Haute École d'Ingénérie et de Gestion
du canton de Vaud

Travail de bachelor

Protection des données personnelles dans le domaine de l'Internet des Objets (IoT)

Auteur
Christopher MEIER

Superviseur
Nastaran FATEMI

Mandant
NGSENS

27 juillet 2018

Résumé

Dans le monde moderne, de plus en plus de données sont fournies, consciemment ou inconsciemment, par l'individu. Il est donc important de pouvoir contrôler quelles sont les informations transmises par les appareils connectés.

Le but de ce travail est de concevoir une solution qui permet de surveiller et bloquer le partage de données personnelles. Cet objectif nécessite la résolution de trois problèmes. En premier lieu, il doit être possible d'intercepter le contenu du trafic internet et ce même s'il est chiffré. Ensuite, il faut pouvoir analyser ce qui a été récolter afin de décider si c'est une atteinte à la vie privée et pour finir il faut pouvoir bloquer les connexions qui envoient des informations sans permission.

L'interception n'est pas toujours possible, car les systèmes de sécurité sont, à raison, fait pour empêcher qu'on lise le contenu sans autorisation. Malgré cela une bonne partie du trafic peut tout de même être analysée. L'outils *Mitmproxy* permet de lire le contenu de la plupart du trafic.

Le blocage des connexions indésirables se fait par le biais d'un pare-feu et d'une liste noire d'adresses IPs. En l'occurrence en utilisant *Netfilter*.

L'analyse est plus délicate, car il y est très difficile de déterminer si une information envoyée est considérée comme privée. Il a donc été décidé de fournir un outil qui permet à un utilisateur de caractériser soi-même la nature des transmissions. L'outil correspond à une interface web créer avec *Django*, *GraphQL* et *React*.

Cahier des charges

But

Conception et développement d'une application qui révèle les enjeux de traçage des données dans le domaine de l'Internet des Objets.

Contexte

Les données de la ville proviennent aujourd'hui de trois sources principales : la municipalité elle-même ; les entreprises qui assurent des missions de service au public ou qui gèrent des « utilities » (énergie, eau) ; les habitants qui peuvent produire de la donnée en conscience sur une base volontaire ou sous forme de traces plus ou moins conscientes. C'est surtout cette dernière source qui, associée aux moyens plus traditionnels dédiés à la sécurité publique, renouvelle les possibilités de surveillance dans l'espace urbain. Les traces parfois inconscientes et passives que laissent les habitants par le simple fait d'être équipés d'un smartphone (géolocalisation), d'échanger sur les réseaux sociaux ou de recourir à des services de proximité peuvent s'avérer précieuses à des fins de surveillance.

L'ensemble des capteurs, dispositifs, traces numériques en lien avec l'espace public sont susceptibles d'être détournés de leur objectif initial à des fins de surveillance.

Problématique

La mise en correspondance des données récoltées sur l'Internet des Objets permettrait d'identifier un individu, le localiser ou récupérer des données personnelles (email, photos, etc.) à révéler ses goûts ou son style de vie.

A développer

Dans le cadre de ce travail de Bachelor, nous proposons de développer un système d'analyse et de visualisation qui permet de révéler les différentes connexions effectuées par un dispositif connecté (smartphone, TV, home caméra), et de les analyser et de visualiser la nature des données échangées. Dans le cadre de ce travail nous nous focalisons plus sur les dispositifs de type médias que sur ceux d'une autre nature.

Le système se composera de deux éléments principaux :

1. La sonde qui s'apparentera à un proxy devra être basée sur Raspberry. Soit, elle sera connectée soit en interruption de trafic soit en réplication. Il devra être possible de connecter soit des systèmes sur Ethernet ou sur Wi-Fi. Plusieurs sondes pourront récolter des informations provenant de plusieurs sources.
2. Un serveur web qui présente le type de données transmises qui pourraient être utilisées pour identifier un individu, le localiser ou récupérer des données personnelles (email, photos, etc.) afin d'être présentée à l'utilisateur. Elle devra aussi révéler si le dispositif accepte une interruption de connexion SSL.

Technologies

— Récolte des données via Raspbery Pi
— Base de données
— Développement Web
— Visualisation des données
— Choix libre du langage de programmation pour le backend et le traitement de données

Information additionnelle

Le développement sera open-source. Suivant les résultats obtenus, une présentation à la presse sera organisée.

Plannification

— État de l'art: Semaine 1 à 3
— Analyse du problème: Semaine 4 à 6
— Analyse de la solution choisie: Semaine 7 à 9
— Conception de la sonde et du serveur de visualisation: Semaine 10 à 13
— Réalisation de la sonde et du serveur: Semaine 14 à 19
— Tests de la solution: Semaine 20
— Rédaction de la documentation associé: Semaine 21

Références

[1] Cahier Innovation et Prospective, No5 : « La plateforme d'une ville – Les données personnelles au cœur de la fabrique de la smart city », 10 octobre 2017, disponible sur : cnil.fr

Table des matières

Chapitre 1

Introduction

Le filtrage et l'analyse du trafic sortant d'un réseau domestique sont importants pour permettre à un particulier de garder le contrôle sur les informations privées envoyées à des personnes tierces. Car une fois que les informations sont sur internet il n'est plus possible de garantir qui pourra y accéder.

Ce travail a pour but de fournir un outil (sonde) permettant de visualiser, d'analyser et de rediriger les flux de données qui sortent d'un réseau domestique pour aller sur internet. En particulier les flux provenant d'objets connectés.

Dans un premier temps, seuls les protocoles HTTP et HTTPS sont pris en compte. Des protocoles supplémentaires pourraient être ajoutés dans des travaux futurs.

Chapitre 2

Analyse du problème

Il y a plusieurs barrières à franchir pour pouvoir concevoir une solution au problème donnée. Chaque barrière est décrite dans une sous-section.

2.1 Interception du trafic

La première condition à satisfaire pour pouvoir analyser et filtrer du trafic est d'avoir accès au trafic. Cette section décrit les problèmes à résoudre pour avoir cet accès.

2.1.1 Redirection du trafic

Au plus bas niveau, une sonde doit recevoir les paquets à destination d'internet. Dans un réseau à diffusion, tel qu'un réseau connecté par le biais d'un concentrateur (hub), n'importe quel élément connecté reçoit une copie de tous les paquets envoyés. Mais depuis la démocratisation du commutateur (switch) la plupart des réseaux modernes sont composés de communications point à points.

Interception physique

Le moyen le plus sûr pour intercepter le trafic est de ce placer physiquement entre le commutateur ou point d'accès et le routeur.

Cette solution nécessite que les rôles de routeur, de commutateur et de point d'accès WiFi soient remplis par des appareils physiques différents. Cette séparation est rare dans les installations domestiques. Une deuxième contrainte est le besoin de la sonde d'avoir au moins deux interfaces réseaux.

Configuration du DHCP

La clé de voûte de n'importe quel sous-réseau est le routeur qui permet d'aiguiller le trafic vers sa destination dans le plus grand réseau (en général internet). Dans les réseaux configurés automatiquement par DHCP (Dynamic Host Configuation Protocol), il existe un paramètre nommé "passerelle par défaut" (gateway) qui permet la diffusion de l'adresse du routeur à tous les clients du protocole.

Une sonde dont l'adresse est utilisée comme passerelle par défaut des autres appareils du réseau reçoit tous les paquets à destination d'internet. Il est alors possible de traiter ce qui doit l'être et rediriger (ou non) le trafic vers le véritable routeur et internet. La sonde doit configurer manuellement l'adresse du routeur, sinon le trafic ne sortirait jamais du réseau domestique.

Détection de route(s) spécifique(s)

Certains systèmes peuvent outrepasser la configuration de la passerelle par défaut s'ils détectent qu'un chemin plus rapide existe pour atteindre une destination donnée. Ce comportement est rare dans des environnements non-professionnel: il n'est donc pas étudié plus en profondeur dans ce travail.

Il est également possible qu'un administrateur ou utlisateur ait configuré manuellement des routes vers internet.

IPv6

Le protocole IPv6 est le successeur de l'IPv4 standard. En plus du nombre d'adresses disponibles beaucoup plus grand, il spécifie

également des systèmes d'autoconfiguration qui permette la découverte du réseau. En particulier, il est possible de détecter le routeur automatiquement.

Utilisation d'un système de communication annexe

Certains appareils connectés utilisent un autre système que le Wi-Fi ou l'Ethernet pour se connecter à internet. De tels systèmes sont, par exemple, le réseau 4G utilisé par les téléphones mobiles ou encore des réseaux à basse consommation d'énergie tel que *LoWPAN*.

Pour intercepter le trafic de ces appareils, il faut pouvoir avoir accès aux tours qui émettent le signal. Cet accès est très difficile voire impossible à obtenir légalement pour des particuliers.

2.1.2 Le protocole HTTP

Le protocole HTTP (Leach et al., 1999) est sans état, c'est-à-dire qu'il n'est pas dépendant des communications précédentes et qu'il n'a pas d'impact sur les communications suivantes. Par contre son contenu peut garder un état.

Chaque requête reçoit en retour une réponse avec un code à trois chiffres qui permet de déterminer sa condition. Par exemple, les codes commençant par `2xx` indique que la requête s'est effectué correctement alors que les codes commençant par `4xx` ou `5xx` indiquent, respectivement, qu'il y a eu une erreur du côté de l' utilisateur ou du côté du serveur.

Les en-têtes de paquets peuvent contenir des informations supplémentaires telles que la ou les langues préférées de la réponse ou encore le format du contenu.

2.1.3 Déchiffrage des connexions TLS

Une analyse approfondie des paquets nécessite d'avoir accès à leur contenu. D'un autre côté, une des exigences de plus en plus répandue dans le web est le chiffrement systématique des transmissions. Une technique courante et sécurisée est d'utiliser le protocole TLS

(Tim Dierks , 2008) pour encapsuler les informations avant de les envoyer. Par exemple le protocole HTTPS consiste en l'encapsulation du protocole HTTP avec le protocole TLS.

Fonctionnement du protocole TLS

Le protocole TLS (Transport Layer Security) permet, en plus du chiffrement, d'authentifier la source d'un message. L'authentification (et la plupart du temps le chiffrement) nécessite que la source soit associée à un certificat. Un certificat est composé de 4 parties:

- Une clé privée, qui permet de chiffrer l'information qui sort de la source ainsi que de déchiffrer l'information qui a été chiffrée par la clé publique. La clé privée doit à tout prix rester secrète afin de garantir la sécurité d'une communication.
- Une clé publique, qui permet de déchiffrer l'information (chiffré par la clé privée) reçue depuis la source ainsi que de chiffrer l'information à destination de la source. La clé publique doit être disponible pour tout client souhaitant communiquer avec la source.
- Des métadonnées dont l'intégrité est garantie par la clé privée. Ces métadonnées contiennent entre autres: le nom, l'adresse et l'e-mail du propriétaire du certificat. Elles peuvent aussi contenir le ou les noms de domaines associés à la source.
- Une référence vers l'autorité certificative (CA) qui a signé et généré le certificat. Cette autorité affirme par sa signature électronique sur le certificat que les métadonnées associées sont correctes. Une CA est elle-même un certificat; on parle alors de chaîne de confiance (Chain of Trust). Le certificat qui commence la chaîne est appelé le certificat racine (root certificate) et il doit être installé (pour avoir une connexion sécurisée) sur chaque appareil qui souhaite communiquer avec un de ses "descendants".

Interception d'une communication sécurisée par TLS

Une communication sécurisée par TLS peut être interceptée uniquement si la sonde se fait passer pour le serveur de destination.

Elle doit donc, pour chaque destination, générer un faux certificat qui correspond au serveur et l'envoyer au client. Le client utilise le faux certificat pour communiquer avec la sonde puis la sonde utilise le vrai certificat pour communiquer avec le serveur.

Le problème provient de la partie authentification du protocole TLS. En effet, il faut faire croire au client que le faux certificat est valide. Pour ce faire la sonde doit générer un certificat racine et l'utiliser pour signer chaque faux certificat. Chaque appareil dont en souhaite pouvoir intercepter les communications protégées par le protocole TLS doit installer ce certificat racine. Malheureusement peu d'appareils embarqués permettent l'installation manuelle d'un certificat racine.

Négociation d'un cipher peu sécurisé

Avec TLS il est possible de négocier les particularités du chiffrement au début de la communication. Il serait donc théoriquement possible de choisir un algorithme de chiffrement faible qu'il est possible de craquer presque en temps réel. Mais malheureusement (ou heureusement du point de vue sécuritaire) les clients et serveurs modernes refusent explicitement de choisir des algorithmes non-sécurisés (Torsten Gigler et al., 2018). Bien sûr il est possible que certains administrateurs n'aient pas fait l'effort de garder à jour leur système.

2.1.4 Épinglage de certificat

La technique d'interception suggérée à la section précédente est bien connue. Certains développeurs ou sociétés qui souhaitent garantir qu'une communication ne soit pas interceptée font recours à une technique appelée "épinglage de certificat" (certificate pinning) (Jeffrey Walton et al., 2017).

Fonctionnement

Pour utiliser l'épinglage de certificat, une application doit avoir en local l'empreinte de chaque certificat qu'il souhaite vérifier (ou en tout cas un des certificats de la chaîne de confiance). Lorsque

l'établissement d'une connexion sécurisée est en cours, l'application calcule l'empreinte du certificat reçu et la compare aux empreintes stockées. Si l'empreinte calculée n'existe pas alors la connexion est refusée.

Pour que cette technique soit viable, l'application doit disposer d'un système de mise à jour qui permet le changement des empreintes de certificats.

Utilisation

Les navigateurs internet tels que *Chrome* ou *Firefox* utilisent l'épinglage de certificat pour certains sites très populaires tel que *Google* ou *Facebook* afin de protéger les utilisateurs.

Les systèmes *Android* fournissent une API qui permet d'épingler des certificats afin de simplifier le travail des développeurs mobile qui utilisent leur plateforme.

Conséquences

Lorsque cette technique est utilisée, la seule façon d'intercepter les communications sécurisées est en allant modifier le code (de façon statique en modifiant l'exécutable ou de façon dynamique en interceptant l'exécution à la volée) qui fait les requêtes.

2.2 Détection des atteintes à la vie privée

Un problème majeur est d'arriver à trier le trafic selon qu'il constitue une atteinte à la vie privée ou non.

