

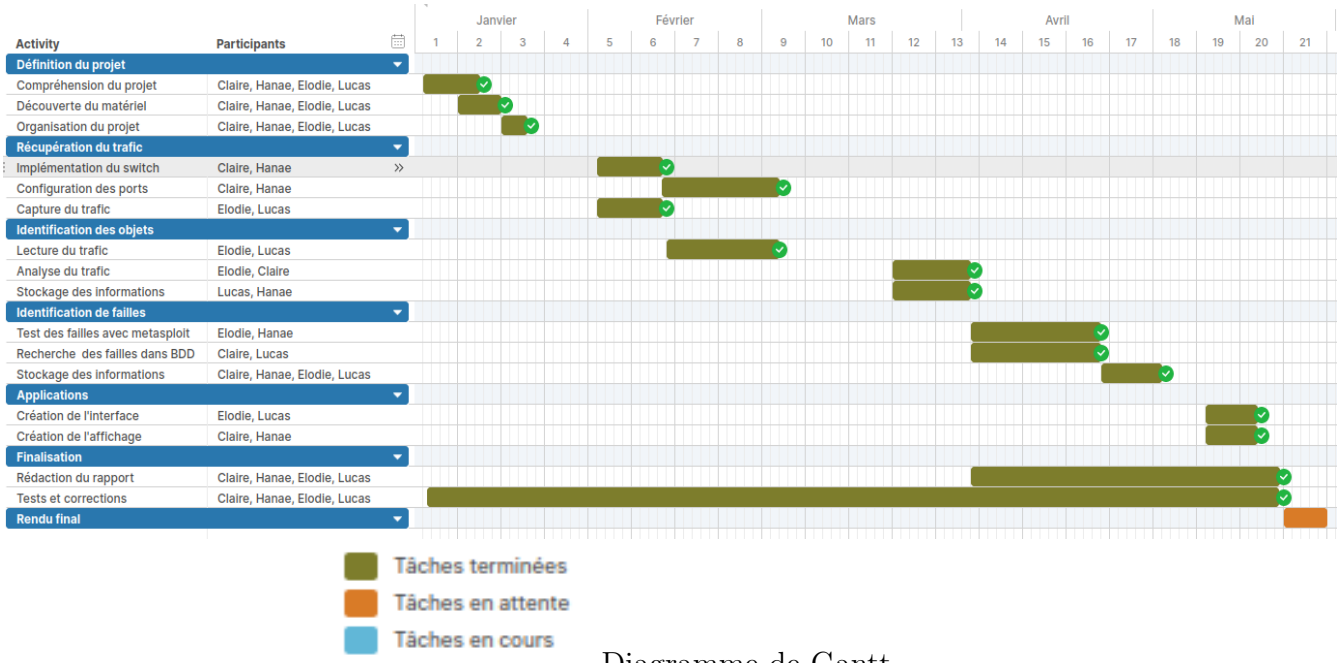


Diagramme de Gantt

3 Analyse

3.1 Internet des objets

L'IoT fait référence au géant réseau d'objets physiques capables de transférer des informations et des données sans avoir besoin d'une intervention humaine. Il existe de nombreux modèles d'architecture IoT qui partagent tous un ensemble de composants de trois couches. Tout d'abord, il y a une couche physique commune qui est une combinaison de capteurs, d'actionneurs et d'objets physiques. Ensuite, nous avons la couche réseau, couche où les composants opérationnels de la couche physique se connectent, communiquent et interagissent les uns avec les autres. Cette couche se compose généralement de passerelles. Pour finir, nous avons la couche application où la plateforme IoT utilise diverses applications et composants pour fournir des services IoT entièrement interopérables et permettre la gestion de ces services.

La plupart des applications IoT est basée sur le modèle client/serveur, ce qui signifie qu'un serveur central peut fournir des ressources à plusieurs clients simultanément. L'utilisation de cette architecture peut conduire à compromettre la confidentialité et l'intégrité des données par l'analyse du trafic, l'écoute clandestine et la surveillance massive du réseau.

3.2 Fonctionnement de l'Internet des objets

L'Internet des objets fonctionne principalement avec des capteurs et des objets connectés placés dans des infrastructures physiques. Ces objets sont capables d'émettre les données grâce à des capteurs et de les remonter à l'aide d'un réseau sans fil sur des plateformes IoT.

Les objets connectés pourront d'une part remonter des informations telles que leur identité, leur état, une alerte ou les données de capteurs et d'autre part recevoir des informations telles que des commandes d'action et des données.
Pour ce faire, différents protocoles de communication sont mis en œuvre pour assurer la transmission des données sur le réseau. Ces protocoles sont choisis en fonction de nombreux critères tels que le débit de données, la distance et le type de communication. Tout d'abord, les protocoles comme LoRa, 4G et 5G sont capables de faire transiter les données sur de vastes distances, et sont souvent utilisés par les entreprises qui connectent des kilomètres d'infrastructure à internet.

Ensuite, d'autres protocoles sont utilisés pour transférer les données sur de faibles distances et sont souvent utilisés dans la domotique. Le Wi-Fi est le protocole le plus couramment utilisé dans les réseaux IoT à courte portée. Il s'agit d'un protocole de communication sans-fil basé sur IP. Ethernet est aussi un protocole de communication de la deuxième couche du modèle OSI pour l'envoi de données entre les appareils dans un réseau local. Celui-ci se base sur la commutation de paquets et sur une connexion physique via un câble Ethernet.

De plus, Bluetooth est un protocole important pour les applications IoT à courte portée, à faible bande passante et à faible latence. Les avantages du Bluetooth incluent une consommation d'énergie réduite, un temps de configuration réduit et la prise en charge de la topologie de réseau en étoile avec un nombre illimité de nœuds. Pour finir, la RFID est un autre type de protocole qui permet d'interconnecter les objets situés à des distances proches. Il existe deux technologies RFID. La RFID active : il s'agit d'objets comprenant un tag et capables d'émettre le signal à proximité d'un autre objet RFID. L'intérêt de de cette technologie est d'étendre la portée du signal à quelques mètres. La RFID passive est utilisée quotidiennement pour le contrôle d'accès ou les pointeuses, mais aussi pour les paiements sans contact. Dans la RFID passive, le tag est une simple antenne qui est activée dès lors qu'elle se trouve à portée d'un émetteur.

3.3 Identification des objets connectés

L'objectif de notre travail est l'écoute du canal, dans le but d'identifier les appareils connectés sur Internet. Ceci devra se faire sans l'utilisation préalable de leur réseau attribué ou de leurs informations d'identification telles que l'adresse IP, l'adresse MAC (Medium Access Control), le numéro de série électronique (ESN), le numéro d'identification de l'équipement de la station mobile internationale (IMEI) ou le numéro d'identification

mobile (MIN).

Tout d'abord, nous avons l'empreinte digitale de l'appareil (DFP - Device fingerprint) qui est la méthode la plus connue pour identifier et suivre un appareil connecté dans un réseau local à l'aide du trafic réseau provenant de l'appareil. Le principe de l'empreinte digitale est de combiner certains attributs de l'appareil tels que le système d'exploitation, le type, la version utilisée, le nom de l'appareil, l'adresse MAC et l'adresse IP, de façon à ce que cette identification soit unique. Cette méthode repose sur la probabilité qu'un appareil reconnu comme ayant certains attributs sur un jour est le même dispositif vu avec ces mêmes attributs un autre jour.

De plus, divers services sont communiqués pendant la capture de trafic réseau (trafic entrant et sortant) sur le point d'accès (AP), par exemple le DNS pour traduire le nom de domaine des ressources Internet, les certificats SSL / TLS pour le chiffrement et déchiffrement des données, le serveur NTP pour synchroniser l'heure de l'appareil et le serveur de fabrication (manufacturer) pour mettre à jour les services du firmware. Nous pouvons aussi nous baser sur l'analyse de protocoles de la couche applicative comme par exemple les protocoles de découvertes de services (SSDP - Simple Service Discovery Protocol)[10][11]. En effet, un objet uPnP envoie sa description au réseau. Cette description sera sous la forme d'un fichier xml permettant de donner des informations sur le nom du modèle, ses capacités, le numéro de série, le nom du fournisseur ou encore les sites webs des fournisseurs. Le protocole DHCP permet quant-à lui la configuration automatique d'un objet dans un réseau par l'attribution automatique d'une adresse IP et d'un masque sous-réseau. Lors de cette connexion, les champs optionnels renseignent différentes informations sur l'objet connecté comme le nom de l'hôte ou le modèle et le système d'exploitation.

De plus, une autre information exploitable est que chaque adresse MAC différente indique la détection d'un nouvel appareil. Cette adresse MAC, obtenue par analyse des trames, renseigne aussi différentes informations. Les trois premiers octets indiquent le nom du fabricant tandis que les trois derniers indiquent le modèle du produit.[1]

Enfin, une intuition selon laquelle les fonctionnalités distinctes des protocoles de l'en-tête de paquet fournissent une empreinte digitale significative pour l'identification de l'appareil. Alors, l'empreinte digitale est générée à l'aide des informations d'en-tête de tous les paquets capturés. Par exemple, une méthode qui peut être utilisée est l'identification d'un objet par le champ User Agent de l'entête des requêtes HTTP. Pour cela, il faut que l'objet envoie des requêtes HTTP sur le réseau. Le système d'exploitation, le modèle et le type d'objet sont des informations qui peuvent être extraites à partir de l'agent utilisateur de la requête. D'autres données pourront être exploitées pour préciser nos analyses comme la taille du flux et des paquets ou encore les protocoles utilisés dans les différentes couches.

3.4 Raspberry

Le Raspberry Pi est un ordinateur de la taille d'une carte de crédit qui ne possède que quelques éléments [4, 28]. En effet, cet appareil est fourni nu, il n'y a que la carte mère, et il exige au minimum que deux éléments pour fonctionner. Tout d'abord, il a besoin d'un support de stockage mémoire contenant un système d'exploitation, dans notre cas il s'agit d'une micro SD contenant le système d'exploitation Raspbian. De plus, il exige une source d'alimentation électrique qui dans notre cas est fournie par un câble micro-USB. C'est pourquoi, il est facile d'obtenir cet ordinateur à faible coût. Pour plus de confort, nous avons dans notre cas un écran, un clavier avec souris intégrée et la possibilité, grâce à une entrée HDMI, de brancher le Raspberry Pi à un plus grand écran.

De plus, nous avons décidé d'utiliser comme système d'exploitation Raspberry Pi OS [5], anciennement Raspbian. Raspberry Pi OS est un système d'exploitation libre et gratuit basé sur Debian. Il permet au Raspberry Pi de fonctionner sans écran, ni clavier, ni souris, car le protocole SSH peut être activé avant le premier démarrage, c'est pourquoi un contrôle à distance est possible. Dans notre cas, cette fonctionnalité ne sera probablement pas utile. De plus, Raspbian nous permet d'utiliser cet appareil comme n'importe quel ordinateur de bureau et il fournit l'environnement de programmation python qui nous intéresse.

Ainsi, pour ce projet nous allons utiliser Raspberry Pi comme sonde pour capturer le trafic d'un réseau domestique à l'aide de Wireshark. Il nous permettra de programmer et d'analyser ce trafic à l'aide de python et sa bibliothèque pyshark. Et pour finir, il nous permettra de travailler sur une interface pour l'affichage des informations pour le client.

3.5 Commutateur réseau

Un commutateur réseau est un boîtier doté de quatre à plusieurs ports Ethernet, permettant de connecter les appareils au sein d'un réseau, souvent un réseau local, et de transférer les paquets de données vers et depuis ces appareils. Contrairement à un routeur, un commutateur envoie les données uniquement à l'appareil auquel elles sont destinées, il peut s'agir d'un autre commutateur, d'un routeur ou d'un ordinateur. Cela signifie que les routeurs sont nécessaires pour une connexion Internet, tandis que les commutateurs ne sont utilisés que pour interconnecter des appareils.

Ainsi, les commutateurs réseaux peuvent fonctionner au niveau de la couche 2, qui est la couche liaison de données, ou au niveau de la couche 3, qui est la couche réseau. La couche 2 transfère les données en fonction de l'adresse MAC de destination, tandis que la couche 3 transfère les données en fonction de l'adresse IP de destination. Certains commutateurs peuvent faire les deux.

Les commutateurs de couche 2 se connectent le plus souvent aux périphériques à l'aide de câbles Ethernet. Ainsi, ils se réfèrent aux adresses MAC afin d'envoyer le trafic Internet

aux bons appareils et non pas aux adresses IP. Les adresses IP changent souvent en raison du nombre limité d'adresses IPv4, les appareils des utilisateurs se voient généralement attribuer de nouvelles adresses lorsqu'ils établissent une nouvelle connexion avec le réseau. Contrairement aux adresses IP, les adresses MAC ne changent pas. Une adresse MAC est un identifiant permanent pour chaque élément matériel, équivalent à un numéro de série.

Pour finir, nous distinguons deux types de commutateurs : le commutateur non configurable qui crée simplement plus de ports Ethernet sur un réseau local, afin que davantage d'appareils locaux puissent accéder à Internet, et le commutateur configurable qui offre aux administrateurs réseaux un contrôle beaucoup plus grand sur la façon dont le trafic est priorisé. Les commutateurs configurables présentent plusieurs services pour la gestion des réseaux locaux. Nous pouvons citer parmi les services, la supervision du réseau en utilisant le protocole Simple Network Management Protocol (SNMP), la configuration des réseaux locaux virtuels, le contrôle de flux, le filtrage par adresse MAC et le port mirroring qui permet la réplication du trafic d'un port vers un autre port de destination choisi.

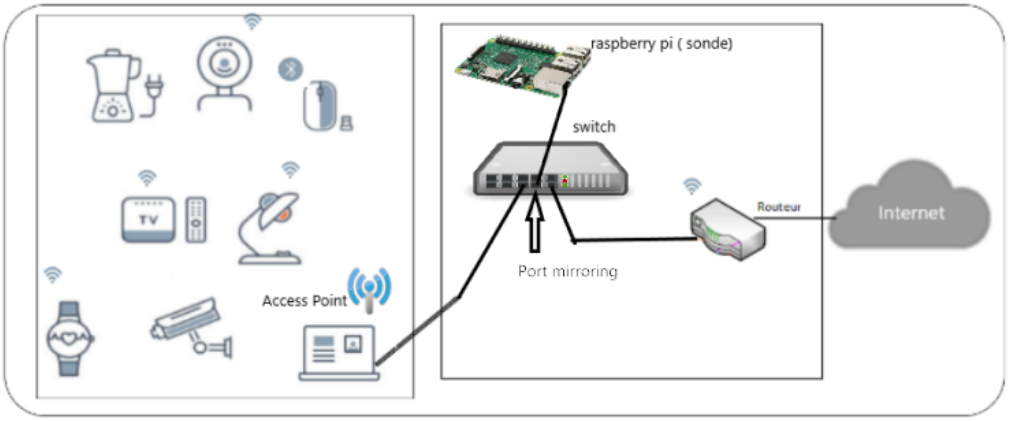
3.6 Metasploit

Metasploit Framework est un sous-projet de Metasploit [3] provenant de l'entreprise Rapid7 qui est en lien avec la sécurité des réseaux informatiques. Le but de ce framework est d'effectuer des tests de pénétration sur des objets à distance et ainsi de fournir des informations sur les vulnérabilités de ces objets.
Plus précisément, cet outil permet d'exécuter, contre une machine distante, des exploits. Ces derniers, sont plus particulièrement des séquences de commandes permettant la pénétration dans un système par un attaquant en exploitant une de ses failles. Nous pouvons citer comme exemple d'attaque, les tests d'intrusion par authentification login/mot de passe.
Nous allons désormais discuter de la série de commandes [32] qui nous permet d'utiliser les fonctionnalités fournies pour nos besoins.

- L'interface msfconsole permet d'accéder à la plupart des fonctionnalités de Metasploit. Son lancement se fait avec la commande "*msfconsole*".
- La commande "*nmap -sV adresseIp*" permet de tester tous les ports ouverts d'un système (ici celui de adresseIp) pour déterminer le service en écoute et sa version.
- Après avoir déterminé les versions, la commande "*search version*" permet de trouver tous les exploits liés à cette version.
- Enfin, la commande "*info exploit*" nous donne des informations sur l'exploit comme une description du module par exemple.

Ainsi, l'utilisation de ces commandes nous permet d'obtenir une liste des vulnérabilités d'un objet connecté simplement grâce à son adresse IP. C'est pourquoi, à terme, cet outil peut nous permettre de recenser toutes les vulnérabilités de chaque objet connecté à un réseau domestique.

4 Conception



Architecture globale de notre réseau domestique

4.1 Redirection du trafic généré par les objets connectés dans un réseau domestique

Notre partie conception consiste donc à exploiter les informations, décrites dans la partie [3] de notre projet, pour élaborer des stratégies et mises en oeuvre qui nous permettront de répondre à notre problématique. Les objets connectés sont des systèmes capables d'interagir entre eux et avec leur environnement grâce à leur interconnexion avec l'Internet. Pour cela, des équipements réseaux permettent de connecter ces objets par WIFI (connexion sans fils) ou Ethernet (connexion filaire).

Notre travail se concentre sur l'identification des objets connectés d'un réseau domestique et l'analyse de leurs vulnérabilités. Une étape essentielle est donc la redirection et la capture du trafic généré par ces objets. Dans notre cas, la box constitue une passerelle de multiplexage de données entrantes ou sortantes exploitables par les différents objets connectés de notre réseau puisqu'elle offre les fonctionnalités d'un routeur et d'un point d'accès (borne Wi-Fi).

La mise en place de notre réseau domestique expérimental consiste à utiliser cette box afin de récupérer le trafic des objets connectés. Pour cela, nous avons à notre disposition un switch Ethernet à 12 ports. Notre switch est un équipement qui nous a été prêté par l'université. Il est habituellement accessible en Telnet via le réseau d'administration (adresse IPv4 10.0.0.0/16) à partir d'un ordinateur de la PPTI. Ainsi cet ordinateur N peut accéder au commutateur N à l'adresse IPv4 10.0.2.N. Cette adresse étant statique, il n'est pas possible de configurer le commutateur si l'ordinateur ne se trouve pas dans le même réseau. Il faut donc configurer manuellement l'adresse IPv4 de l'ordinateur. Pour cela, il faut se rendre dans les paramètres de connexions réseau de l'ordinateur puis modifier

les options d'adaptateur. Dans les propriétés ethernet de notre machine, nous pouvons modifier la section "Protocole Internet Version 4 (TCP/IPv4)". Manuellement, nous allons utiliser l'adresse IP : 10.0.0.2 (adresse quelconque se situant dans le réseau 10.0.0.0/16) et le masque de sous-réseau 255.255.0.0.

Ainsi, nous pouvons maintenant brancher notre ordinateur au commutateur à l'aide d'un câble ethernet. Par le terminal il suffit de se connecter au commutateur avec la commande : telnet 10.0.2.N (N étant le numéro du commutateur) puis rentrer le mot de passe : mmqos.
Il s'agit désormais de changer l'adresse IP du commutateur pour le mettre dans notre propre réseau avec les commandes ci-dessous [2] :

- enable (on nous demande ici un mot de passe : mmqos.)
- show running-config (permet de situer le Vlan dans lequel se situe le commutateur)
- configure terminal
- interface VlanX (X étant le numéro du Vlan)
- ip address xxx.xxx.xxx.xxx xxx.xxx.xxx.xxx (adresse IP puis masque)

A partir de là, nous n'avons plus accès au switch. Il faut désormais changer l'adresse IPv4 de notre ordinateur pour se trouver dans le même réseau que le switch (192.168.0.0/24) sans éteindre le switch. Puis, nous nous connectons de nouveau au switch via telnet avec l'adresse nouvellement configurée.

- enable (mmqos)
- show running-config (permet de vérifier que les modifications ont été actualisées)
- wr mem (permet d'écrire les changements dans la mémoire du switch)

L'accès au commutateur par le protocole telnet nous permet de configurer un port source et destination afin de pouvoir faire du port mirroring. Le port mirroring est un dispositif qui permet de copier les trames transitant d'un port source vers un port destination.
Ainsi, la dernière étape de la configuration du switch est la configuration d'un port mirroring.