Critères

Chaque personne possède ses propres critères: certaines peuvent considérer qu'une certaine information ne devrait jamais être publié alors que d'autres sont tout à fait à l'aise avec sa publication et ne la considèrent aucunement comme privée.

Des exemples d'informations qui sont souvent considérées comme privées sont entre autres: son numéro AVS, son nom de famille, son prénom, sa date de naissance, son numéro de téléphone, ses coordonnées GPS, son ou ses adresses e-mail. Ce sont des motifs qu'il faudrait pouvoir reconnaître dans le contenu de messages.

Ces types de motifs sont utilisés dans des outils ayant pour but d'empêcher la fuite d'informations confidentielles dans des mails.

Opacité du contenu

De plus, il n'est pas toujours possible d'analyser le contenu des paquets et il n'est donc pas possible d'affirmer avec certitude si une communication transmet une information considérée comme privée par son propriétaire. Il faudrait alors utiliser une notion de pourcentage: ce paquet à 25% de chance de contenir une information privée. Mais cela devient vite très arbitraire et peut manquer de sens.

Pour pallier à cette incertitude, il est nécessaire d'effectuer une analyse en utilisant les informations dont on dispose: dans ce cas cela consiste en les métadonnées liées aux protocoles. Au-delà de l'identification de la destination par l'adresse IP ou encore le nom de domaine, il est possible d'utiliser des informations tel que la taille des paquets ou encore la séquence des paquets pour extraire des classifications possibles. Par exemple le rapport entre la taille d'une requête et la taille de sa réponse donnerait une idée quant à la direction des données: est-ce que l'appareil génère de l'information et l'envoi au serveur, ou est-ce qu'il est plutôt entrain de consommer de l'information qui arrive du serveur.

Apprentissage automatique

Un chemin de solution très populaire ces temps-ci est l'apprentissage automatique (Machine Learning). Ces techniques permettent de remplacer la création d'algorithmes spécialisés par un apprentissage liés à des données. Malheureusement, ce sont ces données qui nous manquent. Il serait possible d'aller les récolter personnellement, mais il est très difficile, voire impossible, de les labelliser

en fonction de l'atteinte à la vie privée. Ce qui va à l'encontre de l'apprentissage supervisé.

Une autre technique consiste à l'utilisation d'un apprentissage non supervisé qui s'apparenterait à une notion de détection d'anomalie après une période d'apprentissage du comportement normal (accepté). Le principal souci qui découle de cette approche est qu'il est tout à fait possible et même probable que les violations de la vie privée soient complètement intégrées ou même intégrales au fonctionnement normal des diverses applications.

2.3 Pare-feu contre les infractions détectées

Une fois qu'une atteinte à la vie privée est détectée, il reste à décider quel est la réponse appropriée. On peut bien sûr avertir l'utilisateur, mais doit-on l'avertir pour chaque problème ou est-ce que l'on regroupe les problèmes de même type pour éviter la sur-notification ?

Une possibilité consiste à filtrer ou bloquer les connexions considérées comme indésirable.

2.3.1 Liste blanche/grise/noir d'adresse IP

L'utilisation de deux listes (une noire et une blanche) d'adresses IP qui sont mis à jour dynamiquement après analyse du trafic est une solution simple et rapide afin de filtrer le trafic.

La liste noire contient les adresses indésirables alors que la liste blanche contient les adresses bénignes. Tout le trafic à destination d'une adresse qui n'est pas contenue dans les deux listes est analysé plus en profondeur. Cela correspond à une liste grise virtuelle.

Limitations

Ce système pourrait laisser passer quelques paquets avant qu'il ne s'adapte aux nouveaux ajouts.

Le principal désavantage de cette méthode est le manque de granularité du filtrage. On effet il arrive souvent, surtout pour les plus petits sites, que de multiples domaines soient associé à la même adresse IP. C'est le cas par exemple des entreprises d'hébergement qui utilise des serveurs virtuels dont le contenu est dépendant du nom de domaine associé à une requête.

À l'inverse, il est également possible qu'un nom de domaine soit associé à de multiples IPs et ainsi qu'il faille multiplier les adresses IP présente dans les listes avant de pouvoir réellement considérer une communication comme bloquée (ou acceptée). Un exemple de ce type d'agrégation est les CDN (Content Delivery Network) qui permettent l'accès à un contenu distribué depuis le serveur le plus proche. Cela afin de permettre un accès le plus rapide possible.

2.3.2 Liste de noms de domaines

Un système à la fois plus global et précis que les listes d'IPs est l'utilisation des noms de domaines. Cette liste permettrait de pallier le problème des serveurs virtuels mais il n'est pas forcément possible de récupérer l'information lorsque le message est chiffré. Une extension du protocole TLS (Hopwood et al., 2006) prévoit l'inclusion d'un champ *Server Name Indication* (SNI) qui permet de spécifier le nom de domaine auquel un message souhaite s'adresser.

2.3.3 Pour chaque paquet

Le filtrage après analyse avancée de chaque paquet est plus efficace, mais il ralentit fortement la bande passante du réseau domestique.

2.4 Synthèse

Cette section a montré les éléments à prendre en comptes pour essayer de contrôler et de garantir qu'il n'y ait pas d'atteinte à la vie privée dans les communications sortent d'un réseau domestique. On y parle des problèmes d'interception du trafic, de déchiffrement du contenu et de filtrage des communications indésirables.

Le choix d'une interception par configuration du DHCP est requis car le Raspberry Pi ne dispose que d'une interface cablée[1].

Il est important de choisir un déchiffrement du contenu basé l'interception du certificat, car ce système se base sur les principes même des communications TLS. Si, au contraire, on utiliserait des faiblesses des différentes implémentations on lancerait une course contre la montre et il faudrait être à chaque instant en avance sur les corrections de sécurités.

L'utilisation d'un pare-feu IP permet un filtrage performant qui n'est pas dépendant de la présence d'informations optionnelles comme le nom de domaine.

[1] L'interface Wi-Fi ne permet pas d'atteindre tous les appareils et ne peut donc pas être considérée comme une seconde interface.

Chapitre 3

Analyse des outils

De nombreuses personnes ont participé pour bâtir les fondations sur lesquels est construit ce travail. Cette section décrit les outils tiers qui sont utilisés.

3.1 Wireshark

L'outil Wireshark permet l'interception du trafic qui passe par les interfaces réseaux de l'ordinateur sur lequel il est lancé. C'est un outil puissant qui permet de visualiser le trafic en tant que liste de paquets. Afin de faciliter la tâche à l'utilisateur, il est possible d'afficher uniquement les paquets qui correspondent à une règle donnée. Il propose également certaines fonctionnalités comme le suivit d'une connexion TCP ou encore le déchiffrage d'une conversation TLS. Par contre ce déchiffrage demande l'accès à la clé privée utilisé.

L'application repose sur la libraire de capture de paquets *libpcap*. L'outil et ses dépendances peuvent être installé sur une distribution basée sur *debian* avec la commande suivante:

```
sudo apt install wireshark-common
```

3.2 Iptables

La commande `iptables` permet de contrôler le pare-feu (appelé *Netfilter*) présent dans le noyau Linux. Ce pare-feu permet de manipuler de façon précise le comportement du système lorsqu'un paquet est reçu.

Fonctionnement général

Le pare-feu *netfilter* propose 5 points d'attaches (hooks) sur lesquels un programme peut s'enregistrer (la figure 3.1 illustre ces points d'attache):

- Le point d'attache `PRE_ROUTING` est exécuté lorsqu'un paquet arrive sur l'ordinateur mais avant que n'importe quelle décision de routage n'ait été faites.
- Le point d'attache `INPUT` est exécuté lorsque la décision de routage d'un paquet a été effectuée et que sa destination est le système local.
- Le point d'attache `FORWARD` est exécuté lorsque la décision de routage d'un paquet a été effectuée et qu'il doit être acheminé vers un autre hôte.
- Le point d'attache `OUTPUT` est exécuté lorsqu'un paquet crée localement arrive sur la pile réseau.
- Le point d'attache `POST_ROUTING` est exécuté pour tout trafic sortant ou acheminé juste avant qu'il ne soit transmis sur le réseau.

Un système de tables et de chaînes permet d'organiser les règles du pare-feu. Les chaînes de base commencent par l'exécution d'un des points d'attache. Chaque chaîne est composée d'une suite séquentielle de règles. Si un paquet donné correspond à la règle actuelle alors l'action associée à la règle est effectuée, dans le cas contraire on passe à la règle suivante dans la chaîne actuelle. Lorsqu'il n'y a plus de règles dans la chaîne on passe à la prochaine table qui contient une chaîne correspondant au point d'attache.

Une table peut supporter soit un sous-ensemble soit la totalité des points d'attache.

Une variante de l'exécutable `iptables-restore` permet l'application de plusieurs règles en un seul appel.

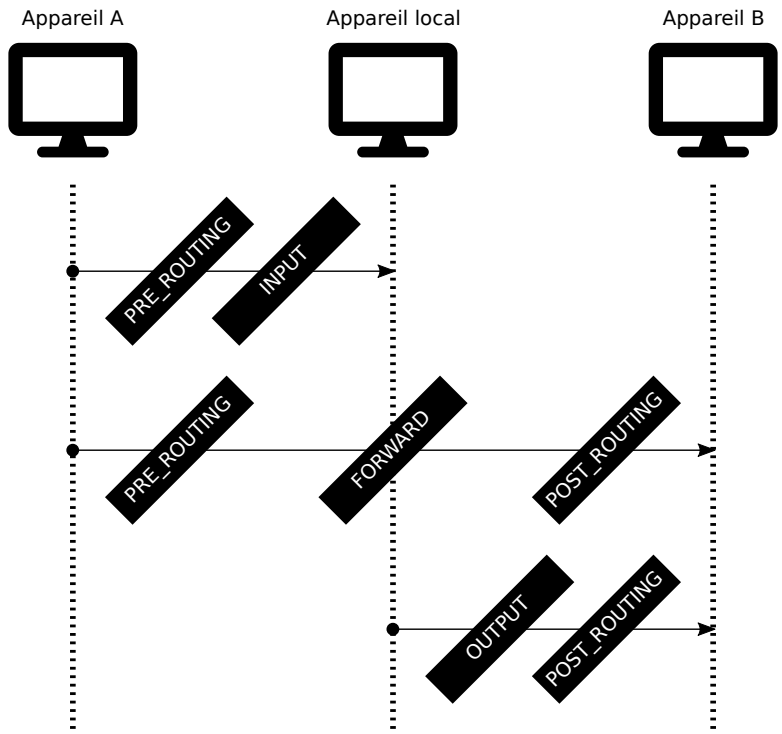


FIGURE 3.1: Points d'attaches selon la source ou la destination d'un paquet

Exemples d'utilisation

Le pare-feu détruit tous les paquets provenant de l'adresse IP `1.2.3.4`. La règle est ajoutée au début de la chaîne `INPUT`:

```
sudo iptables -I INPUT -s 1.2.3.4 -j DROP
```

Le pare-feu détruit tous les paquets sortant vers l'adresse IP `1.2.3.4`. La règle est ajoutée à la fin de la chaîne `OUTPUT`:

```
sudo iptables -A OUTPUT -j DROP -d 1.2.3.4
```

Le pare-feu log tous les paquets TCP ayant 80 ou 443 comme ports de destination dés qu'ils sont reçus. Les logs commence par le préfixe "[BLACKLISTED]".

```
sudo iptables -t nat -A PREROUTING -p tcp --match multiport \
    --dports 80,443 -j LOG --log-prefix "[BLACKLISTED]"
```

Exemple de log

Comme montré dans les exemples de commande, une des actions possible à associer à une règle est la journalisation. Un paquet qui passe par cette règle ajoute une ligne du même type que l'exemple ci-dessous.

```
May 21 19:38:26 PRIVACY_SNIFFER kernel: [27794.825978]
    [BLACKLISTED]
    IN=enxb827ebd9a69b
    OUT=
    MAC=b8:27:eb:d9:a6:9b:60:57:[...]:40:00:40:06:60:a3
    SRC=192.168.1.11
    DST=193.134.222.245
    LEN=60
    TOS=0x00
    PREC=0x00
    TTL=64
    ID=30697
    DF
    PROTO=TCP
    SPT=35266
    DPT=443
    WINDOW=29200
```

```
RES=0x00
SYN
URGP=0
```

3.3 Ipset

La fonctionnalité *ipset* est une extension d'*iptables* qui permet de garder en mémoire des listes ou ensembles d'adresses IPs dans l'espace noyau. Ce qui permet la détection rapide de la présence d'une adresse dans un ensemble donné.

Il existe plusieurs types d'ensembles disponibles. Ils peuvent être regroupé en 3 familles: les *bitmaps*, les *hashs* et les *list-sets*. Les *list-set* contiennent des références vers d'autre ensemble créé avec `ipset`.

En général chaque ensemble peut être composé – au maximum – de 65 535 éléments. Ces éléments peuvent être: des adresses IPs, des adresses réseaux, des adresses MACs, des numéros de port, des interfaces ou encore des tuples composés de ces éléments.

La mise à jour d'un ensemble IP utilisé par *netfilter* est immédiatement répercuté sur les règles qui utilisent cet ensemble. Par contre tous les ensembles IP sont perdus lorsque l'ordinateur s'éteint.

La commande `restore` d'`ipset` permet l'application de plusieurs modifications en un seul appel.

Examples d'utilisation

Créer un ensemble IP nommé `blacklisted`:

```
sudo ipset -N blacklisted iphash
```

Ajouter des adresses IPs à l'ensemble IP `blacklisted`:

```
sudo ipset -A blacklisted 1.2.3.4
sudo ipset -A blacklisted 1.2.3.5
```

Lister les adresses IPs associé à l'ensemble IP `blacklisted`:

```
sudo ipset -L blacklisted
```

Enlever une adresse IP de l'ensemble IP `blacklisted`:

```
sudo ipset -D blacklisted 1.2.3.5
```

Utiliser l'ensemble IP `blacklisted` avec une règle *iptables* afin de détruire tous les paquets dont la destination correspond à une IP présente dans l'ensemble:

```
sudo iptables -A OUTPUT -m set --match-set blacklisted dst,dst \
    -j DROP
```

3.4 Mitmproxy

Mitmproxy est une application écrite en python qui permet d'intercepter des connexions HTTP. Sa particularité est le fait de pouvoir déchiffrer les connexions TLS et d'utiliser une structure extensible.

Outils

Il existe plusieurs outils pour utiliser les fonctionnalités de mitmproxy:

- L'outil `mitmproxy` permet d'interagir à travers une console avec l'application sous-jacente.
- L'outil `mitmdump` permet d'enregistrer le trafic qui est intercepté. Il permet aussi de retransmettre le trafic sauvegardé.
- L'outil `mitmweb` met à disposition une interface web qui permet d'interagir avec l'application sous-jacente.

Chacun de ces outils ou points d'entrées correspond à une classe qui hérite des fonctionnalités principales de l'application. La figure 3.2 montre cette disposition.

Fonctionnement général

Pour pouvoir intercepter le trafic, Mitmproxy se fait passer comme un serveur au client et comme un client au serveur. La figure 3.3 montre les étapes d'une connexion HTTPS.

Le certificat de l'autorité certificative utilisé par Mitmproxy est disponible à l'adresse `mitm.it` pour tous les clients qui sont interceptés.

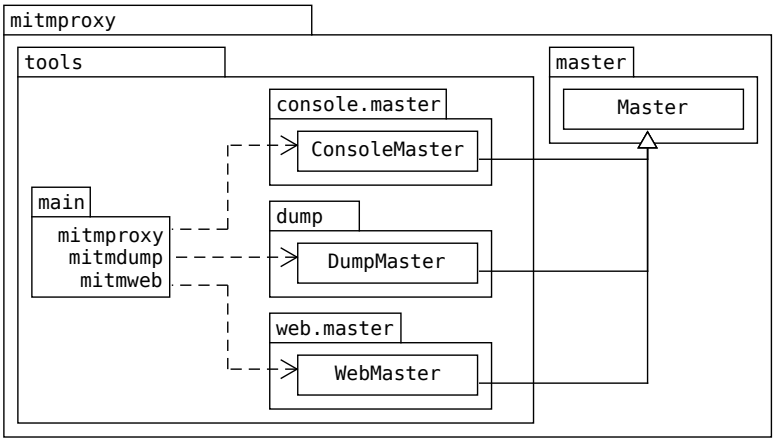


FIGURE 3.2: Point d'entrée de *mitmproxy*

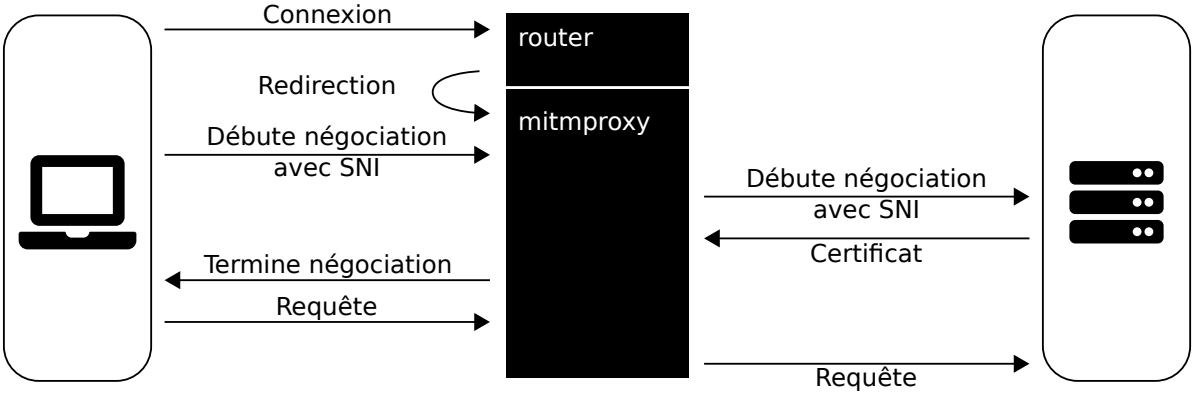


FIGURE 3.3: Fonctionnement de *mitmproxy* pour une connexion HTTP

HTTP

Mitmproxy comprend le protocole HTTP et peut donc organiser les informations du trafic pour être plus clair pour les extensions. En particulier, l'outil décompresse automatiquement les requêtes et réponses dans le contenu est compressé.

Système d'extension

Un système d'extension est fourni pour permettre aux utilisateurs (et aux développeurs) d'ajouter aisément un comportement. Ce système est également utilisé par un grand nombre des fonctionnalités standard.

L'application donne l'accès à ses variables globales grâce à un contexte passé aux extensions. Ce contexte peut être chargé avec l'importation de module de python.

Événements

Les extensions communiquent avec l'application à travers des événements. Lorsqu'un événement X est déclenché, le système vérifie pour chaque extension si elle a une méthode X correspondante et si c'est le cas il exécute la méthode.

Les événements peuvent être séparés en 5 catégories.

— Ceux associés au cycle de vie d'une connexion HTTP: `http_connect`, `requestheaders`, `request`, `responseheaders`, `response`, `error`.
— Ceux associés au cycle de vie d'une connexion TCP: `tcp_start`, `tcp_message`, `tcp_error`, `tcp_end`.
— Ceux associés au cycle de vie d'un Websocket: `websocket_handshake`, `websocket_start`, `websocket_message`, `websocket_error`, `websocket_end`.
— Ceux associés au cycle de vie du réseau: `clientconnect`, `clientdisconnect`, `serverconnect`, `serverdisconnect`, `next_layer`.
— Ceux associés au cycle de vie général de l'application: `configure`, `done`, `load`, `log`, `running`, `tick`, `update`.

Mode d'opération

Les modes d'opérations indiquent à Mitmproxy de quelle manière il est censé intercepter le trafic. Par défaut, Mitmproxy est configuré en tant que proxy HTTP. Le mode le plus intéressant pour ce travail est le proxy transparent. En effet ce dernier ne nécessite aucune configuration au niveau du client pour qu'il fonctionne. Le paramètre `--mode transparent` active ce mode.

Redirection du trafic

Mitmproxy ouvre un port sur lequel il reçoit le trafic (en général le port `8080`). Mais pour que le trafic arrive sur ce port il faut d'abord le rediriger. Les commandes ci-dessous configurent le pare-feu *Netfilter* pour effectuer cette redirection pour les ports standard de HTTP(`80`) et HTTPS (`443`). La variable `$INTERFACE` correspond au nom de l'interface réseau à utiliser.

```
iptables -t nat -A PREROUTING -i $INTERFACE -p tcp \
    --dport 80 -j REDIRECT --to-port 8080
iptables -t nat -A PREROUTING -i $INTERFACE -p tcp \
    --dport 443 -j REDIRECT --to-port 8080
ip6tables -t nat -A PREROUTING -i $INTERFACE -p tcp \
    --dport 80 -j REDIRECT --to-port 8080
ip6tables -t nat -A PREROUTING -i $INTERFACE -p tcp \
    --dport 443 -j REDIRECT --to-port 8080
```

3.5 PostgreSQL

PostgreSQL est un gestionnaire de base de données robuste et performante qui est prêt pour un environnement de production. Il est également open source et gratuit, ce qui on fait un choix idéal.

3.6 Django

Django est un framework qui permet le développement rapide d'application web.

L'environnement permet la gestion et l'accès facile à une base de données. Un système de modèle est utilisé pour accéder aux et créer des éléments dans la base de données. Des outils fournis avec Django permettent de créer les schémas dans la base de données afin qu'ils correspondent au modèle.

Il est également aisé de créer et gérer les points d'entrées (URL) de l'application développée.

Django permet également l'intégration facile d'extension créée par des tiers.

3.7 GraphQL

GraphQL est un language qui permet la requête et la manipulation de données. Il a été produit par Facebook pour une utilisation interne en 2012 puis mis à disposition de la communauté open source en 2015. Il est considéré par beaucoup comme un successeur des API de type REST.

Il permet à un client de spécifier la structure des données dont il a besoin et le serveur renvoie (si possible) les informations demandées en suivant la structure demandée. Cette capacité permet d'éviter la multiplication des requêtes.

Le language permet également la formalisation des communications entre le client et le serveur et aide ainsi au découplage de leur développement.

3.7.1 Graphene

Graphene est un serveur GraphQL écrit en python qui dispose d'une intégration à Django. Cette intégration permet de se baser directement sur le modèle de la base de données ainsi que de filtrer les données en utilisant des mots-clés.

En plus des fonctionnalités standard d'un serveur GraphQL, Graphene met à disposition – entre autres – la gestion du type Date ainsi que la gestion de la pagination de type Relay.

3.7.2 Relay

Relay est un client GraphQL écrit en JavaScript qui utilise un modèle descriptif des dépendances de donnés pour récupérer automatiquement les données lorsqu'elles sont requise.

Relay définit également un système de pagination composé d'arête et de noeud pour permettre de naviguer à travers le contenu disponible.

La prise en main est difficile, et elle n'est pas facilitée par une documentation peu claire et limitée. En particulier la mise à jour du cache local des données lorsqu'un changement a été effectué.

3.7.3 Apollo

Apollo est un autre client GraphQL écrit en JavaScript. Il utilise un système plus explicite mais qui permet de contrôler le déroulement des requêtes et manipulations ainsi que de mettre à jour le cache local.

3.8 React

React est une librairie JavaScript qui permet la création d'une interface web dynamique et interactive.

Une interface React correspond à une hiérarchie de composant affichable dans laquel chacun peut avoir son propre état et réagir au changement d'état de son parent. La syntaxe JSX permet d'insérer des tags de type XML directement dans le code JavaScript.

L'écosystème dispose de nombreux modules qui permettent de simplifier le développement.

AntD

Le module AntD met à disposition un grand nombre d'élément d'interface tel que des boutons, des formulaires ou encore des systèmes de pop-up.

Il permet la création rapide d'une interface élégante.

Victory

Le module Victory est utilisé pour la composition d'élément graphique pour visualiser de l'information.

3.9 Synthèse

Cette section a rapidement discuté des différents outils utilisés lors de ce travail. On y parle des outils d'interception utilisés ainsi que des outils de pare-feu qui permettent de contrôler le trafic réseau et pour finir des technologies requises pour la réalisation.

Mitmproxy a été choisi car il permet l'interception du contenu de la plupart des connexions sécurisé par TLS. De plus son architecture extensible permet la modification facile de son fonctionnement dans le cadre de ce travail.

Dans le monde des systèmes *Linux* le pare-feu Netfilter est la référence et est disponible nativement avec la plupart des distributions. Il a été choisi car il est très performant.

Le prototypage rapide possible avec Django, que ce soit au niveau de l'API web ou du modèle de la base de données en fait un choix idéal. De plus les nombreuses possibilités d'extensions réduisent le besoin de coder des outils que d'autres ont déjà réalisés.

L'utilisation d'une API basée sur GraphQL permet, grâce aux outils disponibles dans leur écosystème, de reduire une bonne partie du code d'échafaudage. Il permet également d'éviter la multiplicité des requêtes.

La bibliothèque React permet de séparer les différents composants de l'interface. Cela facilite leur réutilisation à d'autre endroits et augmente également la clarté du code.

Chapitre 4

Présentation de l'équipement

Dans le cadre de ce travail, un certain nombre d'équipements ont été utilisés ou considérés. Cette section décrit cet équipement et leur utilisation.

4.1 Raspberry Pi 3 Model B

Le Raspberry Pi est un ordinateur à bas coût de la taille d'une carte de crédit. Il est conçu et vendu par la fondation Raspberry Pi dans une perspective d'éducation.

Sa taille et son prix en font le choix idéal pour la création d'une sonde d'exploration.

Spécification

Malgré sa petite taille, le Raspbery Pi dispose d'un certain nombre d'interfaces:

- Un port Ethernet permet une connexion câblée avec le réseau à un débit de 100 Mb/s.
- 4 ports USB 2 permettent l'utilisation de nombreux périphériques.
- Un port HDMI pour un affichage vidéo

— Une interface Wi-Fi permet une connexion sans fil avec le réseau.
— Une interface Bluetooth Low Energy permet un appareillage sans fil.
— Un emplacement Micro SD utilisé pour le stockage du système d'exploitation et des données diverses.
— Un connecteur GPIO (General Purpose Input/Output).
— Un port Micro USB pour alimenter la carte en électricité.

Architecture

Le processeur utilise le jeu d'instructions ARMv8. Il est donc nécessaire de récupérer des fichiers exécutables compatibles ou de les compiler soi-même.

Système d'exploitation

Le système d'exploitation officiel fourni par la fondation Raspberry Pi s'appelle *Raspbian*. Ce système est basé sur *Debian*.

En plus de ce dernier, il existe un certain nombre de systèmes d'exploitation fournis par la communauté. La plupart sont basé sur le noyau Linux, mais il en existe d'autres. Par exemple, Microsoft a fourni une version de Windows pour l'internet des objets fonctionnant sur Raspberry Pi.

4.2 Boîtiers TV

Avant d'intercepter du trafic provenant des objets connectés pour analyse, il faut déjà avoir des objets connectés.

4.2.1 Google chromecast ultra

L'appareil Google chromecast permet de diffuser du contenu depuis un appareil compatible vers une télévision.

Spécifications

Le boîtier est petit (seulement 58x58x13 mm) et il ne dispose que de 2 interfaces:

- Une sortie HDMI (mâle) qui se connecte à la télévision.
- Une interface Wi-FI.
- Une entrée MicroUSB utilisé comme alimentation. Ce port peut simultanément être utilisé comme entrée d'un adaptateur Ethernet. Le bloc d'alimentation fourni possède un port Ethernet.

4.2.2 AndroidTV: Vensmile

L'appareil Vensmile (modèle S850) permet d'utiliser une télévision normale comme une télévision intelligente.

Spécifications

Le boîtier tourne sur le système d'exploitation Android 4.2 et il dispose de quelques interfaces:

- 4 ports USB 2 pour la connexion de périphériques.
- Un port Ethernet.
- Une interface Wi-Fi.
- Une interface Bluetooth 4.0.
- Une sortie HDMI (femelle).
- Une entrée d'alimentation 5V.
- Un senseur infrarouge pour communiquer avec une télécommande.

4.2.3 AndroidTV: WeTek Play II

L'appareil WeTek Play II est modèle récent qui dispose d'un système modulaire pour se connecter au réseau de télévision.

Spécifications

Le boîtier tourne sur le système d'exploitation Android 6.0 et il dispose de plusieurs interfaces:

- Un emplacement modulaire pour le tuner TV.
- Un port Gigabit Ethernet.
- Une interface Wi-Fi.
- Une interface Bluetooth 4.0.
- Une sortie HDMI (femelle).
- 3 ports USB 2 pour la connexion de périphériques.
- Une fente MicroSD.
- Une sortie A/V compatible avec les câbles jack TRRS 17mm.