Tout d'abord, nous avons configuré le port de sortie auquel nous connecterons le raspberry afin qu'il se trouve dans le même Vlan. Ainsi, si nous choisissons le port numéro 6 nous utilisons les commandes suivantes :

- show run (pour vérifier que cet interface appartient au Vlan de notre réseau)

Si le Vlan est le même alors il n'y a pas de manipulation supplémentaire, mais dans le cas contraire il faut utiliser les commandes suivantes :

- SwN#configure terminal
- SwN(config)#interface Fa0/6
- SwN(config-if)#switchport mode access
- SwN(config-if)#switchport access vlan X
- SwN(config-if)#exit
- SwN(config)#exit
- SwN#sh running-config interface Fa0/6 (afin de vérifier que la configuration est bonne)
- SwN#wr mem

Puis, nous sommes passés à la configuration de port mirroring. Nous avons tout d'abord vérifié l'existence d'un port mirroring d'une configuration précédente à l'aide de "show run". Dans notre cas, il existait déjà une configuration, nous l'avons donc supprimé afin de configurer la notre :

- SwN#configure terminal
- SwN(config)#no monitor session 1 (permet de supprimer la configuration précédente)
- SwN(config)#monitor session 1 source interface Fa0/1 both
- SwN(config)#monitor session 1 destination interface Fa0/6
- SwN(config)#exit
- SwN#show run (afin de vérifier que la configuration est bonne)
- SwN#wr mem

Pour finir, un des problèmes que nous avons rencontré est que le trafic émis par les objets connectés via Wi-Fi ne passe pas dans le switch. Pour pouvoir récupérer et analyser ces données, il nous a fallu trouver un dispositif récupérant le trafic Wi-Fi et traversant le commutateur afin d'être analysé au même titre que le trafic filaire.
Dans notre cas, nous avons décidé d'utiliser un ordinateur connecté en Ethernet sur un port du switch appartenant au même Vlan que les ports source et destination. En effet, cet ordinateur nous permet d'avoir un accès Wi-Fi sur lequel connecter nos objets. Ainsi, les trames Wi-Fi deviennent des trames Ethernet en passant par l'ordinateur de telle sorte à ce que la sonde puisse écouter le trafic.

En conclusion, nous avons décidé de connecter notre routeur sur le port 1 du switch, port source, notre Raspberry Pi 3 sur le port 6 pour sonder le trafic, port destination, et un ordinateur permettant de faire office d'*access point* sur un autre port du même Vlan.

4.2 Capture et identification des objets connectés

Le trafic généré par les différents objets connectés à notre réseau domestique sera capturé à l'aide de la bibliothèque "subprocess", qui lancera la commande tshark sur l'interface eth0 de notre Raspberry Pi, pendant une durée définie par le client sur l'interface.

Afin d'être certains de détecter absolument tous les objets connectés (certains n'envoient pas systématiquement du trafic), il est préférable que le client laisse la sonde écouter le trafic durant une durée d'environ une journée.

Comme nous l'avons développé dans la partie 3.3 de notre rapport, il existe plusieurs moyens d'analyser et de détecter nos appareils connectés, reposant sur l'analyse de trames. Le code que nous avons implémenté utilise le langage python afin d'utiliser le package *pyshark* permettant l'analyse de paquet. A la différence de beaucoup d'autres analyseurs de paquets, *pyshark* utilise *tshark* (utilitaire de ligne de commande Wireshark) afin d'exporter les paquets sous forme de fichier XML et pouvoir l'analyser. Ce module permet l'analyse de paquet à partir d'un fichier de capture de type pcapng généré par Wireshark ou bien *tshark*. Il aide notamment à la récupération d'informations comme l'adresse IP/ MAC, les ports et les protocoles utilisés.

Une fois la capture terminée par la sonde, nous la filtrons pour garder uniquement les protocoles DNS, SSDP, DHCP, HTTP [10, 20]. Puis, à l'aide de cette capture, notre code nous permet tout d'abord de recenser sous forme de liste toutes les adresses MAC de tous les paquets de la capture, avec leurs différentes adresses IPs. Par conséquent, notre point d'accès, dont l'adresse MAC aura été précisée sur l'interface, sera lié à de nombreuses adresses IPs. Cette liste est en fait une liste de sous-listes d'objets dont le premier élément est l'adresse MAC. Ainsi chaque premier élément de chaque sous-liste, l'adresse MAC, est comparé à la base de données "macaddress.io-db.json" [1] contenant les informations des adresses MAC des vendeurs. Ainsi, grâce à cette base de données, nous pouvons ajouter à chaque objet son constructeur, si celui-ci est renseigné, dans la sous-liste de l'objet.

Par la suite, nous filtrons la capture en fonction des protocoles dans lesquels nous allons récupérer des informations plus précises sur les objets.
Pour commencer nous filtrons avec le protocole DHCP, ce protocole nous permet de récupérer les "*hostname*" de chaque objet. Ainsi, si l'adresse MAC de l'objet ne correspond pas à l'adresse MAC du point d'accès, nous ajoutons dans sa sous-liste son hostname. [voir annexe 7.2.1]

Puis, nous filtrons désormais avec le protocole SSDP, ce protocole renvoie dans un champ précis, *location*, un lien vers un fichier xml de l'objet. Ainsi pour certains objets nous pouvons récupérer, dans ce fichier xml, l'adresse MAC de l'objet, le vendeur, le modèle, la description de l'objet et type d'objet. Par conséquent, si ce champ xml est dans un paquet dont l'adresse MAC correspond à celle du point d'accès (AP), alors nous ajoutons ces informations dans une liste différente. Cette liste concentrera les informations des objets connectés en Wi-Fi via l'AP. Sinon, nous pouvons ajouter les informations recueillies de ce fichier xml dans la sous-liste de l'objet, l'objet étant connecté via Ethernet.

Pour finir, nous filtrons la capture avec le protocole HTTP, celui-ci nous permet d'obtenir le champ User-Agent des objets. Ceci aura pour effet d'obtenir les différentes versions des OS des objets. De la même manière que pour SSDP, si le champ User-Agent se trouve

dans un paquet dont l'adresse MAC correspond à l'AP, nous mettons ces informations dans une liste qui concentrera toutes les informations des objets connectés en Wi-Fi via l'AP. Sinon, nous pouvons ajouter ce champ dans la sous-liste de l'objet, l'objet étant connecté via Ethernet.

Pour finir, grâce à toutes ces données collectées nous pouvons désormais identifier les différents objets du réseau domestique et passer à la partie sur la recherche de leurs vulnérabilités.

4.3 Identification des vulnérabilités

Une fois avoir réalisé l'identification des objets connectés à notre réseau domestique, il s'agit de déterminer leurs failles. En effet, les objets connectés sont la cible d'attaques visant l'architecture de leurs réseaux, en particulier par leur protocole de routage. Ainsi, leurs fonctions sécuritaires (chiffrement et authentification par exemple) ne suffisent pas à sécuriser le système.

Par l'analyse des trames émises par nos objets, nous avons pu identifier des spécificités propres à chacun d'eux et qui constituent des failles qui rendent possibles certaines attaques. Une version d'un système d'exploitation d'un ordinateur peut, par exemple, présenter des défauts sécuritaires que les développeurs tenteront de combler par l'amélioration de cette version.

Une première phase d'approche dans l'identification de ces vulnérabilités est l'exploitation des champs d'informations contenues dans nos trames. Les données HTTP, en particulier, nous fournissent des informations concernant le système d'exploitation utilisé par nos objets, et, dans la majorité des cas, nous informent aussi de leur version. Par l'analyse du champ User-Agent de nos trames HTTP, nous avons pu détecter des modèles récurrents selon les systèmes d'exploitation et leur version.
Par exemple, les objets possédant le système d'exploitation Ubuntu version 16.10 ont dans leurs trames HTTP un champ User-Agent de la forme *Mozilla/5.0 (X11 ; Ubuntu ; Linux x86_64 ; rv :50.0) Gecko/20100101 Firefox/50.0*. En réunissant ces modèles récurrents, nous avons pu constituer notre propre base de données associant des systèmes d'exploitation connus et leur champ User-Agent, afin d'effectuer une première phase d'identification.

Cependant, notre base de données étant incomplète, il a fallu traiter les champs manuellement. Dans une majorité des cas, il suffisait d'extraire l'information directement en clair. Suite à cette phase, il s'agit d'identifier, pour chaque modèle d'OS trouvé, les vulnérabilités qui lui sont associées. Pour cela, nous nous aidons d'un dictionnaire référençant les vulnérabilités de sécurité (Common Vulnerabilities and Exposures ou CVE). En nous munissant de la base de données disponible [25], nous pouvons lister toutes les failles et leur identifiant qui ont été trouvés pour cette version d'OS. [voir annexe 7.2.2]

Enfin, comme nous avons pu le voir dans la partie 3.6 du projet, différentes commandes permettent d'obtenir les vulnérabilités d'un objet connecté grâce à Metasploit. Néanmoins, pour éviter que notre client ait à utiliser Metasploit manuellement, nous avons décidé de l'automatiser. Ainsi, nous avons automatisé cette phase de recherche des vulnérabilités via Metasploit avec la bibliothèque *Pymetasploit3* qui permet d'interagir avec Metasploit grâce à un plugin msgrpc dans msfconsole. Il suffit de démarrer le serveur RPC sur le port 55552 et de connecter un client RPC grâce à la classe MsfRpcClient qui fournit les fonctionnalités du framework Metasploit. Cette classe implémente plusieurs modules de gestion et nous utiliserons plus particulièrement le module consoles qui permet l'interaction avec les consoles/shells. Il suffit ensuite d'exécuter via ce client, les commandes décrites plus haut et de stocker les informations des vulnérabilités détectées relatives à une adresse IP que l'on aura fourni en paramètre. Pour tester nos fonctions sur un système présentant plusieurs failles, nous avons téléchargé Metasploitable, une machine virtuelle présentant intentionnellement plusieurs vulnérabilités exploitables.

5 Compte rendu du projet

L'objectif de notre projet est de réaliser un audit des réseaux locaux, qui permet d'évaluer les vulnérabilités des objets connectés dans n'importe quel réseau local. Le but est de concevoir un outil facile d'utilisation à l'aide d'une interface graphique afin de permettre à chaque utilisateur d'analyser son réseau domestique en identifiant ses objets connectés et leurs vulnérabilités.

Pour ce faire, nous avons utilisé un ordinateur de type Raspberry Pi qui fait une analyse interne du réseau en suivant trois phases. L'écoute du réseau, l'identification des objets et l'évaluation de vulnérabilités des objets ainsi identifiés.

La mise en place de l'audit s'appuie sur une infrastructure dédiée à ce réseau. Tout d'abord, un routeur connecte les différents équipements dans le réseau local.
Ensuite, la configuration d'un port mirroring sur notre commutateur permet au trafic entrant par le port relié à notre box, d'être copié sur un port de sortie auquel nous rattacherons notre Raspberry Pi, qui aura un rôle de sonde. Ainsi, il pourra récupérer le trafic Ethernet du réseau.
Puis, à l'aide d'un troisième équipement filaire qui jouera le rôle de point d'accès, notre sonde aura la capacité de récupérer le trafic Wi-Fi. En effet, en passant par le point d'accès les trames Wi-Fi sont transformées en trames Ethernet, de sorte à ce que la sonde puisse écouter le trafic venant des équipements Wi-Fi.

Néanmoins, un problème auquel nous nous sommes confrontés est la configuration du point d'accès. Tout d'abord, notre première idée fut de configurer ce point d'accès directement sur notre Raspberry Pi pour respecter la contrainte de n'utiliser qu'une sonde pour l'identification des objets. Celui-ci n'aura qu'à écouter son interface Wi-Fi afin capturer les trames des équipements qui lui seront connectés. Or, nous avons tout de suite remarqué qu'il ne serait pas possible de mettre en place une telle manipulation. En effet, lors d'une configuration d'un port mirroring, le trafic entrant par le port de sortie de notre configuration est bloqué, ceci empêchant nos objets Wi-Fi de se connecter à notre réseau local.
C'est pourquoi, nous avons finalement décidé de rattacher un ordinateur à notre switch qui aura le rôle de point d'accès. Ainsi, les trames des objets se connectant à notre équipement peuvent être perçus par le Raspberry Pi.
Cependant, un autre problème s'est présenté. Les trames passant par le point d'accès sont désencapsulées à la sortie du point d'accès pour se voir attribuer l'adresse MAC de ce dernier. Il a donc fallu s'adapter à cette contrainte et trouver un autre moyen d'identifier cet appareil sans se baser sur son adresse MAC.

De plus, l'utilisation de Wireshark nous a permis d'entreprendre la deuxième phase de notre projet, qui consiste à identifier les objets connectés à notre réseau local. En ob-

servant les trames émises, nous avons pu repérer plusieurs types de trames contenant des champs essentiels au processus d'identification.

Tout d'abord, en se basant sur les champs de l'entête Ethernet, nous avons pu récupérer les adresses MAC de différents objets, ce qui nous a permis de connaître leur constructeur de carte réseau.
Par ailleurs, les trames DHCP nous ont servi pour associer aux objets leur Hostname, si le champ hostname avait été déclaré. Les Trames DHCP n'ont pas de fonctions essentielles à part celle d'aider à l'identification générale des objets, telle que connaître l'adresse mac, l'adresse IP et le Hostname. Mais ils ne permettent pas de connaître le système d'exploitation de l'objet ni sa version.
Ainsi, à l'aide du protocole SSDP et du fichier XML fourni par la trame, nous avons pu identifier les objets par leur adresse MAC, leur vendeur, leur modèle, leur description et leur type. Le protocole SSDP ne nous a pas permis d'identifier tous les objets connectés au réseau puisque certains objets n'émettent pas de trames SSDP.
C'est pourquoi, les trames HTTP nous ont été d'une plus grande utilité puisqu'elles nous ont servi à avoir d'autres informations sur l'objet, telles que le système d'exploitation et sa version, tout ceci grâce au champ User-Agent. En finalité, parmi toutes les trames que nous avons analysées, seulement une nous permet d'identifier précisément les informations qui nous aident à l'identification du système d'exploitation et de sa version des équipements. Puisque tous les objets n'émettent pas forcément de trames HTTP, notre analyse de leurs vulnérabilités a été limitée à certains objets.

Suite à la phase d'identification, il nous a fallu exploiter les informations extraites des trames afin d'obtenir les vulnérabilités référencées pour ces systèmes.
Les données essentielles à traiter sont le système d'exploitation utilisé par l'appareil ainsi que sa version, qui ont été extraites du champ User-Agent des trames HTTP. Or, le problème de ces champs est qu'il en existe beaucoup et que chacun d'entre eux diffère en fonction de l'OS et de sa version. Ainsi, notre première approche a consisté à concevoir une base de données référençant les systèmes d'exploitation les plus connus, ayant donc de grandes chances d'apparaître dans nos trames. Cependant, cette base de données est très limitée ce qui nous a conduit à une seconde approche qui consiste à extraire manuellement les informations que nous voulons directement dans le champ User-Agent. Cette technique est aussi limitée puisque certain champ ne précise pas ces informations. Mais, en combinant les deux solutions, nous espérons optimiser cette recherche. Enfin, grâce à une base de données répertoriant une liste de vulnérabilités en fonction d'une version d'OS, nous avons pu compléter notre phase d'évaluation des vulnérabilités des objets connectés à notre réseau.

L'utilisation de Metasploit a été décrite dans la partie 3.6 et 4.3 du rapport. Nous allons cependant discuter des problèmes que nous avons rencontrés au cours de son automatisation. Notre programme lance une interface graphique qui sert de fenêtre d'affichage des vulnérabilités pour le client. Puisqu'il faut charger manuellement un plugin afin de

pouvoir interagir avec la console msf et ainsi utiliser les fonctions de Metasploit, il nous a fallu gérer le fait d'utiliser le terminal pour l'entrée des commandes de Metasploit, et en parallèle l'interface graphique pour l'affichage des résultats. Pour cela, nous avons décidé d'utiliser la bibliothèque *pyautogui* afin d'interagir avec les fenêtres grâce à notre script, et pouvoir se balader d'une fenêtre à l'autre selon nos besoins. Cette contrainte empêche l'utilisateur de changer de fenêtre de lui-même, puisque le curseur de la souris doit se trouver à des fenêtres spécifiques pour assurer bon fonctionnement du code. Le manque de temps nous a empêché de nous pencher plus en profondeur sur la question. Enfin, le manque d'objets connectés nous a limité durant cette phase d'analyse des vulnérabilités. En effet, nos objets à disposition n'avaient pas ou très peu de vulnérabilités. Cela nous a poussé à chercher des solutions annexes afin de tester la validité de nos codes, comme par exemple en installant une machine virtuelle ayant volotairement des vulnérabilités comme Metasploitable et en effectuant nos analyses dessus.

L'interface graphique a été conçue pour faciliter l'usage et l'opérabilité de projet et permettre aux utilisateurs d'interagir facilement avec le système. Au lancement de notre programme, le client doit choisir parmi deux modes d'analyse des vulnérabilités. Il peut choisir d'analyser une capture pcap ou pcapng déjà réalisée. Ou bien de réaliser une capture en ligne en spécifiant la durée en seconde qu'il souhaite faire durer cette capture, la durée de la capture et l'adresse MAC du point d'accès si besoin. Une fois l'analyse faite, l'application retourne dans la fenêtre des log une liste de tous les objets identifiés par leur adresse MAC, leur système d'exploitation et version si précisés, ainsi que les noms des vulnérabilités liés à chaque OS et leur description détaillée.

Durant toute la durée de nos travaux, nous avons pu discuter avec notre encadrant de projet, Monsieur Sébastien Tixeuil, via des réunions ZOOM. Nous avons pu échanger à propos de nos problèmes, comme celui du point d'accès par exemple, pour lequel il nous a proposé différentes solutions. De plus, Monsieur Tixeuil a pu suivre l'avancement de notre projet et valider les idées que nous avons eues.

En conclusion, notre programme permet un audit d'un réseau local et permet à un client d'obtenir une liste des vulnérabilités des objets connectés à son réseau à travers une interface facile d'utilisation et totalement autonome.

6 Bibliographie

Références

[1] About MAC vendors database | Database download | MAC Address Vendor Lookup.

[2] Configurer le port-mirroring sous Cisco | Cisco | IT-Connect, June 2014.

[3] Metasploit, November 2020. Page Version ID : 176767154.

[4] Raspberry Pi, January 2021. Page Version ID : 178907433.

[5] Raspberry Pi OS, January 2021. Page Version ID : 1001464797.

[6] Tarek ABBES. *Classification du trafic et optimisation des règles de filtrage pour la détection d'intrusions*. thèse de doctorat, Université Henri Poincaré, Nancy, France, December 2004.

[7] Vivien Admin. Liste des User Agent par système d'exploitation et navigateur.

[8] Abbas Ahmad. *Model-Based Testing for IoT Systems - Methods and Tools*. thèse de doctorat, Université Bourgogne Franche-Comte, Besançon, France, June 2018.

[9] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi. Internet of Things (IoT) communication protocols : Review. In *2017 8th International Conference on Information Technology (ICIT)*, pages 685–690, Amman, Jordan, May 2017.

[10] Nesrine Ammar, Ludovic Noirie, and Sébastien Tixeuil. Identification du type des objets connectés par les informations des protocoles réseaux. In *Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*, Roscoff, France, May 2018.

[11] Nesrine Ammar, Ludovic Noirie, and Sébastien Tixeuil. Amélioration de l'identification du type des objets connectés par classification supervisée. In *CORES2019 - Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*, Jardins de Saint Benoît, France, June 2019.

[12] Pooja Anand, Yashwant Singh, Arvind Selwal, Pradeep Kumar Singh, Raluca Andreea Felseghi, and Maria Simona Raboaca. IoVT : Internet of Vulnerable Things ? Threat Architecture, Attack Surfaces, and Vulnerabilities in Internet of Things and Its Applications towards Smart Grids. *Energies*, 13(18) :4813, September 2020. Number : 18 Publisher : Multidisciplinary Digital Publishing Institute.

[13] Kishore Angrishi. Turning Internet of Things(IoT) into Internet of Vulnerabilities (IoV) : IoT Botnets. *arXiv :1702.03681 [cs]*, February 2017. arXiv : 1702.03681.