4.2.4 Autres appareils considérés

Plusieurs autres appareils ont été considérés. Quelques explications sont données pour ceux utilisant les systèmes suivants.

AppleTV

L'écosystème *Apple* est très fermé. Il est nécessaire de développer sur une de leurs plateformes et nécessite souvent d'avoir une licence payante.

Tizen

Le système d'exploitation *Tizen* développé par *Samsung* à l'air intéressant. Par contre il ne semble pas permettre l'installation d'une autorité certificative (CA) personnelle.

4.3 Synthèse

Cette section a décrit les appareils utilisés dans ce travail. Ça passe du raspberry pi utilisé comme sonde aux différents boîtiers TV.

L'utilisation et l'analyse des différents boitiers TV ont permis de réaliser un grand nombre des problèmes qui font obstacles à la protection des données personnelles dans un réseau domestique.

Le choix du Raspberry Pi comme support pour une sonde rend disponible une solution à bas prix pour prendre contrôle de sa vie privée.

Chapitre 5

Conception

Cette section définit les spécificités de la solution choisie.

5.1 Fonctionnalités

Les fonctionnalités qui suivent sont celles de la première version de la solution. Elles sont regroupées dans quatres sections.

5.1.1 Capture du trafic

Capture d'information à propos du trafic

Le système peut récupérer de l'information à propos du trafic internet . Idéalement depuis plusieurs sources afin qu'il soit possible de faire une analyse centralisé.

Capture d'information chiffrée

Le système tente au mieux de déchiffrer le trafic afin de pouvoir avoir une analyse du contenu.

5.1.2 Visualisation de l'information

Visualisation du trafic

Le système dispose d'une interface qui permet à l'utilisateur de visualiser les métadonnées associées au trafic.

Les métadonnées sont, entre autres:

— L'adresse IP de destination d'un paquet
— L'adresse IP source
— Le nom de domaine (s'il est disponible)
— La taille de la requête
— La taille de la réponse
— La date de la requête
— Le temps de réponse
— Quelles catégories ont été assigné au paquet lors de la phase d'analyse
— Si le paquet appartient à une des listes (blanche ou noire)

Les figures 5.1 et 5.2 sont des maquettes de visualisation possibles. En plus de ces dernières il est également possible de visualiser:

— La liste des paquets.
— La liste des adresses IPs de destination.
— La liste des tags associés avec un camembert.
— Une comparaison entre la taille des requêtes et la taille des réponses.

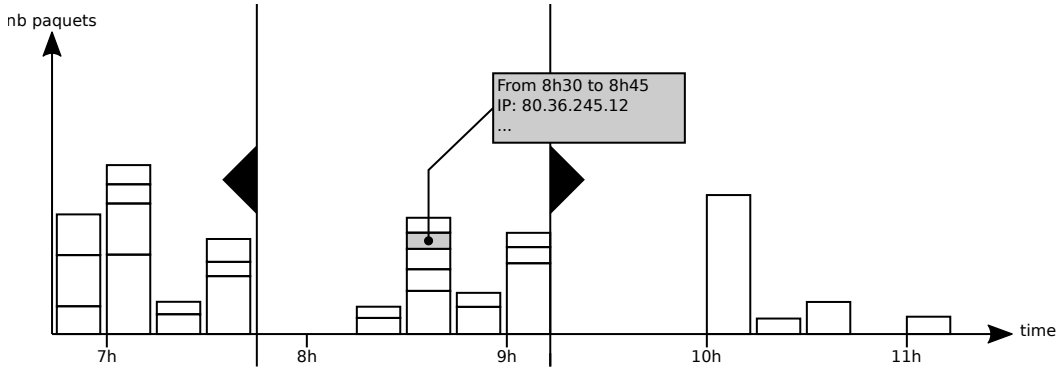
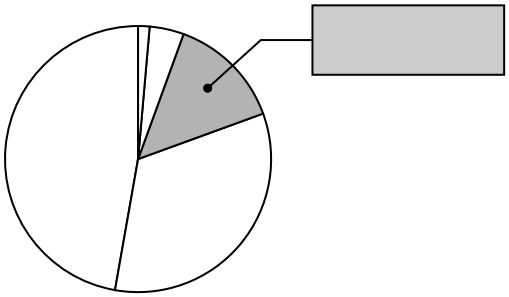


FIGURE 5.1: Visualisation du nombre de paquets à travers le temps (avec fenêtre de sélection temporelle)



	Adresses IPs	Nom FQDN	Nombre de paquets
☐	192.168.0.10	APPAREIL1.local	2
☐	192.168.0.13		6
☐	192.168.0.22	BOITIERTV	15
☐	192.168.0.39		30
☐	192.168.0.124		45

FIGURE 5.2: Visualisation des sources du trafic (avec sélection)

Filtrage de l'information par critères

L'interface permet de filtrer l'information, afin d'afficher un sous-ensemble plus pertinent à l'utilisateur. Afin de simplifier l'utilisation, un filtre peut être sauvegardé.

Par exemple, il est possible de visualiser le trafic qui provient d'un appareil spécifique et dans une fenêtre de temps donnée. Et ce en sélectionnant les éléments souhaités dans les visualisations associées.

Détection de la différence entre des fenêtres de captures

Un outil simplifie la visualisation des différences entre des captures du trafic. La capture de référence se base sur un sous-ensemble (filtre) défini par l'utilisateur.

La différence est calculée au niveau des adresses IPs.

Annotations utilisateurs

L'interface permet à un utilisateur d'insérer et de visualiser des annotations afin de garder en mémoire qu'une action a été effectuée à un instant donné. Une annotation est caractérisée par son contenu, la date et l'heure associée, l'adresse IP source associé et un système de tag personnalisé. Les tags permettent par exemple de regrouper les annotations qui correspondent à une application donnée.

Un système d'annotation rapide est également disponible pour rendre l'insertion facile et immédiate.

5.1.3 Analyse du trafic

Modules d'analyse du contenu

Le système permet l'ajout de module d'analyse. Ces modules permettent d'ajouter des catégories aux métadonnées d'un paquet et serait associé à une hiérarchie de tags.

Par exemple, un module pourrait indiquer si un paquet contient des informations bancaires.

Configuration des modules d'analyses activés

L'interface dispose d'un panneau qui permet la configuration de quels modules d'analyses sont activés.

5.1.4 Blocage de connexions indésirables

Blocage de connexion par adresses IPs

Le système permet de définir une liste d'adresses IPs indésirables. Lorsque la sonde reçoit un paquet dont l'adresse de destination appartient à cette liste alors le paquet est redirigé vers un faux site. Ce faux site renvoie en réponse un code d'erreur.

Dans une future version ...
Il serait intéressant de pouvoir changer le code ou le contenu de la réponse selon l'IP de destination ou le tag associé. Cette amélioration nécessiterait une multiplication des règles de filtrage et de faux site. Cette fonctionnalité serait configurable dans l'interface.

Configuration simplifiée du pare-feu

L'interface dispose d'un panneau qui permet la configuration du pare-feu. Il est possible d'activer ou désactiver la fonctionnalité pare-feu ainsi que de contrôler les listes d'adresses IPs.

5.2 Architecture générale

La solution est divisée en plusieurs modules, chacun ayant un rôle précis. La figure 5.3 montre comment le trafic internet interagit avec ces modules.

Lorsque le trafic arrive au pare-feu il est redirigé selon l'adresse de destination. S'il est dans la liste noire il est redirigé avec une

réponse 404, s'il est dans la liste blanche il est redirigé directement vers sa destination finale sur internet mais s'il n'appartient à aucune liste alors il est intercepté par le module d'interception puis les métadonnées et le contenu sont envoyés au module d'analyse. Pendant l'analyse, le paquet continue vers sa destination originelle. Le module de pare-feu et le module d'analyse mettent à jour la base de données avec les informations qu'ils collectent. Pour finir, le module d'interface permet de visualiser et modifier (pour les informations de configuration) les entrées de la base de donnée.

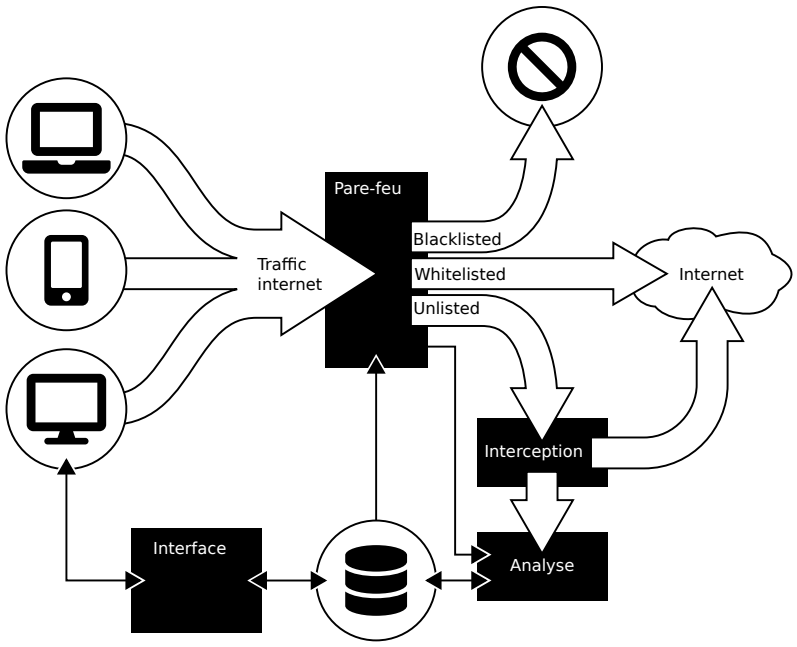


FIGURE 5.3: Diagramme du flux de trafic internet

5.3 Base de données

La base de données permet le stockage à long terme de l'information nécessaire à la solution. Plus spécifiquement, elle permet d'enregistrer:

— Les métadonnées des paquets qui ont été interceptés.
— La configuration utilisateur des options et des modules.
— Les listes blanches et noires pour le pare-feu.
— Les tags et leurs relations.

— Les annotations entrées par les utilisateurs.

La figure 5.4 présente les entités et leurs relations.

— La table `Flow` correspond à un couple de requête–réponse HTTP.
— La table `Connection` correspond à une connexion TCP.
— La table `Source` correspond à un objet connecté qui communique vers internet.
— La table `Event` correspond à une annotation d'un utilisateur. Il permet d'indiquer qu'un événement s'est déroulé à moment donné.
— Le champ `listed` indique quelle redirection doit ou a été effectué par le pare-feu.

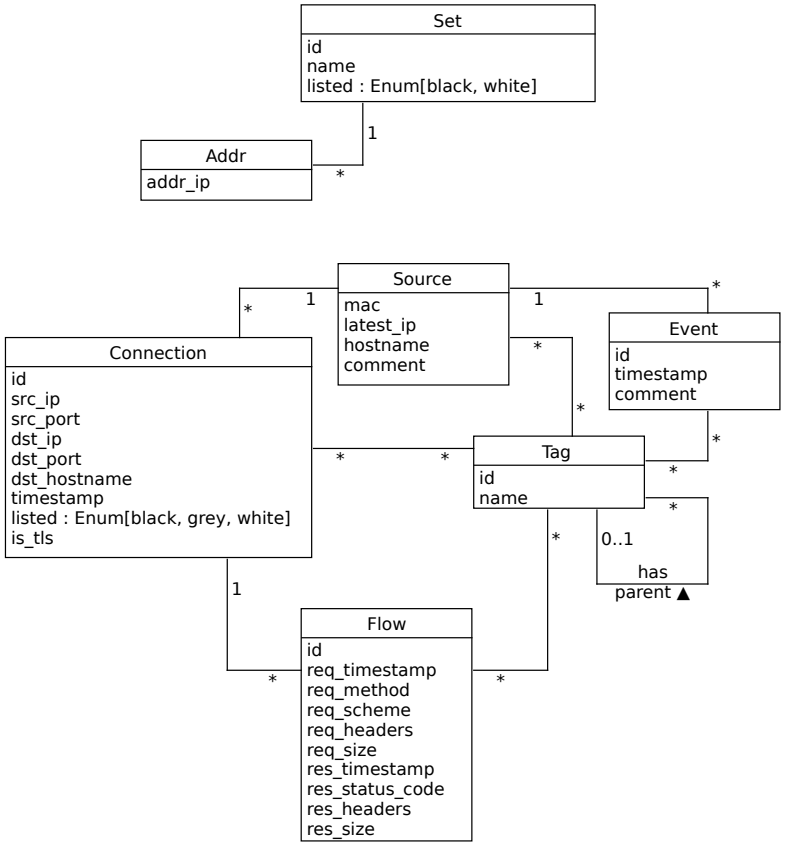


FIGURE 5.4: Modèle entité-associations de la base de donnée

5.4 Pare-feu

Le module de pare-feu permet de rediriger le trafic en se basant sur des listes d'adresses IPs. Il utilise les outils `iptables` et `ipset`.

Dans une future version ...
Il serait intéressant d'intégrer un pare-feu plus applicatif pour permettre l'utilisation de règle plus fine, d'être plus facilement configurable et d'avoir un temps de réaction plus faible.

Architecture

La figure 5.5 montre l'architecture du module de pare-feu. Le module est divisé en deux parties:

— La première partie permet de contrôler et de configurer le pare-feu (`netfilter`). En interne cela consiste à exécuté les programmes `ipset` et `iptables` avec les bonnes entrées. Afin d'éviter de nombreux appels couteux, le module utilise la fonctionnalité `restore` pour configurer plusieurs règles ou ensemble-IP en une fois.

— La seconde partie est un fil d'exécution qui écoute les journaux émits par `netfilter` puis les traduit en un format compréhensible par le module d'analyse.