[14] Ralph Ankele, Stefan Marksteiner, Kai Nahrgang, and Heribert Vallant. Requirements and Recommendations for IoT/IIoT Models to automate Security Assurance through Threat Modelling, Security Analysis and Penetration Testing. In *Proceedings of the 14th International Conference on Availability, Reliability and Security - ARES '19*, pages 1–8, Canterbury, CA, United Kingdom, August 2019. ACM Press.

[15] Fahad Azam, Rashid Munir, Mehboob Ahmed, M. Ayub, Ahthasham Sajid, and Zaheer Abbasi. INTERNET OF THINGS (IOT), SECURITY ISSUES AND ITS SOLUTIONS. *Science Heritage Journal*, 3(2) :18–21, October 2019.

[16] Yann Bachy. *Sécurité des équipements grand public connectés à Internet : évaluation des liens de communication.* thèse de doctorat, Université Fédérale Toulouse Midi-Pyrénées, Toulouse, France, July 2015.

[17] EMBARKA BEN BRAHIM and SELYNA AMICHE. *Mise en place d'une solution de détection d'intrusion.* Mémoire de fin d'étude de Master Académique, UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU, Tizi Ouzou, Algérie, 2017.

[18] Muhammad Burhan, Rana Asif Rehman, Bilal Khan, and Byung-Seo Kim. IoT Elements, Layered Architectures and Security Issues : A Comprehensive Survey. *Sensors*, 18(9) :2796, August 2018. Number : 9 Publisher : Multidisciplinary Digital Publishing Institute.

[19] Aaron Yi Ding, Gianluca Limon De Jesus, and Marijn Janssen. Ethical hacking for boosting IoT vulnerability management : a first look into bug bounty programs and responsible disclosure. In *Proceedings of the Eighth International Conference on Telecommunications and Remote Sensing*, ICTRS '19, pages 49–55, New York, NY, USA, September 2019. Association for Computing Machinery.

[20] Brad Duncan. Wireshark Tutorial : Identifying Hosts and Users, March 2019.

[21] Lisa Goeke. *Security Challenges of the Internet of Things.* thèse de doctorat, Haaga-Helia, University of Applied Sciences, Helsinki, Finland, May 2017.

[22] Mohamed Tahar Hammi. *Sécurisation de l'Internet des objets.* thèse de doctorat, Université Paris-Saclay, Paris, France, September 2018.

[23] Karey Helms. Leaky Objects : Implicit Information, Unintentional Communication. In *Proceedings of the 2017 ACM Conference Companion Publication on Designing Interactive Systems*, DIS '17 Companion, pages 182–186, New York, NY, USA, June 2017. Association for Computing Machinery.

[24] Rwan Mahmoud, Tasneem Yousuf, Fadi Aloul, and Imran Zualkernan. Internet of Things (IoT) Security : Current Status, Challenges and Prospective Measures. In *The 10th International Conference for Internet Technology and Secured Transactions*, pages 336–341, London, United Kingdom, December 2015.

[25] NIST. NVD - Data Feeds, May 2021.

[26] Jean-François Pillou. Systèmes de détection d'intrusion (IDS), October 2008.

[27] Rapid7. rapid7/metasploit-framework.

[28] Raspberry Pi. What is a Raspberry Pi ?, 2015.

[29] Jonathan Roux. Détection d'Intrusion dans l'Internet des Objets : Problématiques de sécurité au sein des domiciles. In *Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information (RESSI)*, page 4, Grenoble, France, May 2017.

[30] Jonathan Roux. *Détection d'intrusion dans des environnements connectés sans-fil par l'analyse des activités radio*. thèse de doctorat, Université Fédérale Toulouse Midi-Pyrénées, Toulouse, France, February 2020.

[31] Mustafizur Shahid, Gregory Blanc, Zonghua Zhang, and Hervé Debar. IoT devices recognition through network traffic analysis. In *BIG DATA 2018 : IEEE International Conference on Big Data*, pages pp.5187–5192, Seattle, United States, December 2018.

[32] Guillaume Simard. Exploiter une faille de sécurité avec Metasploit, October 2017. Section : Piratage éthique.

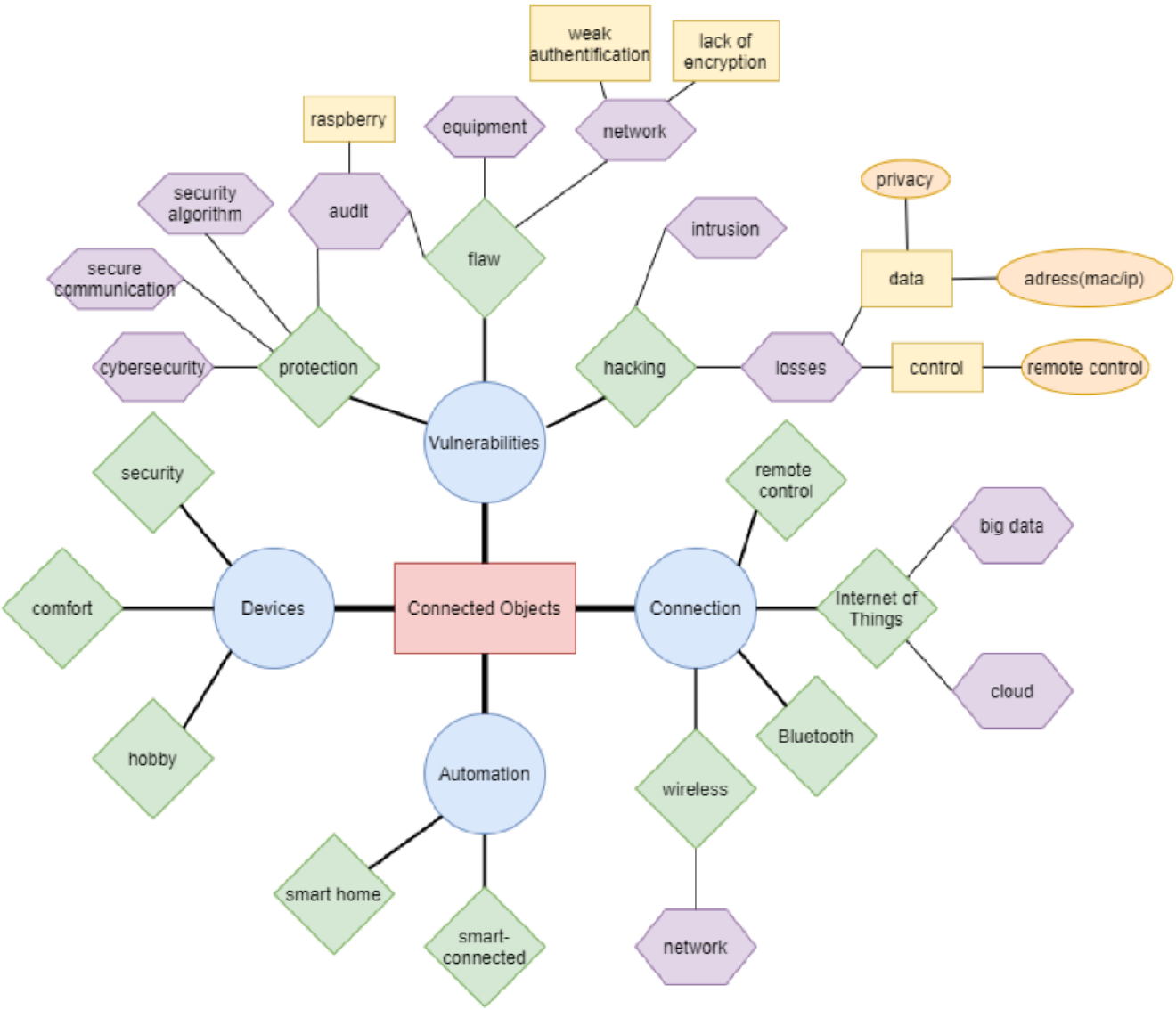
[33] Alanoud Subahi and George Theodorakopoulos. Detecting IoT User Behavior and Sensitive Information in Encrypted IoT-App Traffic. *Sensors (Basel, Switzerland)*, 19(21), November 2019.

[34] Jonathan Tournier, François Lesueur, Frédéric Le Mouël, Laurent Guyon, and Hicham Ben-Hassine. Audit d'un système IoT par test d'intrusion. In *RESSI 2018 - Rendes-Vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information*, pages 1–3., Nancy, France, May 2018.

7 Annexes

7.1 Annexe 1 - Carnet de Bord

Mots clés retenus



Descriptif de la recherche documentaire

Avant de débuter notre recherche, nous avons relevé les mots clés et thèmes essentiels au développement de nos recherches. Il a fallu nous concentrer sur les thèmes de "Sécurité/Vulnérabilité de l'IoT", "IoT", "Audit du réseau", "Classification des objets connectés".

A l'aide du site Sorbonne Université, nous avons pu accéder à des bases de données telles que ACM Digital Library Guide to Computing Literature et Europresse. Néanmoins, lors de l'utilisation du site de Sorbonne Université, nous avons été immédiatement confrontés à une difficulté primordiale : le nombre de ressources est important et les filtres de précision des recherches sont au contraire assez limités. En effet, nous limiter à la section "Informatique" du site nous a fait passer à côté de sites de ressources non spécialisées en informatique, mais qui contenaient pourtant beaucoup d'informations traitant notre sujet, comme Europresse. De plus, la plupart de ces sites nous font accéder à des extraits d'ouvrages ou d'articles dont l'intégralité du contenu ne peut être consultée qu'après l'achat de l'œuvre en question, comme c'est le cas pour la base de données ACM Digital Library qui offre des accès limités à certains ouvrages.

Nous avons donc débuté nos recherches grâce à ACM Digital Library Guide to Computing Literature. Cette base de données permet d'accéder à des journaux, revues, comptes-rendus de conférences publiés par l'Association for Computing Machinery. Ainsi, nous avons pu trouver plusieurs comptes-rendus de conférences en rapport à notre sujet avec un niveau de spécialisation élevé. Ensuite, nous avons pu accéder à la base de données Europresse. Celle-ci permet d'obtenir de nombreux articles de presse en français et en anglais notamment, qu'ils soient spécialisés ou non. Sur la page de description des articles, la section "Related articles" nous a permis de faire rebondir nos recherches sur d'autres articles traitant aussi de notre sujet. Sur ce site, nous nous sommes principalement concentrés sur la recherche du thème de la sécurité dans le domaine des objets connectés.