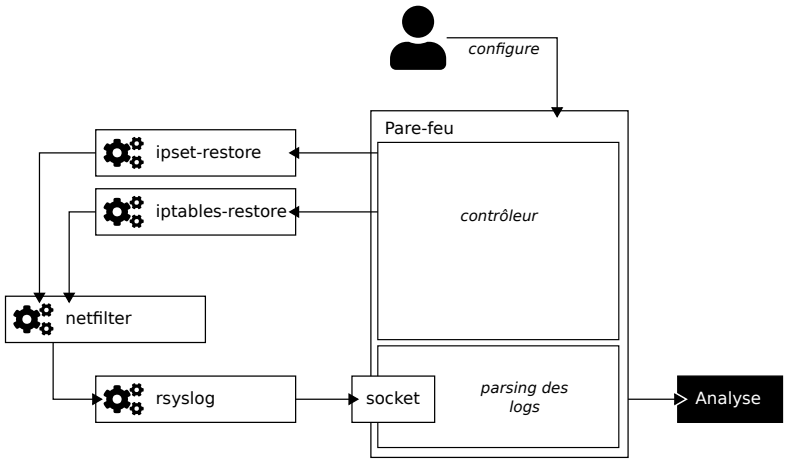


FIGURE 5.5: Architecture du module de pare-feu

Journalisation des connexions

Iptables journalise uniquement les connexions et pas les paquets.

5.5 Interception

Le module d'interception permet la compréhension des paquets HTTP ainsi que le déchiffrement (au possible) des connexions HTTPS. Il est basé sur l'outil Mitmproxy.

Architecture

La figure 5.6 montre le fonctionnement du module. Chaque connexion entre un client et un serveur peut avoir plusieurs couples requête-réponse. Au moment où une réponse du serveur est interceptée, la sonde collecte des informations à propos de la connexion, de la requête et de la réponse, les rassemble puis envoi le tout au module d'analyse.

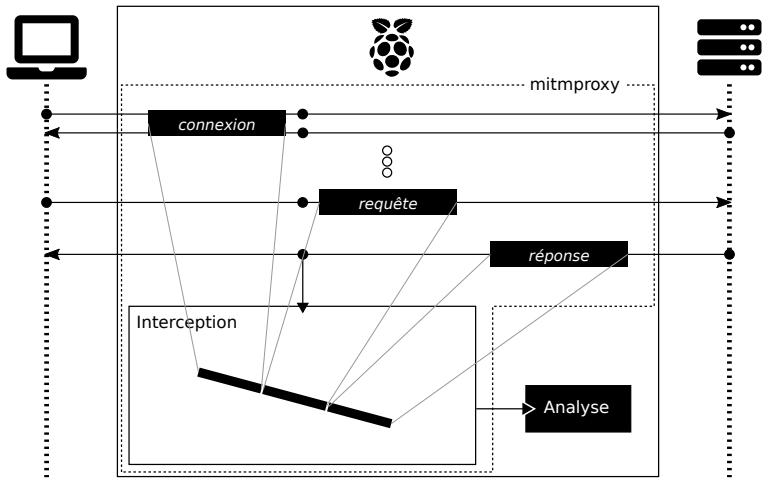


FIGURE 5.6: Architecture du module d'interception

5.6 Analyse

Le module d'analyse est composé de sous-modules qui, s'ils détectent une correspondance dans le contenu, ajoutent un tag aux

métadonnées d'un paquet.

Architecture

Le lancement du module d'analyse dans un fil d'exécution qui lui est propre permet de réduire le temps de latence lié à l'interception. La figure 5.7 montre l'architecture qui en découle. Une queue concurrente permet aux modules de pare-feu et d'interception d'envoyer l'information sans avoir besoin de se soucier de problèmes de concurrence. Ensuite chaque tuple d'information est traité un par un à travers les sous-modules activés. Une fois l'analyse terminée, les informations recoltées sont enregistrées dans la base de données.

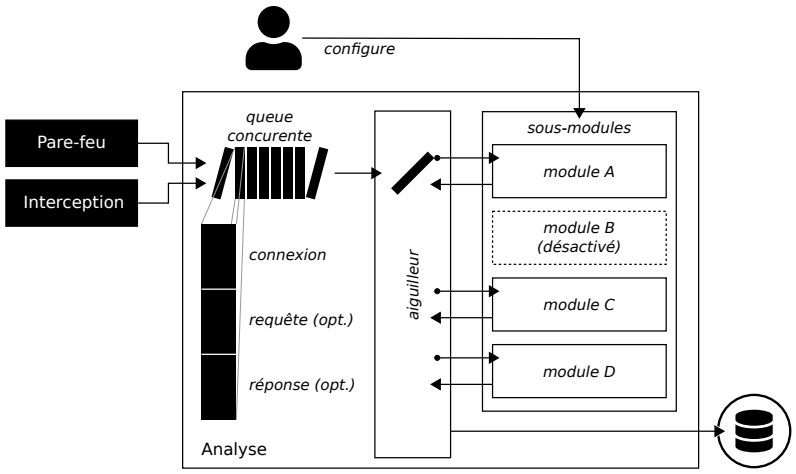


FIGURE 5.7: Architecture du module d'analys

5.7 Interface

L'interface permet à un utilisateur de visualiser ce qui se passe sur le réseau domestique et propose également des options de configurations.

La consultation de l'interface ce fait par le biais d'un navigateur internet. Ceci afin d'éviter l'installation d'un client sur chaque appareil utilisateur.

Communication asynchrone

Pour permettre une interaction plus fluide, la communication entre le client et le serveur se fait de manière asynchrone. Une API est fournie par le serveur afin de formaliser les échanges de données et de permettre une séparation des rôles.

Modules de visualisations

Chaque visualisation disponible dans l'interface correspond à un module. Un module prend en entrée une liste de couple requête/réponse et peut avoir en sortie un élément de filtrage. Si un module peut filtrer le contenu, il faut qu'il indique la nature du filtre afin que ce dernier ne soit pas appliqué sur le contenu.

Affichage des événements à résoudre

L'interface met en évidence une liste des événements qui requière une intervention de l'utilisateur. Quand c'est possible un choix d'actions est proposé.

5.8 Synthèse

Cette section a présenté les fonctionnalités offertes par la solution ainsi que les modules utilisés.

La conception de l'application peut être divisée en quatre parties. La première consiste au contrôle du pare-feu afin de décider quelles sont les adresses IPs de destination dont les paquets doivent être bloqués, interceptés ou simplement acheminés à leur destination. Des informations sur les connexions internet sont aussi récupérées.

La seconde partie va intercepter le trafic internet HTTP et permettre la compréhension du contenu et ce même si le contenu est chiffré.

La troisième partie utilise les données récoltées par les deux premières parties afin de catégoriser le contenu et enregistre le tout dans la base de données.

La quatrième et dernière partie permet à l'utilisateur d'interagir avec le système à travers une interface web. Il peut contrôler les redirections faites par le pare-feu. Et surtout il peut visualiser et analyser les données récoltées par les trois premières étapes pour prendre une décision quant à la validité des transmissions.

Chapitre 6

Réalisation

6.1 Général

Afin de pouvoir réagir en permanence au flux de trafic internet et aussi répondre à la demande lorsqu'un utilisateur souhaite contrôler l'application, il est nécessaire de séparer les tâches en deux processus. Un qui tourne constament en tâche de fond et l'autre qui se lance uniquement lorsqu'une requête arrive. La figure 6.1 illustre cette division. Le processus partage les données à travers la base de données.

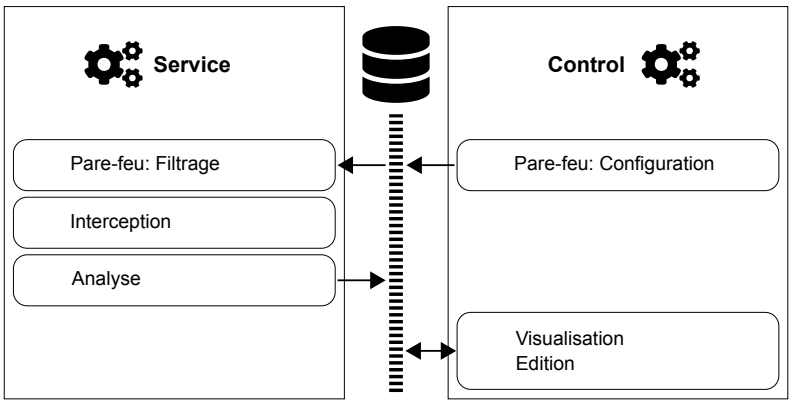


FIGURE 6.1: Les tâches qui incombent à chaque processus

6.2 Base de données

Django permet de se connecter facilement avec plusieurs types de base de données. Il suffit d'entrer les paramètres nécessaires à la connexion dans le fichier `settings.py` et le framework s'occupe du reste.

Gestionnaire de base de données

Le gestionnaire de base de données nécessite tout de même la configuration manuelle de deux points:

— Un utilisateur (avec mot de passe) est créé afin de pouvoir manipuler avec précision les droits accordés à l'application.
— Une base de données est créée afin que Django puisse y implémenter le modèle de donnée lorsque l'utilisateur exécute l'outil approprié[2].

[2] Les commandes `makemigrations` et `migrate` de `manage.py` permettent la synchronisation entre le modèle définit et les schémas dans la base de données.

SQL

Une base de données de type relationnel est appropriée à cause des relations entre les différentes entités. La figure 6.2 montre les tables et les relations créées par Django.

Modèles

Chaque module possède, s'il a besoin de stocker de l'information dans la base de données, son propre fichier de modèle. Cela évite les dépendances trop fortes entre les modules.

6.3 Pare-feu

Le module de pare-feu possède deux sections. La première est de permettre le contôle de la redirection des paquets. C'est-à-dire, permettre de choisir si pour une adresse IP de destination donné, le trafic est bloqué ou s'il passe directement vers internet ou encore s'il est intercepté pour analyse.

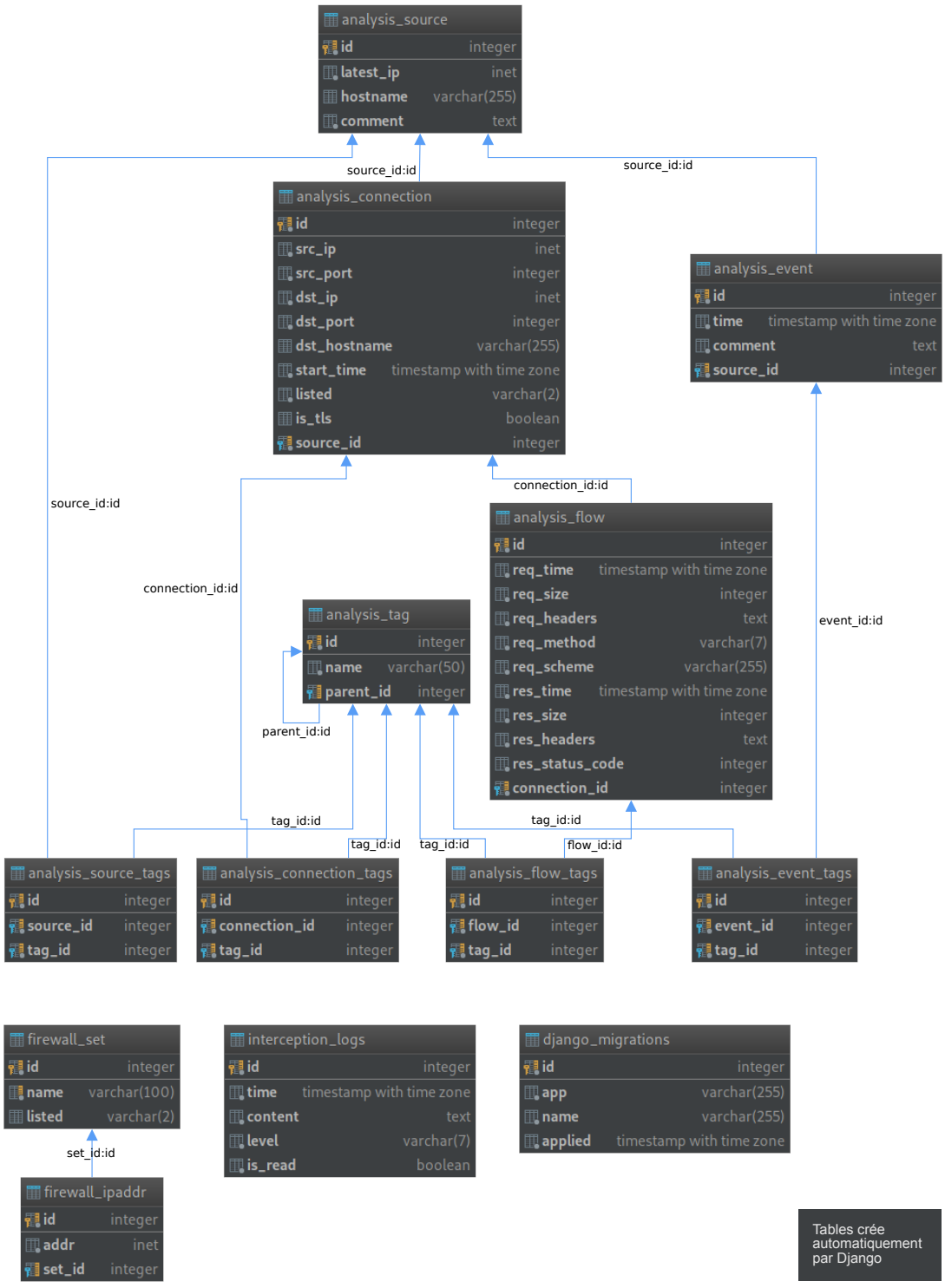


FIGURE 6.2: Les tables créées par Django à partir du modèle fournit

La seconde section tourne en tâche de fond et redirige les logs soumis par le *Netfilter* vers le module d'analyse. Les logs contiennent uniquement de l'information à propos de la connexion TCP et ne contiennent aucune information à propos du protocole HTTP.

Les sous-modules `iptables` et `ipset` permettent de gérer les exécutables du même nom. Ils possèdent également des méthodes qui permettent la mise en place de l'environnement de base (règles du pare-feu et ensembles IP) au démarrage.

Si le processus de service s'arrête de manière anormale, il est possible qu'il faille manuellement remettre en état les règles *Netfilter* (ou simplement redémarrer la sonde).

Ipset

Les fonctionnalités mises à disposition permettent l'utilisation de toutes les commandes associées à `ipset`. Afin d'éviter un éventuel conflit avec des ensembles IPs créés par d'autres programmes, le préfix `itop.` est ajouté au nom des ensembles.

Les ensembles `iotp.WHITELISTED` et `iotp.BLACKLISTED` sont réservés pour spécifier quelle redirection doit être réalisée sur le trafic internet.