Afin d'élargir les supports d'information, nous avons décidé de faire nos recherches dans une base de données regroupant des thèses de doctorat soutenues en France sur le site theses.fr. Nous avons ici utilisé des mots clefs en français comme "intrusion", "sécurité", "sans-fil" complété soit par "IoT", soit par "Internet des objets". Après cela, il a fallu nous concentrer sur les thèses ayant reçu la mention "Validée par le jury" pour être sûr de la fiabilité de ces thèses.
De plus, nous avons aussi trouvé des ouvrages grâce à l'encyclopédie wikipédia sur la page Internet of Things (en). Malheureusement, la plupart des ouvrages ne sont pas consultables intégralement en ligne. Pour vérifier la disponibilité des ouvrages sélectionnés durant nos recherches, nous avons tout de même utilisé l'outil SUper qui permet d'accéder au catalogue des bibliothèques de l'université et nous avons demandé l'aide de nos professeurs qui eux, auraient pu accéder au contenu complet de l'ouvrage.

En vue d'obtenir des résultats différents et de pouvoir élargir les informations sur

notre sujet nous avons utilisé BASE. Ce moteur de recherche de l'université de Bielefeld est spécialisé dans le domaine universitaire. De plus, il effectue une recherche plein texte ce qui nous permet d'obtenir de nombreux nouveaux résultats, tout en gardant la possibilité d'appliquer de nombreux filtres sur notre recherche afin de la limiter. BASE possède un grand échantillon de documents universitaires de nationalités différentes. Nous avons donc effectué nos recherches sur BASE à l'aide de mots clés en anglais tels que "vulnerabilities" ou "vulnerability", "hacking", "security" complété du mot clef principal "IoT", afin d'obtenir le plus de résultats possibles. Dans le même esprit, l'utilisation de TEL nous a été utile, car il permet de faire une recherche, avec une grande possibilité de filtres, dans l'archive ouverte pluridisciplinaire HAL. Celle-ci regroupe une grande quantité de documents scientifiques de niveau recherche de tout horizon. Cela nous permet donc d'avoir des documents assez pointus et de sources fiables. En effet, le site nous donne directement des informations sur les auteurs de ces documents ainsi que les sociétés et/ou universités auxquelles ils appartiennent. Ainsi, nous avons pu avoir accès à des sources à l'aide des mots clés en anglais "audit", "intrusion", "devices", en plus du mot clef "IoT".

Par la suite, l'accès à toutes ces informations nous a permis de rebondir facilement sur de nouveaux documents en nous servant des sources dans les documents que nous avons consultés. Ou bien en consultant les autres travaux des auteurs des documents sur ce sujet, nous sommes tombés sur des nouvelles sources intéressantes.

Evaluation des sources

Audit d'un système IoT par test d'intrusion [34]

Pour trouver ce document, nous avons utilisé le moteur de recherche proposé par HAL en utilisant les mots clés "audit" et "objets connectés". Nous sommes alors tombés sur cette communication à un congrès datant de mai 2018 ce qui nous satisfait, car cela reste dans l'air du temps. Nous avons pu par la suite nous concentrer sur son contenu qui traite un "audit d'un système IoT par test d'intrusion", thème intéressant pour nous puisqu'il rejoint notre sujet. En effet, nous allons sûrement devoir effectuer un audit des objets connectés d'un réseau local ainsi que les classifier comme abordé dans ce document. De plus, les informations sont adaptées à un congrès qui est un public composé de chercheurs. Cela nous permet donc d'obtenir des informations précises et fiables.

De plus, cette communication nous provient de cinq chercheurs et professionnels de la sécurité ayant tous obtenu un diplôme d'ingénieur, en collaboration avec le laboratoire CITI-lab, de l'INSA-Lyon et AlgoSecure entreprise spécialisé dans la sécurité informatique. C'est pourquoi nous pouvons nous fier aux propos tenus dans leur travail. Ainsi, cette communication pour un congrès nous semble donc une source fiable et intéressante.

IoT Elements, Layered Architectures and Security Issues : A Comprehensive Survey [18]

La revue universitaire choisie a été trouvée par le biais d'une recherche dans le catalogue de la bibliothèque de Sorbonne Université. Une recherche de mots clés en anglais nous a semblé utile pour élargir nos domaines de réponses et en effet, les mots "Iot" et "Security issues" nous ont permis d'accéder à cette revue. Sa date de publication, assez récente (2018), nous permet de garantir l'actualité des informations données. Étant écrite par quatre auteurs qui, après approfondissement des recherches, proviennent tous d'un milieu informatique (certains provenant d'un département informatique de l'Université nationale d'informatique et des sciences émergentes du Pakistan, un autre du département de génie logiciel et des communications, de l'Université Hongik en Corée), nous pouvons nous assurer une certaine fiabilité des informations. De plus, l'article a été révisé un mois après son écriture ce qui nous confirme cette fiabilité du travail.

Le document nous offre une analyse détaillée des mécanismes de sécurité de plusieurs types de système IoT, ceci va nous être utile pour la compréhension du fonctionnement de ces mécanismes ainsi que pour la détection de leurs vulnérabilités. Chaque point essentiel a été détaillé avec des mots compréhensibles par un grand public, qu'il soit compétent dans ce domaine ou non. En effet, les éléments compliqués sont expliqués méticuleusement. Enfin, les trois dernières pages de la revue ont été dédiées aux références qui, après vérification, traitent bien du sujet de l'Internet of Things et de sa sécurité. De plus, elles sont toutes complétées d'un lien menant vers la référence en question. La plupart des références proviennent du site Google Scholar qui est un service de Google permettant la recherche d'articles et de publications scientifiques et universitaires. Cet article a été soutenu par le National Research Foundation of Korea (NRF) qui est une fondation qui soutient la recherche de nouvelles théories pour l'avancement de la science, de l'informatique notamment, ce qui nous prouve une certaine fiabilité de la recherche. De plus, la section "Conflicts of Interest" indique qu'il n'y a eu aucun conflit d'intérêt concernant cette publication.

Plus généralement, l'article a été publié sur la revue scientifique Sensors publiant des articles en libre accès de recherche dans le domaine de la science et la technologie des capteurs. Ce journal fait preuve de beaucoup d'attention sur la fiabilité de ses articles puisque ceux-ci sont évalués par des pairs, et un compte rendu détaillé leur est fourni après l'évaluation. Après exploration des avis des auteurs des articles à propos du journal, nous remarquons que la majorité voire la totalité des avis sont positifs. Nous pouvons en déduire que Sensors a une bonne réputation auprès des chercheurs. En conclusion, cette revue universitaire nous semble être une source fiable et complète.

Amélioration de l'identification du type des objets connectés par classification supervisée [11]

Ce document est une communication à un congrès qui a été trouvé par le biais d'une recherche sur le moteur de recherche "Google" via les mots clés "Identification des objets connectés dans un réseau". Les informations sont adaptées à un congrès, public composé de chercheurs, ce qui nous permet d'affirmer la précision et la fiabilité de ce document. Il provient de l'archive ouverte pluridisciplinaire HAL, site de dépôt d'articles scientifiques, de recherches et de thèses venant d'établissements d'enseignement, ou encore de laboratoires de recherche publics ou privés. De plus, ce site met un point d'honneur à la vérification de la fiabilité des informations puisque chaque article publié doit passer par une étape d'inspection par une équipe de modération. Ces informations permettent d'affirmer le sérieux du site de dépôt.

Le document a été publié en 2019, date récente nous garantissant l'actualité des informations et des expériences menées, par trois auteurs : Nesrine Ammar et Ludovic Noirie ont travaillé en tant qu'ingénieurs de recherche dans le laboratoire Nokia Bell Labs, laboratoire spécialisé dans le domaine du réseau. Sébastien Tixeuil travaille quant-à lui au laboratoire d'informatique de Sorbonne Université. De plus, Nesrine Ammar a été l'auteure d'une thèse sur l'identification des IoT autonomes. Le parcours de ces auteurs ainsi que leur connaissance sur le sujet des IoT nous confortent dans la solidité des informations du document.

Ce document repose sur une recherche d'identification des objets connectés via des protocoles réseaux au préalable, il est donc une suite à une recherche antérieure et permet ainsi de la compléter. Un résumé de cette recherche antérieure est d'ailleurs décrit dans la deuxième partie du document, et nous permet ainsi de comprendre le but des algorithmes décrits. Cette recherche aborde les algorithmes de classification des objets connectés et permet ainsi de choisir un algorithme qui permet une classification précise. En effet, il nous est informé que la précision des résultats obtenus est égale à 99% en moyenne, ce qui nous conforte dans la fiabilité des expériences menées. Le sérieux des auteurs du document est aussi perceptible dans la conclusion. En effet, il est précisé que des recherches ultérieures allaient être menées sur un jeu plus important de données par l'ajout d'autres objets connectés. Enfin, les références citées sont bien en lien avec le sujet décrit.
En finalité, ce document nous semble être une source fiable et complète.