Iptables

L'organisation des méthodes et classes pour gérer l'exécutable `iptables` sont génériques et devraient pouvoir être utilisé avec toutes les fonctionnalité offertes[3]. Avec l'ajout de fonctionnalité de confort d'utilisation et une légère réorganisation du code, ce sous-module pourrait être utilisé par lui-même dans d'autres projets.

[3] Il y a une exception et c'est l'instruction `--goto` , cette dernière n'est aucunement nécessaire au fonctionnement de ce travail.

Modèle

Afin de pouvoir garder en mémoire les ensembles IPs décidés par l'utilisateur ainsi que les activer au démarrage de l'application, il est nécessaire de les stocker en base de données. Un modèle est donc présent pour spécifier les informations à sauvegarder ainsi que leurs relations.

GraphQL

Une partie de l'API GraphQL est présente dans ce module. L'API permet de faire de requête et d'effectuer des changements de manière dynamique sur les ensembles IP.

Les changements sont appliqués dans la base de donné et également sur les ensembles IP présent dans le noyau (à traves le sous-module `ipset`).

L'utilisateur peut lire, créer, modifier ou détruire des ensembles IPs ainsi qu'ajouter ou enlever des adresses de ces même ensembles.

6.4 Interception

Le travail effectué par le module d'interception est fourni par mitmproxy. Il est donc uniquement nécessaire d'englober la bibliothèque en exposant un point d'entrée et une extension qui retransmet les informations aux modules d'analyse.

Point d'entrée

Une classe (`InterceptionMaster`) héritant de `mitmproxy.master.Master` est implémentée afin de pouvoir lancer l'interception avec l'extension qui permet la communication avec le module d'analyse. Cette classe est utilisée dans une autre classe `MitmproxyRunner` qui configure les options nécessaires au bon fonctionnement de mitmproxy et permet le lancement et l'arrêt de l'interception dans un fil d'exécution.

Extension

Une extension nommée `Intercept` permet l'envoi des informations concernant chaque tuple connexion–requête–réponse au module d'analyse. Une partie de l'information est pré-traitée pour correspondre au format voulu.

En plus, l'extension enregistre également les logs de l'application ainsi que leur niveau de sévérité dans la base de données afin

qu'un utilisateur puisse avoir accès à ces informations depuis une interface.

6.5 Analyse

Le module d'analyse reçoit les informations (`item`) envoyé par le module de pare-feu et le module d'interception et y ajoute des tags selon le résultat de chaque sous-modules.

Sous-modules d'analyse

Chaque sous-module doit hériter de la classe `Submodule`. La classe requiert que les méthodes `createPossibleTags` et `parse` soient surchargées. La première permet de s'assurer que les tags que l'extension pourrait appliquer un `item` soient bien présent dans la base de données. La seconde a comme paramètre le contenu à analyser et doit renvoyer quels sont les tags qui s'appliquent.

Le sous-module `email` est un exemple qui indique qu'une adresse e-mail est présente dans une requête HTTP.

Format d'entrée

La figure 6.3 décrit les informations contenues dans un `item` reçu par le module d'analyse. Les objets `requête` et `réponse` sont optionnelles. Les sous-modules doivent donc pouvoir gérer leur absence.

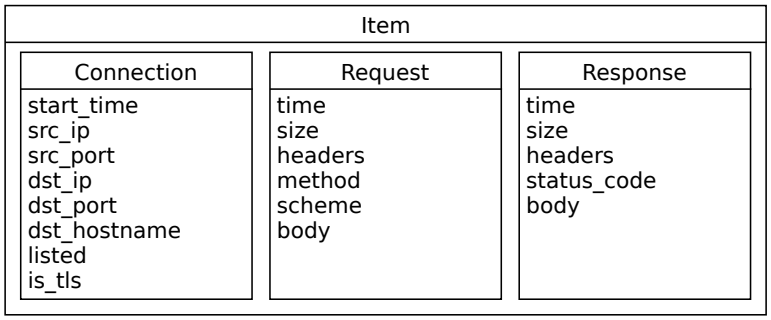


FIGURE 6.3: Modèle des entrées du module d'analyse

Activation

Pour l'instant, l'activation d'un sous-module doit se faire manuellement en changeant le script python. En effet, il faut ajouter une instance du sous-module dans la liste `submodules` présente dans le fichier `service.py`.

Service

Le fichier `service.py` présent dans le module permet le lancement du service d'arrière-plan. En plus du lancement des fils d'execution pour les modules de pare-feu et d'interception, il s'occupe également de charger les règles `iptables` ainsi que de synchroniser les ensembles IP de `ipset` avec ceux de la base de données.

6.6 Interface

GraphQL

L'API permet de faire des requêtes avec filtres et tri sur les entités suivantes: `Tag`, `Source`, `Connection`, `Flow`, `Event` et `Log`. En particulier les entités avec un champ temporel peuvent être filtrées selon un intervalle de temps.

Au niveau des modifications, l'API met à disposition des mutations pour créer, supprimer ou modifier des `Tags` ou `Events` et aussi pour modifier certains paramètres des `Sources`, `Connection`s ou `Flows`.

Les modifications d'une entité peuvent être partielles. Dans ce cas seul les champs modifiés sont mis à jour.

Client

L'interface client est divisée en 5 parties[4]:

- La partie par défaut correspond à la visualisation des données sur le trafic internet.
- La partie `redirections` permet le contrôle du pare-feu.

[4] Cette division est effectué avec la bibliothèque *React-Router*. Elle permet de choisir les composants qui sont affichés selon l'URL actuel.

— La partie `events` permet la création, modification et récupération des événements utilisateurs.
— La partie `tags` permet la création, modification et récupération des tags.
— La partie `logs` permet de récupérer les logs produits par le module d'interception.

Dù à un manque de temps et plusieurs imprévus, cette partie n'est pas terminée et ne dispose donc pas de toutes les fonctionnalités prévues.

Les figures 6.4 à 6.9 sont des captures d'écran de l'interface.

6.7 Script d'installation

Le script d'installation permet la configuration simplifiée des différents outils qui sont nécessaires au bon fonctionnement de l'application. Le script est très spécifique et fonctionne uniquement avec le système d'exploitation *Raspbian*.

Sudoers

Afin de permettre l'exécution de certaines commandes avec des droits super-utilisateurs (`ipset` et `iptables`), il est nécessaire de configurer le fichier `sudoers` afin d'autoriser leurs lancements sans demander un mot de passe. Cela évite de devoir lancer toute l'application en tant qu'administrateur.

De plus un utilisateur est créé spécifiquement pour l'application afin d'éviter de donner des droits superflus à un utilisateur lambda (celui qui installe).

Forwardings

Par défaut, le système d'exploitation *Raspbian* ne transmet pas en avant les paquets qui ne lui sont pas destinés (ceux dont l'adresse IP de destination ne correspond à aucune interface). Il est donc nécessaire de modifier ce comportement.

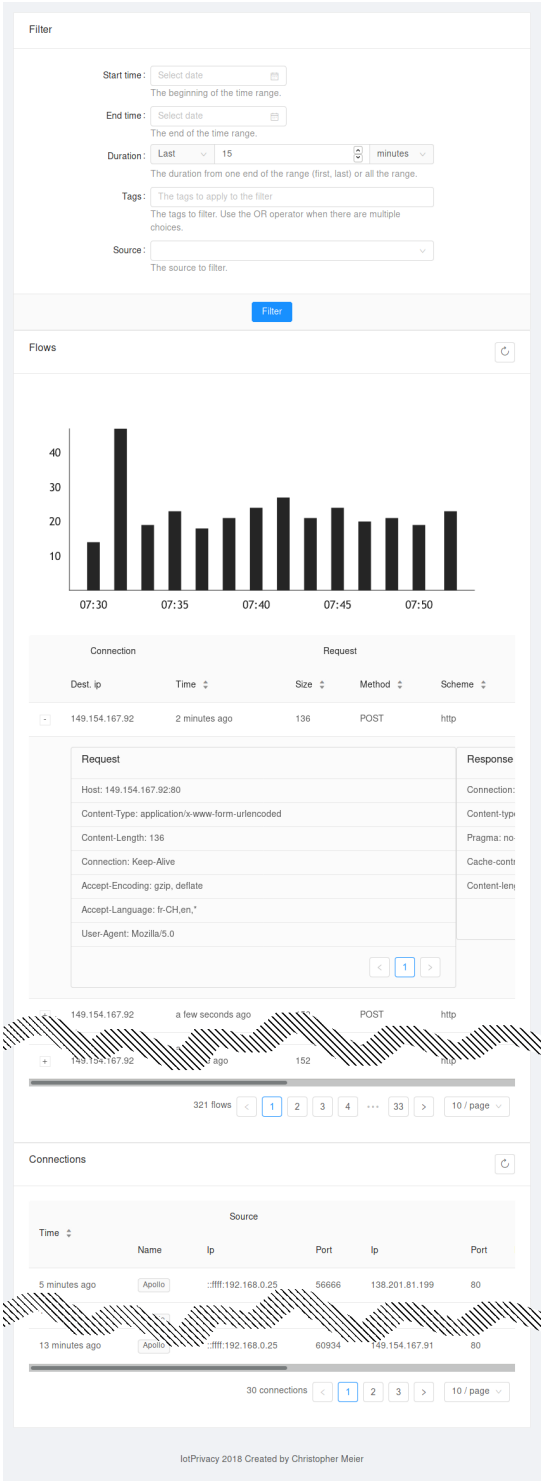


FIGURE 6.4: Capture d'écran de la partie visualisation

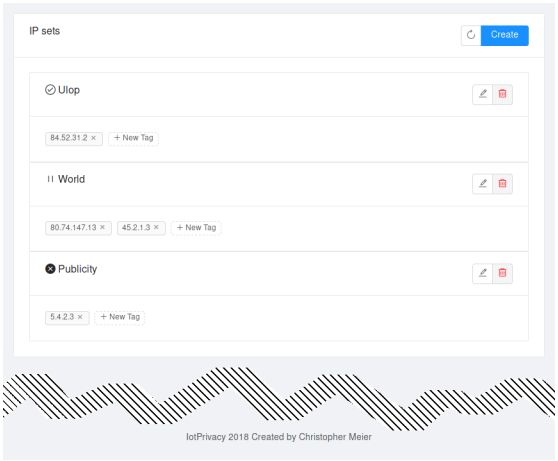


FIGURE 6.5: Capture d'écran de la partie redirection

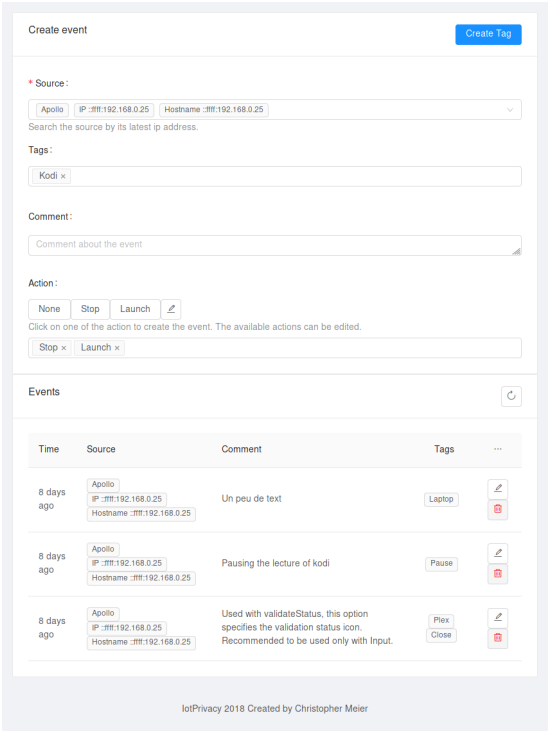


FIGURE 6.6: Capture d'écran de la partie évenements

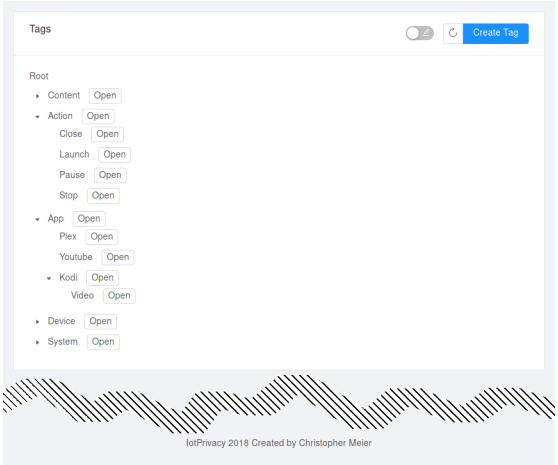


FIGURE 6.7: Capture d'écran de la partie tags (à partir de la racine)

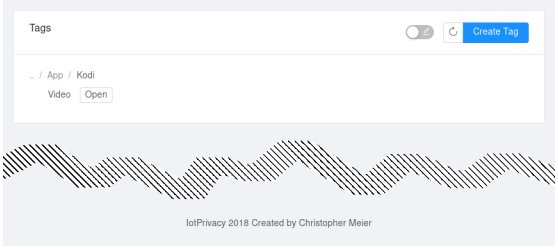


FIGURE 6.8: Capture d'écran de la partie tags (à partir d'un tag enfant)

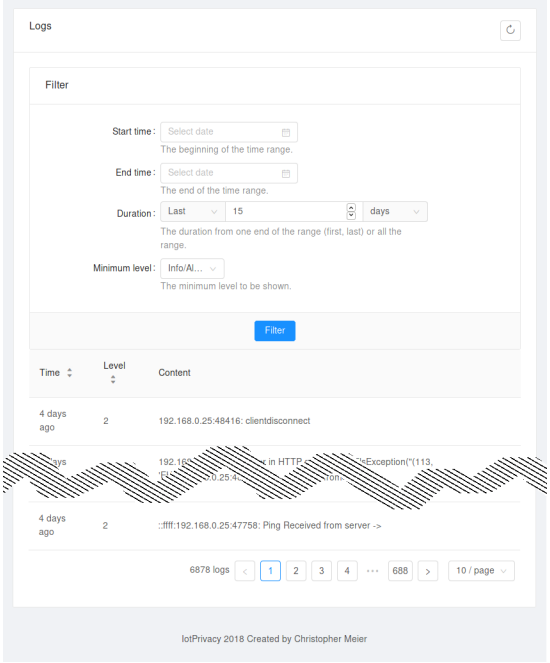


FIGURE 6.9: Capture d'écran de la partie logs

Rsyslog

Comme tous les services de *Raspbian* le pare-feu Netfilter envoie ses logs à *Rsyslog*. Afin que le service d'analyse puisse récupérer les informations des connexions qui ne sont pas interceptées par Mitmproxy, il est nécessaire de configurer *Rsyslog* afin que les messages qui correspondent à notre application y soient redirigés.