7.2 Annexe 2 - Extraits de codes commentés

7.2.1 Extrait numéro 1 :

```python
def list_op_dhcp(cap, L, access_p):
    """
    Fonction qui associe aux appareils leur champ HostName du protocole DHCP si
    ils en ont émis. Liste les HostName des appareils WIFI dans une nouvelle
    liste.

    Parameters
    ----------
    cap : Liste des trames DHCP de la capture.
    L : Liste des appareils Ethernets
    access_p : Adresse MAC du point d'accès.

    Returns
    -------
    Liste des appareils associés à leur HostName ainsi que la liste des Hostname
    des appareils WIFI.

    """
    done = []
    dhcp = []
    for pkt in cap:     # On parcourt l'intégralité des trames dhcp de la capture.
        try:
            if access_p == pkt.bootp.hw_mac_addr and pkt.bootp.option_hostname not in dhcp:     # Si cette trame provient du point d'accès et que nous
                dhcp += [['hn', pkt.bootp.option_hostname]]                                     # avons pas encore croisé cet Host Name, on le stocke.
            else:
                for i in L:     # on parcours la liste de nos appareils
                    if i[0] not in done and i[0] == pkt.bootp.hw_mac_addr:      # On cherche de quel appareil cette trame provient en comparant
                        i += [['hn', pkt.bootp.option_hostname]]                # les adresses MAC puis on ajoute l'Host Name à son appareil.
                        done += [pkt.bootp.hw_mac_addr]                         # Nous stockons les appareils déjà identifié pour ne pas les refaire.
                        break
        except AttributeError:      # try/except qui permet d'ignorer les trames contenant pas de Host Name.
            pass
    return L, dhcp
```

Figure 1 : Fonction list_op_dhcp() du fichier identification.py

7.2.2 Extrait numéro 2 :

```python
def list_cve(app):
    """
    Fonction qui cherche des failles dans la base de donnée pour notre OS et
    les trie par catégorie.

    Parameters
    ----------
    app : os de notre appareil à tester.

    Returns
    -------
    Quatre listes représentant une catégorie de failles chacune pour notre OS.

    """
    i = 0
    application = []        # Création des différentes listes pour classer les failles.
    os = []
    vs_os = []
    gen_os = []
    no_vs = replace(app)
    for p in data['CVE_Items']:     # On parcourt la base de donnée.
        try:
            for y in p['configurations']['nodes']:
                for x in y['cpe_match']:
                    boinf, bosup = '', ''
                    try:                                                    # On teste si il y a des bornes de versions.
                        boinf = 'versionStartIncluding' + x['versionStartIncluding']    # Si oui on les stocke.
                    except KeyError:
                        pass
                    try:
                        boinf = 'versionStartExcluding: ' + x['versionStartExcluding']
                    except KeyError:
                        pass
                    try:
                        bosup = 'versionEndIncluding' + x['versionEndIncluding']
                    except KeyError:
                        pass
                    try:
                        bosup = 'versionEndExcluding: ' + x['versionEndExcluding']
                    except KeyError:
                        pass
                    tmp = search(app, x['cpe23Uri'])            # On cherche si on trouve des failles pour la version précise de notre OS.
                    tmp1 = search(no_vs[0], x['cpe23Uri'])      # On cherche si on trouve des failles pour notre OS sans version.
                    tmp2 = search(no_vs[1], x['cpe23Uri'])      # On cherche si on trouve des failles pour un interval de versions de notre OS.
                    if tmp[0] and x['vulnerable']:              # Avec les 3 résultats précedent, on teste si on trouve des failles et on les stocke dans
                        if tmp[1]:                              # leur liste respective. On cherche simultanément si on trouve des failles d'applications sous notre OS.
                            application += [[x['cpe23Uri'], p['cve']['CVE_data_meta']['ID'], p['cve']['description']['description_data'][0]['value'], boinf, bosup]]
                        else:
                            os += [[x['cpe23Uri'], p['cve']['CVE_data_meta']['ID'], p['cve']['description']['description_data'][0]['value'], boinf, bosup]]
                    elif tmp1[0] and x['vulnerable']:
                        if tmp1[1]:
                            application += [[x['cpe23Uri'], p['cve']['CVE_data_meta']['ID'], p['cve']['description']['description_data'][0]['value'], boinf, bosup]]
                        else:
                            vs_os += [[x['cpe23Uri'], p['cve']['CVE_data_meta']['ID'], p['cve']['description']['description_data'][0]['value'], boinf, bosup]]
                    elif tmp2[0] and x['vulnerable']:
                        if tmp2[1]:
                            application += [[x['cpe23Uri'], p['cve']['CVE_data_meta']['ID'], p['cve']['description']['description_data'][0]['value'], boinf, bosup]]
                        else:
                            gen_os += [[x['cpe23Uri'], p['cve']['CVE_data_meta']['ID'], p['cve']['description']['description_data'][0]['value'], boinf, bosup]]
        except KeyError:
            print('KEYERROR')           # try/except permettant d'ignorer les failles contenant pas les informations nous intéressant.
        except IndexError:
            print('IndexERROR')
    return application, os, vs_os, gen_os
```

Figure 2 : Fonction list_cve() du fichier search_os.py

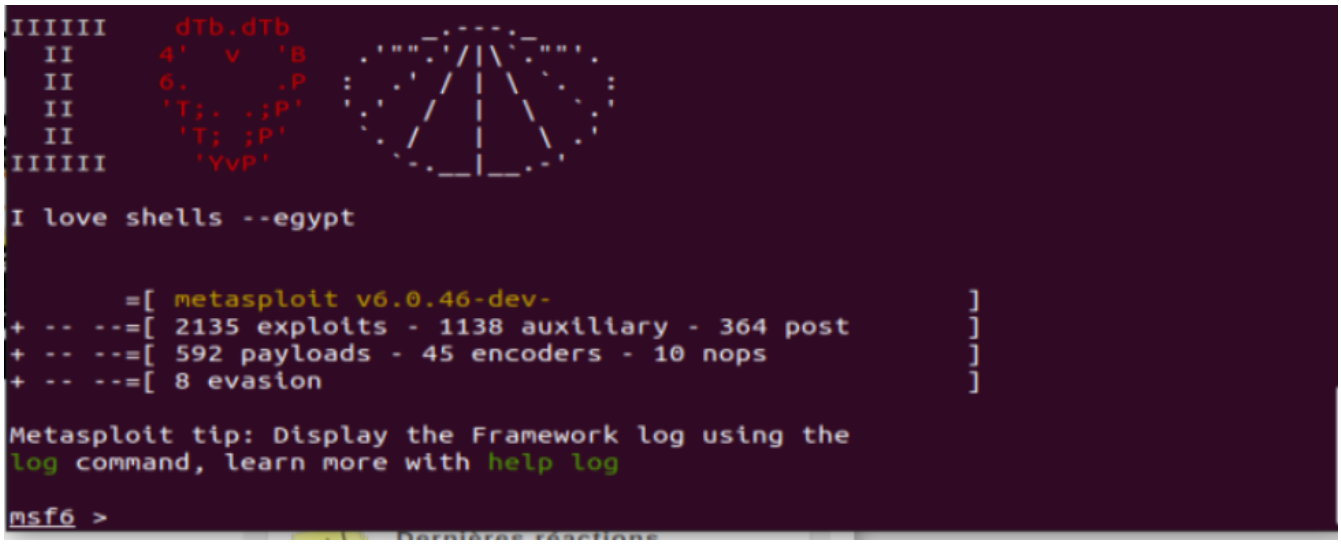
7.3 Annexe 3 - Manuel d'utilisation

7.3.1 Prérequis

Tout d'abord, avant de lancer le programme, il faut installer Metasploit. A l'aide de la commande à copier coller sur un terminal qui se trouve : `https://github.com/rapid7/metasploit-framework/wiki/Nightly-Installers` [27].

Puis il faut lancer le Metasploit à l'aide de la commande : *msfconsole* sur un terminal.

taper : yes
taper : root
taper : fgh



Affichage lors de la fin de l'installation de Metasploit.

Lorsque vous avez cet affichage vous pouvez fermer le terminal, Metasploit est installé.

Ensuite, il faut s'assurer d'avoir installé différents packages comme :
- pymetasploit3
- nmap
- pyatogui
- pyshark
- subprocess

A l'aide de la commande dans un terminal : pip3 install "nom_package"

Pour finir, il faut avoir téléchargé tous les fichiers python suivant dans le même répertoire :
- interface.py
- identifications.py
- lancement_metasploit.py
- arret_metasploit.py
- auto_metasploit.py
- search_os.py
- msfconsole.py
- msfrpc.py

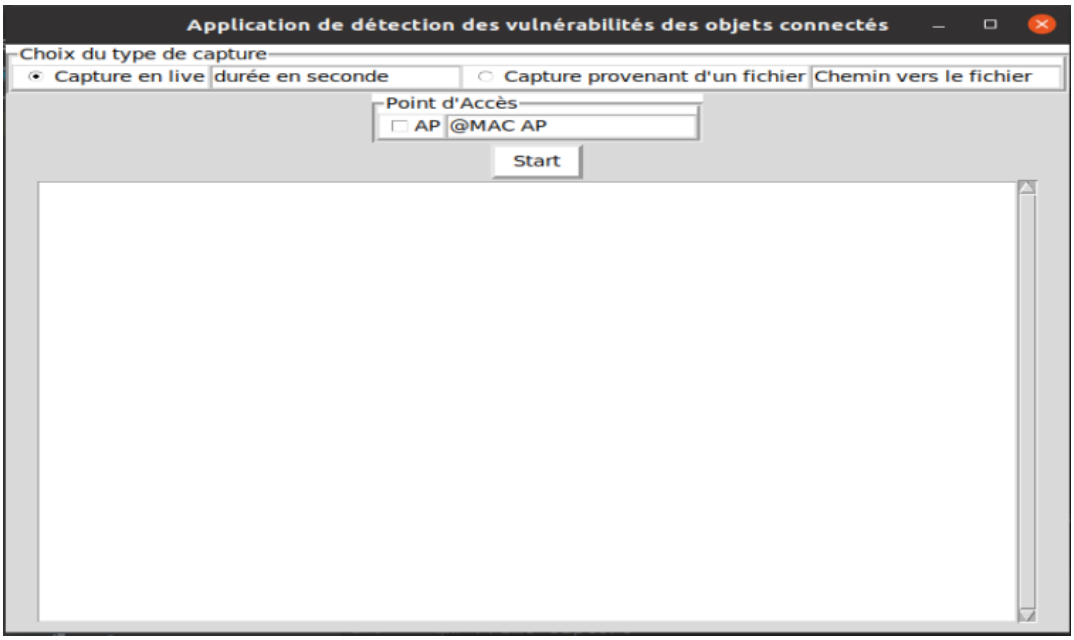
Ainsi que les bases de données json :
- macaddress.io-db.json [1]
- nvdcve-1.1-2021.json [25]

Et le fichier : user-agent.txt

7.3.2 Lancement du programme

Pour lancer le programme, il suffit d'ouvrir un terminal, de se positionner dans le répertoire contenant tous les fichiers cités précédemment et de taper la commande :

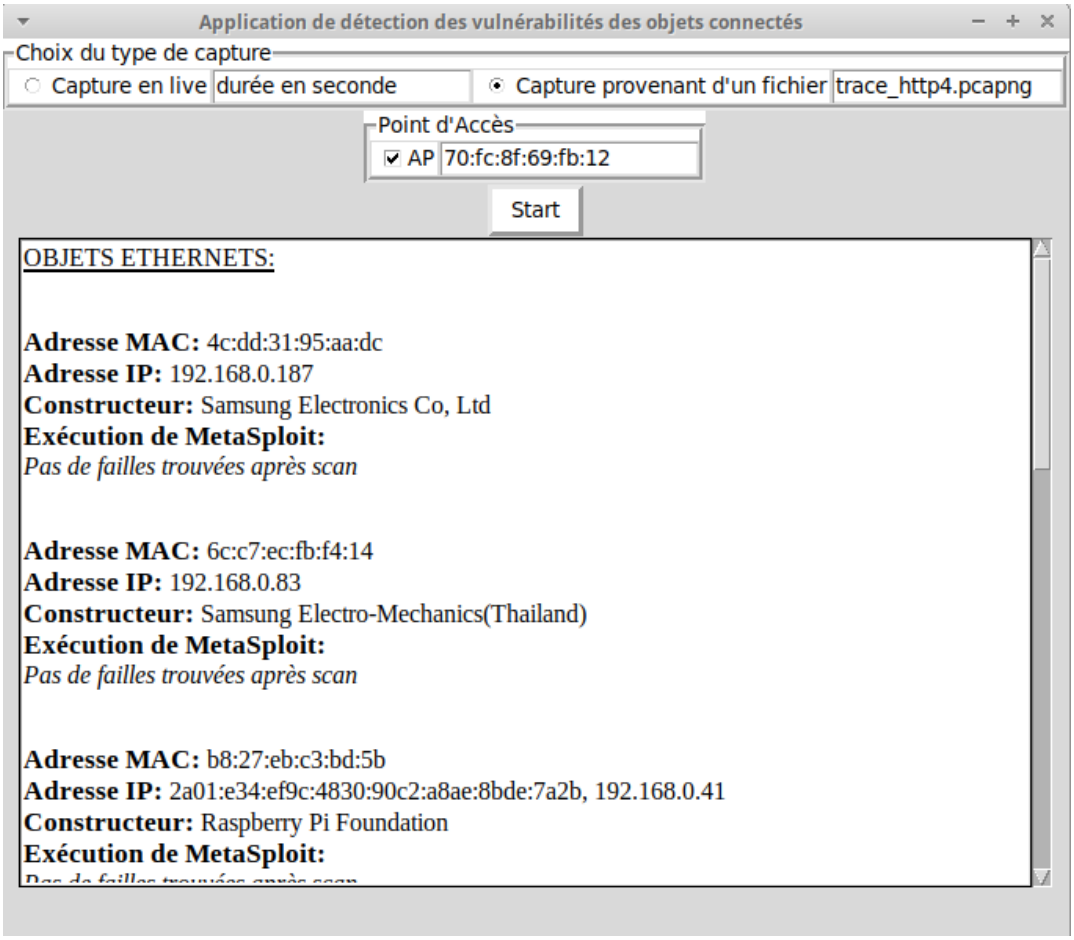
python3 interface.py



Affichage de l'interface lors du lancement du programme.

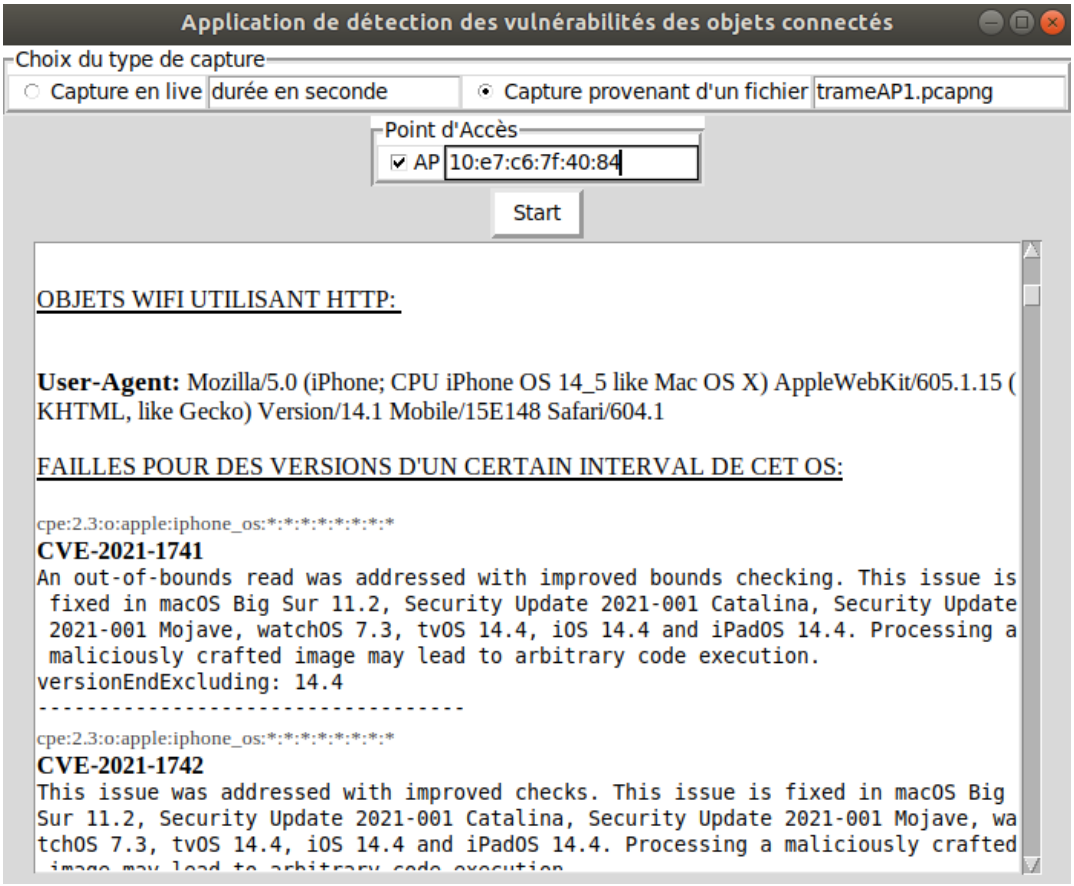
L'interface va s'ouvrir et vous proposer soit de faire une capture en live du trafic avec la durée de cette capture à indiquer, soit d'ouvrir un fichier .pcapng déjà existant et d'en

préciser le chemin.
Puis si la configuration est celle de notre conception, il faut ajouter l'adresse MAC du point d'accès et cocher la case AP.
Pour finir, il faut cliquer sur start, ne plus cliquer ailleurs et attendre.

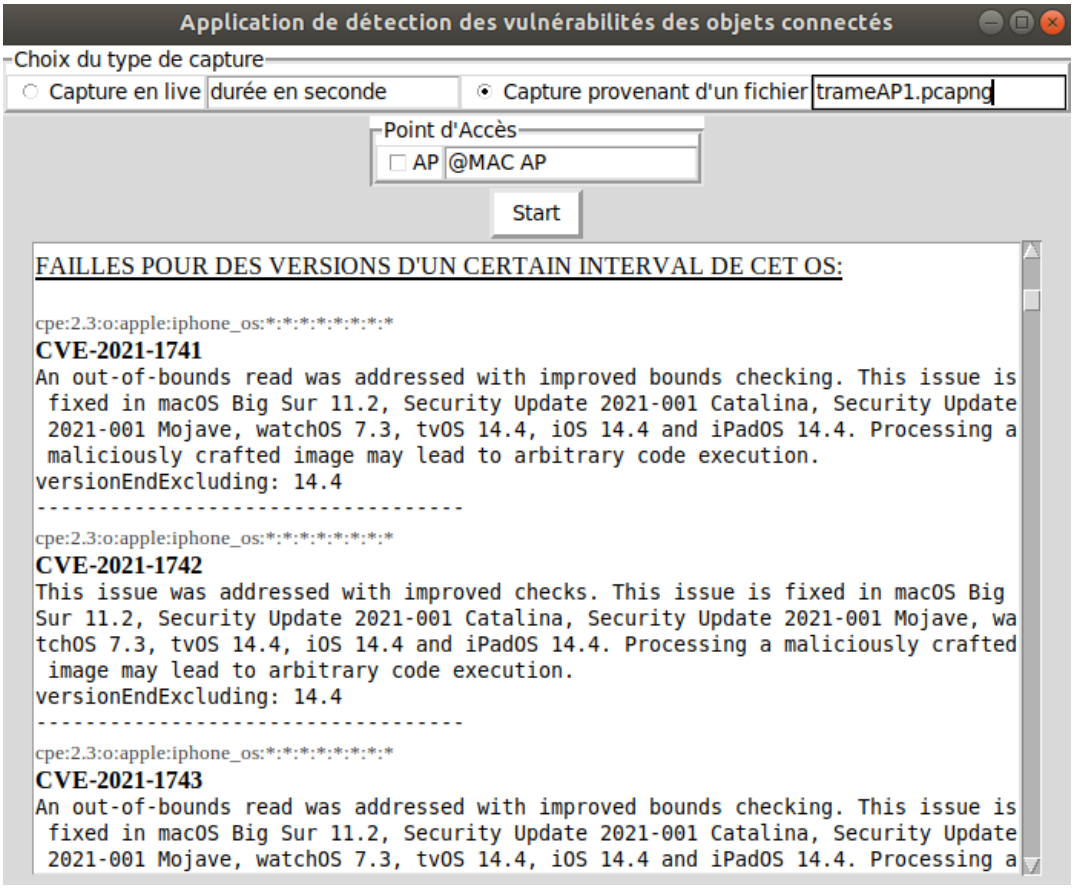


Affichage de l'interface après lancement du programme.

Remarque : ici le fichier "trace_http4.pcapng" se trouve dans le même répertoire que le code. Ainsi il n'est pas nécessaire de préciser le chemin.



Affichage des failles pour un interval de versions de l'OS.



Affichage des failles pour un interval de versions de l'OS.

A la fin vous obtiendrez les différentes vulnérabilités trouvées pour chaque objet. Il y aura les vulnérabilités propres à une version d'un OS, propres à un OS tout simplement, propres à un intervalle d'OS et propres à des applications utilisant cet OS.