Client web

Avant de pouvoir être mis à disposition sur un serveur le code client doit être transformé et compilé.

Apache2

Le serveur Apache2 permet la mise à disposition du client web ainsi que de l'API à d'autre appareil du réseau. Pour ce faire, il est nécessaire de configurer l'emplacement des fichiers du client web ainsi que de configurer l'environnement d'exécution de point d'entrée Django.

Service

La configuration d'une unité *systemd* permet à l'utilisateur de lancer facilement le service qui tourne en arrière-plan de l'application. En bonus, cela permet également de spécifier facilement l'exécution automatique au démarrage.

6.8 Problèmes rencontrés

Une bonne partie des problèmes proviennent de l'apprentissage de nouvelles technologies.

Django

Le manque d'expérience avec le framework Django a eu comme conséquence pas mal de tâtonnement avant que la séparation en

deux processus (un pour le service et un pour le contrôle) n'ai permis de résoudre le problème des tâches de fond.

Serveur GraphQL

La prise de connaissances de meilleures pratiques concernant les API GraphQL a entrainé plusieurs réécritures du code. En particulier l'API de contrôle du pare-feu.

Mitmproxy

L'intégration de mitmproxy avec le service à donner lieu à plusieurs erreurs due à la fusion des fils d'exécution avec la programmation événementielle fournis par asyncio. De plus la résolution des problèmes dûs aux signaux de type SIGTERM, dûs fait que seul le fil d'exécution principale peut les utiliser a pris un certain temps.

Python

L'impossibilité de prendre connaissance de l'adresse MAC source d'un paquet en utilisant les socket Python a suscité un changement dans la façon de spécifier la source dans la base de données. D'un autre côté, ce changement permet l'utilisation de la sonde dans un environnement qui contient des sous-réseaux.

Client GraphQL

L'utilisation en premier lieu du client GraphQL Relay a aussi causer des délais. En effet, le manque de documentation sur la mise à jour du cache a obligé l'utilisation du client GraphQL Apollo. Ce dernier étant plus simple et mieux documenté.

6.9 Synthèse

La réalisation consiste au développement des deux processus et du client. Le processus d'arrière-plan utilise des fils d'exécution qui permettent le filtrage, l'interception et l'analyse. Il remplit la base

de données avec les nouvelles informations qui ont été analysées. Le processus de contrôle est le serveur qui répond aux demandes faite par le client selon l'API GraphQL. Pour finir le client permet un utilisateur de visualiser et de modifier les informations présente dans la base de données.

Afin de simplifier l'installation un script a été conçu qui configurent la plupart des dépendances nécessaires au bon fonctionnement de l'application.

Chapitre 7

Tests

Deux types de tests ont été effectués pour vérifier le bon fonctionnement de l'application: les tests unitaires et les tests fonctionnels.

Tests unitaires

Plusieurs tests unitaires ont été mis en place pour vérifier le fonctionnement des différentes méthodes.

Malheureusement, le manque de temps et le caractère expérimental (les meilleures pratiques étant inconnues, plusieurs façons de faire étaient essayées à chaque fois) du développement ont limité leur nombre.

Tests fonctionnels

Au fur et à mesure du développement, des tests ont été effectué pour vérifier le bon fonctionnement de chaque nouvelles fonctionnalités.

7.1 Bugs connus

— Le rafraichissement du client web lorsque l'url correspond à une route renvoi une erreur de type "page non trouvé".
— Le client n'avertit pas lorsqu'il y a filtrage sans date de début. Afin d'éviter une erreur silencieuse.

Chapitre 8

Conclusion

Ce rapport a d'abord discuté des problèmes qui peuvent barrer la route à une solution que ce soit pour l'interception du trafic, la détection des atteintes à la vie privée ou de la redirection du contenu indésirable. Ensuite il y a eu une courte présentation des outils et de l'équipement utilisé. Pour finir ce travail a abordé la question des fonctionnalités et de la conception de la solution choisie puis de la réalisation de cette conception.

Pour pouvoir effectuer ce travail, il a fallu se renseigner sur beaucoup de domaines ayant rapport à l'informatique. Tout d'abord, une bonne compréhension des protocoles réseau est nécessaire pour pouvoir intercepter et filtrer le trafic. Ensuite des compétences en développement logiciel sont essentielles au bon fonctionnement du service d'arrière-plan, du serveur et du client web. Le déchiffrement des connexions a également requis quelques connaissances en sécurité. Pour finir l'installation et la configuration de l'environnement pour l'application et l'analyse des technologies ont demandé pas mal de savoir-faire système.

Au cours de ce semestre, j'ai appris l'utilisation de nombreuses nouvelles technologies. En effet peu des outils utilisés étaient déjà maitrisés. Ce manque de maitrise à eux un coût en matière de temps consacré à l'apprentissage, mais cela permettra d'avoir de nouvelles perspectives pour les prochains travaux.

La conception et surtout la concrétisation d'une interface claire et facile d'utilisation avec des visualisations utiles et contextuelles

prend beaucoup de temps. Malheureusement le temps a manqué pour atteindre les objectifs voulus pour le client web.

La récolte d'informations est fonctionnelle, mais pour l'instant, les outils permettant à un utilisateur de se rendre compte si une atteinte à la vie privée a pris place sont minimes.

Chapitre 9

Perspectives

Dans l'état actuel, l'application réalisée peut encore bénéficier de nombreuses améliorations en plus des fonctionnalités de l'interface qui ne sont pas encore implémentées.

9.1 Améliorations possibles

— Permettre la multiplicité des sondes. Cette amélioration permettrait de centraliser les informations récoltées depuis plusieurs sous-réseaux en une analyse cohérente.
— Supporter d'autre protocole que HTTP.

Service

— Ajout du support pour l'IPv6 dans le module pare-feu et pour la configuration de la passerelle par défaut.
— Gérer la saturation possible de l'espace disque par la base de données. C'est un point à prendre en compte particulièrement à cause de l'espace en général limité des *Raspberry Pi*.
— Compléter le module de gestion de iptables pour être une bibliothèque réutilisable.
— Ajouter des scripts d'analyse. L'ajout de script permettrait d'avoir une meilleure compréhension du type de trafic qui est intercepté.

Interface

- Augmenter les possibilités de filtrage (ET, OU, NON sur les tags) afin de pouvoir faire une analyse plus fine du trafic internet.
- Améliorer les performances de l'API.
- Ajouter des visualisations à l'interface afin d'améliorer la reconnaissance de motifs ou d'éléments qui pourrait indiquer une violation de la vie privée.
- Améliorer l'expérience utilisateur pour l'interface.
- Utilisation du système de mise à jour optimiste[5] d'Apollo pour améliorer la perception de réaction pour l'utilisateur.
- Mise à jour en temps réel des informations avec les souscriptions GraphQL.
- Utilisation de fragments[6] GraphQL pour éviter la duplication de code.
- Utiliser des systèmes d'information externes pour afficher plus d'informations. Par exemple récupérer à qui appartient un nom de domaine ou une adresse de destination.
- Augmenter la sécurité en obligeant l'utilisation du protocole HTTPS.
- Ajouter un système d'authentification pour avoir accès à l'interface, afin d'éviter que n'importe qui puisse modifier le fonctionnement ou prendre connaissance d'informations qui peuvent être personnelle.
- Ajouter un gestionnaire des scripts d'analyse pour faciliter la configuration et permettre un utilisateur moins expérimenté d'installer facilement des scripts provenant de tiers.

[5] La mise à jour optimiste met à jour le cache comme si une mutation était réussie avant même de recevoir la réponse. Si la mutation échoue alors la mise à jour est annulée.

[6] Les fragments GraphQL sont des blocs réutilisables dans de multiples requêtes.

Communauté

- Un service de mise à jour des ensembles IPs pour permettre la synchronisation avec des listes publiées par des tiers.
- Un service pour installer automatiquement des extensions créées par des tiers.

Chapitre 10

Annexes

10.1 Manuel d'installation

Installation du Raspberry Pi

La première étape consiste à installer le système d'exploitation *Raspbian Lite* sur le Raspberry Pi. La procédure est décrite sur leur site web[7].

[7] http://bit.ly/2A8x30m

Configuration du Raspberry Pi

Afin de permettre la transmission de paquets à travers le Raspberry Pi, il est nécessaire de configurer une adresse statique[8].

[8] http://bit.ly/2uY77hZ

Configuration du service DHCP

Pour permettre la configuration automatique de la passerelle par défaut il est nécessaire de paramétrer un serveur DHCP. Si le serveur DHCP actuel ne permet pas de change la passerelle par défaut, il est possible d'en installer un sur le Raspberry Pi.

Le paquet `isc-dhcp-server`[9] est recommandé.

[9] http://bit.ly/2LhoJ3d

Installation de Python

Le code nécessite d'avoir une version de python supérieur ou égal à 3.6. Comme *Raspbian Stretch* ne possède pas de paquet assez récent il est nécessaire de l'installer depuis les sources[10].

[10] http://bit.ly/2JRNVrO

Installation et configuration de PostgreSQL

Le gestionnaire de base de données PostgreSQL peut être installé depuis les paquets du système:

```
sudo apt install postgresql libpq-dev postgresql-client
```

Une fois installé, il reste plus qu'a créé l'utilisateur et la base de donné. Par défaut, l'application utilise la base de données `iotp`, l'utilisateur `iotpuser` et le mot de passe `iotppass`. Bien sûr il est possible de changer ces valeurs dans le fichier `settings.py`.

```
sudo -u postgres -i
createuser iotpuser -P --interactive
createdb -U iotpuser iotp
```

Installation de Nodejs et Yarn

La plateforme Node.js[11] est utilisé pour le développement du client. Il est nécessaire pour permettre la compilation du client web.

[11] http://bit.ly/2mABtmX

Le script Yarn[12] permet l'installation et la gestion de paquets *Node.js*. Il est nécessaire pour permettre la compilation du client web.

[12] http://bit.ly/2Ob9tTB

Installation et configuration d'Apache

Le serveur Apache permet l'accès au client web et à l'API GraphQL par les ordinateurs du réseau.

```
sudo apt install apache2 apache2-dev
```

L'API mise à disposition par Django nécessite également l'installation du module `wsgi`[13] afin de communiquer avec le code python.

[13] http://bit.ly/2LixkTe

Afin de rendre le module disponible au démarrage d'Apache il faut créer le fichier `/etc/apache2/mods-available/wsgi.load` avec le contenu suivant:

```
LoadModule wsgi_module /usr/lib/apache2/modules/mod_wsgi.so
```

Récupération des sources

Le code source nécessaire est disponible soit en annexe de ce document soit depuis un dépôt GitHub[14].

[14] http://bit.ly/2uFR60M

Environnement virtuel

Un environnement virtuel python est nécessaire pour séparer les dépendances du reste du système. Une fois l'environnement créé (avec l'interpréteur python installé dans une étape précédente), il suffit d'installer les dépendances listées dans le fichier `requirements.txt`

```
cd iotPrivacy
virtualenv -p /usr/local/bin/python3.7 venv
. venv/bin/activate
pip install -r requirements.txt
```

Le script d'installation a besoin que le dossier `venv` soit au bon endroit.

Base de données

Il est nécessaire de créer le schéma de la base de donné avant que des informations puissent y être enregistrées. Pour ce faire il suffit d'utiliser l'utilitaire de gestion fourni par Django (`manage.py`) qui se trouve dans le répertoire (`iotPrivacy`).

```
# Avec l'environnement venv activé
./manage.py makemigrations
./manage.py migrate
```

Script d'installation

Un script d'installation permettant la configuration automatique de plusieurs étapes est disponible dans le répertoire `Script` sous le nom de `setup.sh`.

```
bash ./setup.sh install
```

Pour que les changements effectués par le script soient pris en compte, il faut redémarrer l'appareil.

Il est possible qu'il faille ajouter l'adresse ip de la sonde à la liste `allowed_hosts` dans le fichier `settings.py` pour pouvoir accéder à l'API.

Vérification des règles Netfilter

À cette étape il est possible de vérifier si les paquets sont correctement redirigés. Il y a deux vérifications:

- Utiliser l'utilitaire `traceroute` pour verifier que le Raspberry Pi est bien une des étapes d'un paquet se dirigeant vers internet.

  ```
  traceroute ngsens.com
  ```

  L'adresse IP statique qui a été affecté au Raspberry Pi doit être présente dans la sortie de la commande.
- Ouvrir le navigateur internet et vérifier qu'il est possible d'accéder à un site web quelconque.

10.2 Manuel d'utilisation

10.2.1 Service

Le service peut être lancé avec *systemd*:

```
sudo systemctl start iotp.service
```

Il est également possible de le lancer automatiquement au démarrage avec la commande suivante:

```
sudo systemctl enable iotp.service
```

Installation des certificats sur les appareils

Une fois le service lancé, il faut installer le certificat sur les appareils afin de pouvoir intercepter le trafic internet chiffré par TLS.

Le certificat ainsi que des instructions d'installations sont disponibles à l'adresse `mitm.it`.

10.2.2 Serveur web

Le serveur web est lancée automatiquement avec le service apache2. Si ce n'est pas le cas il suffit de le lancer avec *systemd*:

```
sudo systemctl start apache2.service
```

10.2.3 Interface web

L'interface web est disponible sur le port `8020` de la sonde. Il suffit d'entrer l'adresse suivante (en remplaçant `<Adresse IP statique>` par l'adresse IP statique de la sonde) dans la barre d'URL du navigateur de votre choix:

```
http://<Adresse ip statique>:8020
```

Sur le côté se trouve le menu qui permet d'accéder aux différentes sections.

10.3 Configuration de l'environnement d'analyse

Pour effectuer l'analyse et différents tests un environnement à été mis en place. Ce dernier utilise un réseau configuré derrière un NAT (Network Address Translation) afin d'être isolé du réseau domestique normal. L'isolation permet de garder le contrôle sur le trafic internet. Le réseau domestique se trouve également derrière un NAT. La figure 10.1 montre cet arrangement.

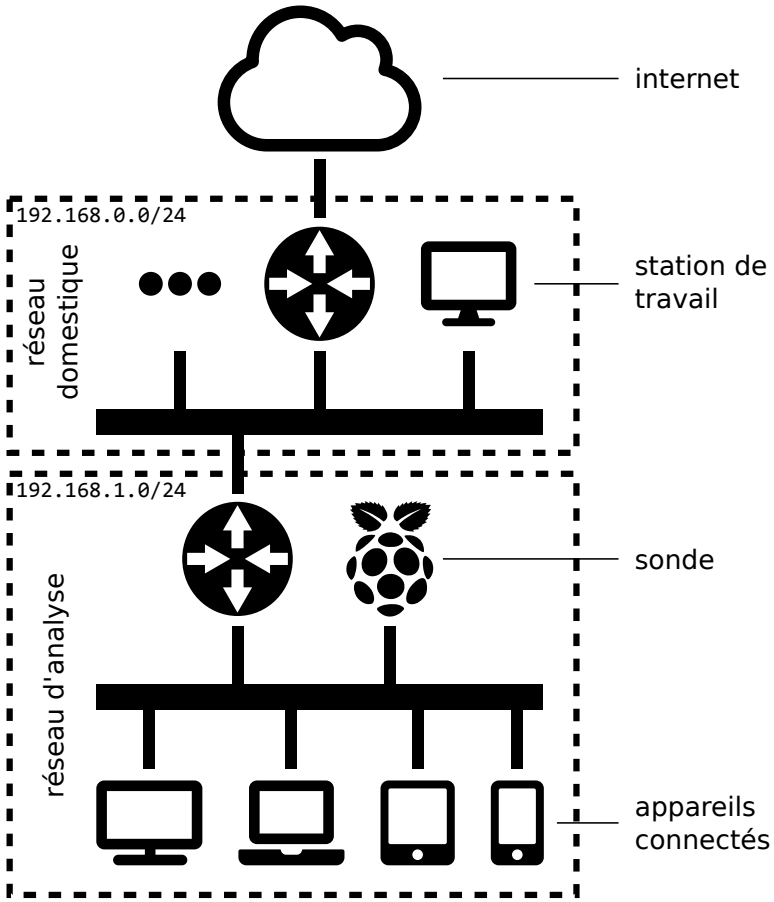


FIGURE 10.1: Diagramme du réseau d'analyse

10.3.1 Configuration du routeur

L'interface de configuration du routeur est disponible à travers le navigateur internet. Les informations d'identification par défaut de l'administrateur sont `admin` et `admin`. Une capture d'écran est disponible à la figure 10.2.

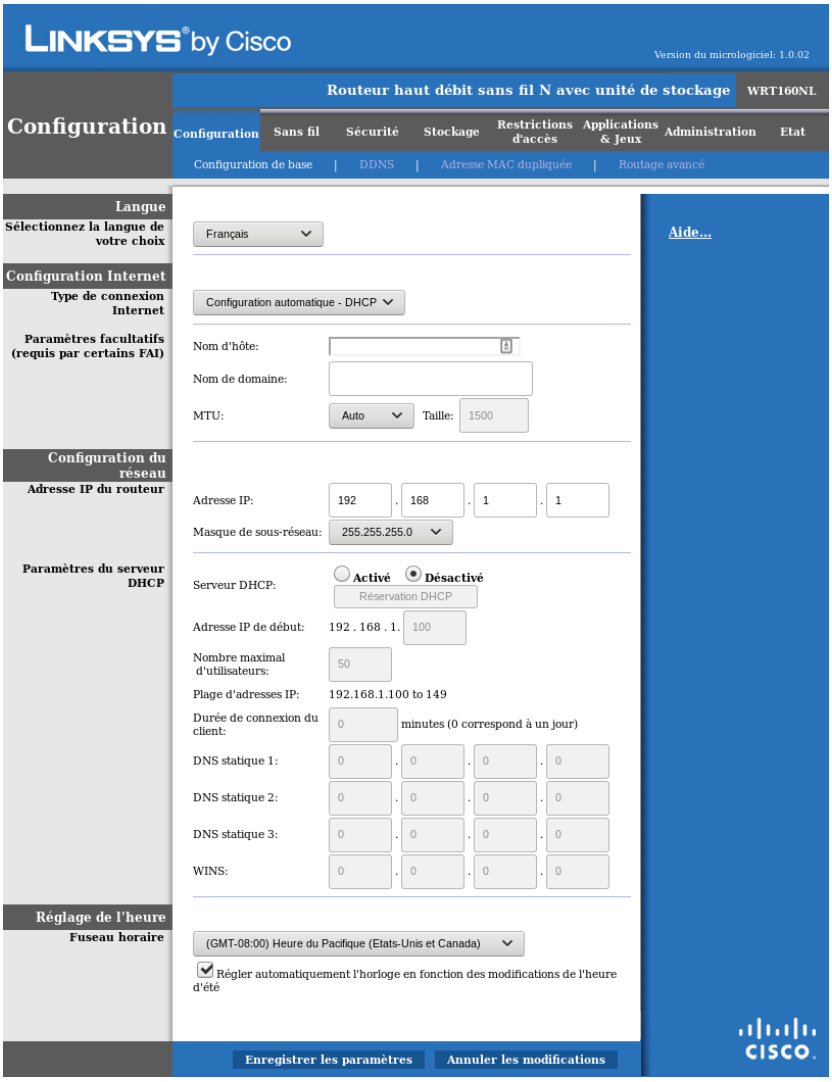


FIGURE 10.2: Interface de configuration du routeu

Clé	Valeur
Modèle	Cisco Linksys WRT160NL
Adresse IP	192.168.1.1

Clé	Valeur
Serveur DHCP	Désactivé
NAT	Activé
Nom du réseau (SSID)	TB
Sécurité sans-fil	WPA2 personnel
Adresse IP de la DMZ	192.168.1.2

La configuration de la DMZ permet à la station de travail de se connecter directement à la sonde (par exemple en utilisant SSH) sans avoir besoin de d'ajouter des règles de redirection de port.

10.3.2 Configuration de la sonde

Installation de raspbian

L'image du système d'exploitation *raspbian* est disponible sur le site web de la fondation Raspberry Pi.

La commande suivante copie cette image sur une carte SD mapper sur le fichier `/dev/sdg`. Elle nécessite d'avoir les autorisations de super-utilisateur.

```
sudo dd bs=4M if="2017-11-29-raspbian-stretch-lite.img" \
    of=/dev/sdg conv=fsync status=progress
```

Il suffit ensuite d'insérer la carte SD dans l'appareil et de brancher l'alimentation.

Configuration du système

L'utilitaire `raspi-config` permet d'effectuer les configurations de base d'un système *raspbian*.

Clé	Valeur
Nom d'hôte	`PRIVACY_SNIFFER`
Disposition du clavier	`fr_CH`
Serveur SSH	Activé

La commande suivante permet de mettre à jour automatiquement la plupart des applications installées.

```
sudo apt-get update && sudo apt-get upgrade
```

Pour configurer de manière statique l'interface réseau de l'appareil il faut ajouter les lignes suivantes au fichier `/etc/dhcpcd.conf`:

```
# static ip configuration

interface eth0
static ip_address=192.168.1.2
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Il est possible qu'il faille remplacer le nom de l'interface par son équivalent. La commande `ifconfig` permet de retrouver cet équivalent.

Configuration du serveur DHCP

Comme le routeur *Cisco* disponible ne permet pas la configuration d'une passerelle par défaut différente que son adresse IP il est nécessaire d'installer un serveur DHCP sur la sonde.

```
sudo apt-get install isc-dhcp-server
```

Il est nécessaire d'indiquer l'interface réseau sur laquel le serveur va répondre aux requêtes. Pour ce faire il faut modifier le fichier `/etc/default/isc-dhcp-server` avec le contenu suivant:

```
INTERFACESv4="eth0"
```

La configuration du serveur se fait avec des commandes inscrites dans le fichier `/etc/dhcp/dhcpd.conf`:

```
option domain-name "local";
option domain-name-servers 192.168.1.1;

subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.10 192.168.1.100;
    option routers 192.168.1.2;
}
```

La commande suivante active le lancement automatique du serveur au démarrage:

```
sudo systemctl enable isc-dhcp-server.service
```

Références

ANON., 2018. *d3: Bring data to life with SVG, Canvas and HTML. :bar_chart::chart_with_upwards_trend::tada:* [en ligne]. JavaScript. S.l. : D3. [Consulté le 26 juillet 2018]. Disponible à l'adresse : https://github.com/d3/d3.

ANON., [sans date]. Ant Design of React - Ant Design. In : [en ligne]. [Consulté le 26 juillet 2018a]. Disponible à l'adresse : https://ant.design/docs/react/introduce.

ANON., [sans date]. Django documentation | Django documentation | Django. In : [en ligne]. [Consulté le 26 juillet 2018b]. Disponible à l'adresse : https://docs.djangoproject.com/en/2.0/.

ANON., [sans date]. django-filter — django-filter 2.0.0 documentation. In : [en ligne]. [Consulté le 26 juillet 2018c]. Disponible à l'adresse : http://django-filter.readthedocs.io/en/latest/index.html.

ANON., [sans date]. Getting Started. In : [en ligne]. [Consulté le 26 juillet 2018d]. Disponible à l'adresse : https://formidable.com/open-source/victory/docs/.

ANON., [sans date]. Getting Started - React. In : [en ligne]. [Consulté le 26 juillet 2018e]. Disponible à l'adresse : https://reactjs.org/docs/getting-started.html.

ANON., [sans date]. Graphene-Python. In : [en ligne]. [Consulté le 26 juillet 2018f]. Disponible à l'adresse : http://docs.graphene-python.org/en/latest/.

ANON., [sans date]. GraphQL: A query language for APIs. In : [en ligne]. [Consulté le 24 juillet 2018g]. Disponible à l'adresse : http://graphql.org/.

ANON., [sans date]. Introduction | Apollo Client. In : [en ligne]. [Consulté le 26 juillet 2018h]. Disponible à l'adresse : https://www.apollographql.com/docs/react.

ANON., [sans date]. Introduction to Relay · Relay. In : [en ligne]. [Consulté le 26 juillet 2018i]. Disponible à l'adresse : https://facebook.github.io/relay/index.html.

ANON., [sans date]. MDN Web Docs. In : *Documentation du Web - MDN* [en ligne]. [Consulté le 26 juillet 2018j]. Disponible à l'adresse : https://developer.mozilla.org/fr/.

ANON., [sans date]. mitmproxy - an interactive HTTPS proxy. In : *mitmproxy* [en ligne]. [Consulté le 27 avril 2018k]. Disponible à l'adresse : https://mitmproxy.org/.

ANON., [sans date]. mod_wsgi — mod_wsgi 4.6.5 documentation. In : [en ligne]. [Consulté le 26 juillet 2018l]. Disponible à l'adresse : http://modwsgi.readthedocs.io/en/develop/index.html.

ANON., [sans date]. Moment.js | Docs. In : [en ligne]. [Consulté le 26 juillet 2018m]. Disponible à l'adresse : https://momentjs.com/docs/.

ANON., [sans date]. Python Documentation contents — Python 3.7.0 documentation. In : [en ligne]. [Consulté le 26 juillet 2018n]. Disponible à l'adresse : https://docs.python.org/3/contents.html.

ANON., [sans date]. React Router: Declarative Routing for React. In : *ReactRouterWebsite* [en ligne]. [Consulté le 26 juillet 2018o]. Disponible à l'adresse : https://reacttraining.com/react-router.

ANON., [sans date]. systemd/Services - Debian Wiki. In : [en ligne]. [Consulté le 26 juillet 2018p]. Disponible à l'adresse : https://wiki.debian.org/systemd/Services.

FACEBOOK, INC, [sans date]. *GraphQL* [en ligne]. S.l. [Consulté le 24 juillet 2018]. Disponible à l'adresse : http://facebook.github.io/graphql/June2018/.

HOPWOOD, David, WRIGHT, Tim, BLAKE-WILSON, Simon, MIKKELSEN, Jan et NYSTROM, Magnus, 2006. rfc4366 : *Transport Layer Security (TLS) Extensions* [en ligne]. S.l. IETF. [Consulté le 14 juin 2018]. Disponible à l'adresse : https://tools.ietf.org/html/rfc4366.

JEFFREY WALTON, JOHN STEVEN, JIM MANICO, KEVIN WALL et RICARDO IRAMAR, 2017. Certificate and Public Key Pinning. In : *OWASP* [en ligne]. 27 décembre 2017. [Consulté le 14 juin 2018]. Disponible à l'adresse : https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning.

JUSTIN ELLINGWOOD, 2015. A Deep Dive into Iptables and Netfilter Architecture. In : *DigitalOcean* [en ligne]. 20 août 2015. [Consulté le 29 mai 2018]. Disponible à l'adresse : https://www.digitalocean.com/community/tutorials/a-deep-dive-into-iptables-and-netfilter-architecture.

LEACH, Paul J., BERNERS-LEE, Tim, MOGUL, Jeffrey C., MASINTER, Larry, FIELDING, Roy T. et GETTYS, James, 1999. rfc2616 : *Hypertext Transfer Protocol – HTTP/1.1* [en ligne]. S.l. IETF. [Consulté le 15 juin 2018]. Disponible à l'adresse : https://tools.ietf.org/html/rfc2616.

PABLO NEIRA AYUSO, [sans date]. netfilter/iptables project homepage - The netfilter.org project. In : *The netfilter.org project* [en ligne]. [Consulté le 29 mai 2018]. Disponible à l'adresse : https://netfilter.org/.

TIM DIERKS , 2008. rfc5246 : *The Transport Layer Security (TLS) Protocol Version 1.2* [en ligne]. S.l. IETF. [Consulté le 4 juin 2018]. Disponible à l'adresse : https://tools.ietf.org/html/rfc5246.

TORSTEN GIGLER, MICHAEL COATES, DAVE WICHERS, TYLER REGULY et TONY HSU, 2018. Transport Layer Protection Cheat Sheet - OWASP. In : [en ligne]. 14 juin 2018. [Consulté le 14 juin 2018]. Disponible à l'adresse : https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet#Rule_-_Only_Support_Strong_Cryptographic_Ciphers.

Table des figures

Par la présente, je soussigné, Christopher MEIER, déclare avoir réalisé seul ce travail et ne pas avoir utilisé d'autres sources que celles citées dans la bibliographie.