2017

Rapport
Projet Tuteuré

Sujet :

Attaque Zombie Par Déni De Service

R&T
Département Réseaux & Télécommunications
IUT Belfort-Montbéliard

6/13/2017

Table des matières

I. Abstract du Projet en Anglais

The aim of this project was to attack by denial of service a server on a local network, thanks to a camera which was connected to it.
To do this, a hacker infiltrates the network and launches a payload to take the control of the camera. In this malware, a hacker sends two other viruses to it. They are scripts which allow the hacker to DOS the server, remotely and on another machine.

To succeed in the project we had to configurate a RaspberryPi, a Linksys Hotspot and a Virtual Machine. The RaspberryPi plays the role of the IP Camera: it will do the DOS on the Linksys Hotspot. This hotspot is the router of the local network and possess a configuration website.
Finally, the hacker needs to have a Linux session to use several tools of hacking to do the denial of service on the router's website.

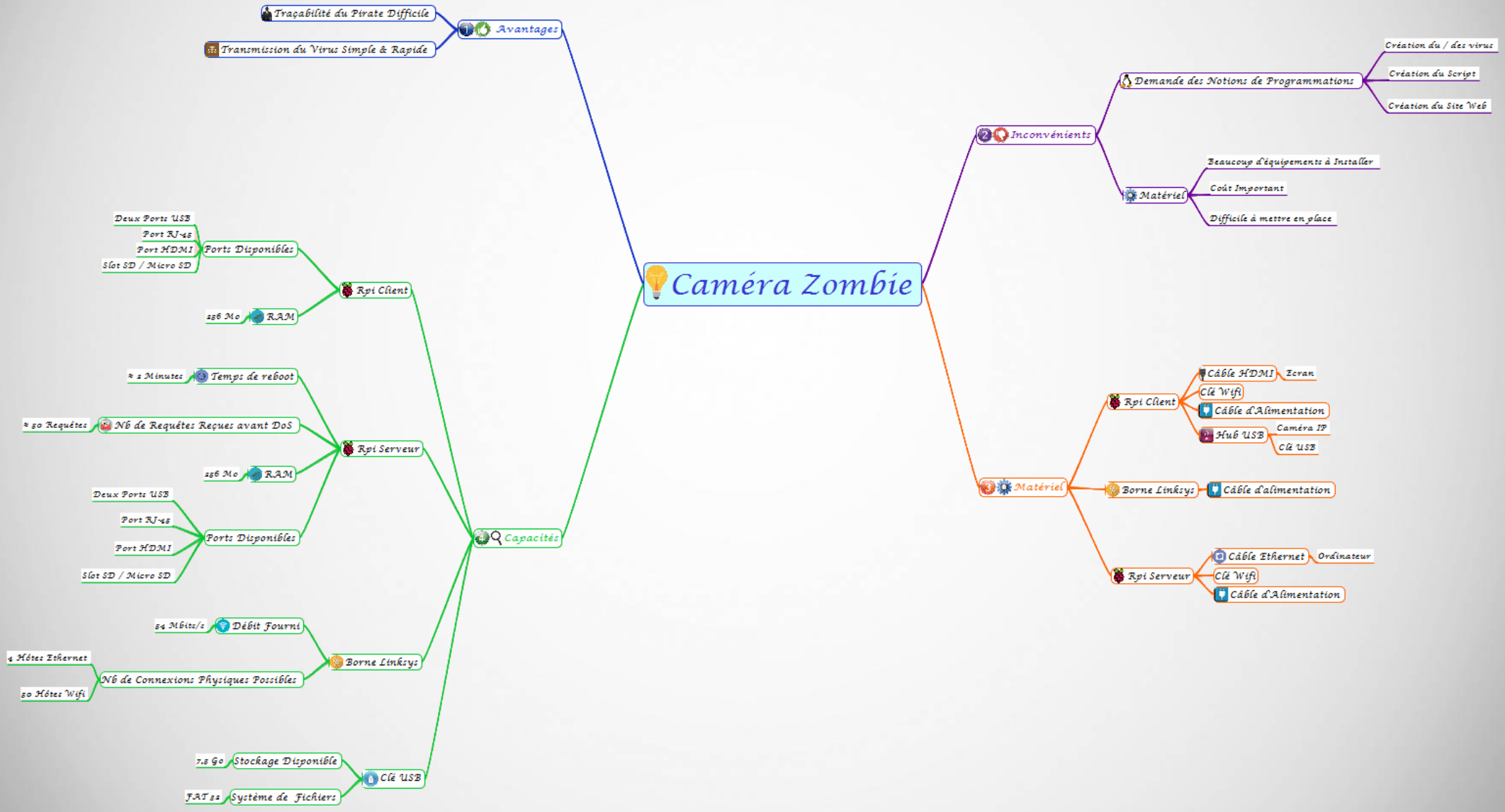
Once the payload is executed on the Linux hacker's machine, the IP Camera begins to send SYN Flood packets. Each packet contains a different source IP address and a different source port.
The Linksys Hotspot must manage a massive arrival of requests which are sent to it in a short time. Obviously, the router cannot support this attack and the configuration website becomes unavailable.

II. Description

Le but premier de notre projet était de montrer qu'une caméra classique que l'on peut distinguer dans la rue pouvait être piratée et à la merci d'un hacker qui lui aurait un total contrôle sur celle-ci. Ici, le hacker l'utiliserait afin de mettre hors service un site web. Il provoquera un déni de service grâce à la caméra. C'est pour cette raison que notre projet se nomme Attaque Zombie Par Déni De Service.

III. Freemind et Gantt du Projet

Freemind


Caméra Zombie
Avantages
Traçabilité du Pirate Difficile
Transmission du Virus Simple & Rapide
Capacités
Rpi Client
Ports Disponibles
Deux Ports USB
Port RJ-45
Port HDMI
Slot SD / Micro SD
256 Mo
RAM
Rpi Serveur
± 2 Minutes
Temps de reboot
± 50 Requêtes
Nb de Requêtes Reçues avant DoS
256 Mo
RAM
Ports Disponibles
Deux Ports USB
Port RJ-45
Port HDMI
Slot SD / Micro SD
Borne Linksys
54 Mbits/s
Débit Fourni
4 Hôtes Ethernet
50 Hôtes Wifi
Nb de Connexions Physiques Possibles
Clé USB
7.8 Go
Stockage Disponible
FAT 32
Système de Fichiers
Inconvénients
Demande des Notions de Programmations
Création du / des virus
Création du Script
Création du Site Web
Matériel
Beaucoup d'équipements à Installer
Coût Important
Difficile à mettre en place
Matériel
Rpi Client
Câble HDMI
Ecran
Clé Wifi
Câble d'Alimentation
Hub USB
Caméra IP
Clé USB
Borne Linksys
Câble d'alimentation
Rpi Serveur
Câble Ethernet
Ordinateur
Clé Wifi
Câble d'Alimentation

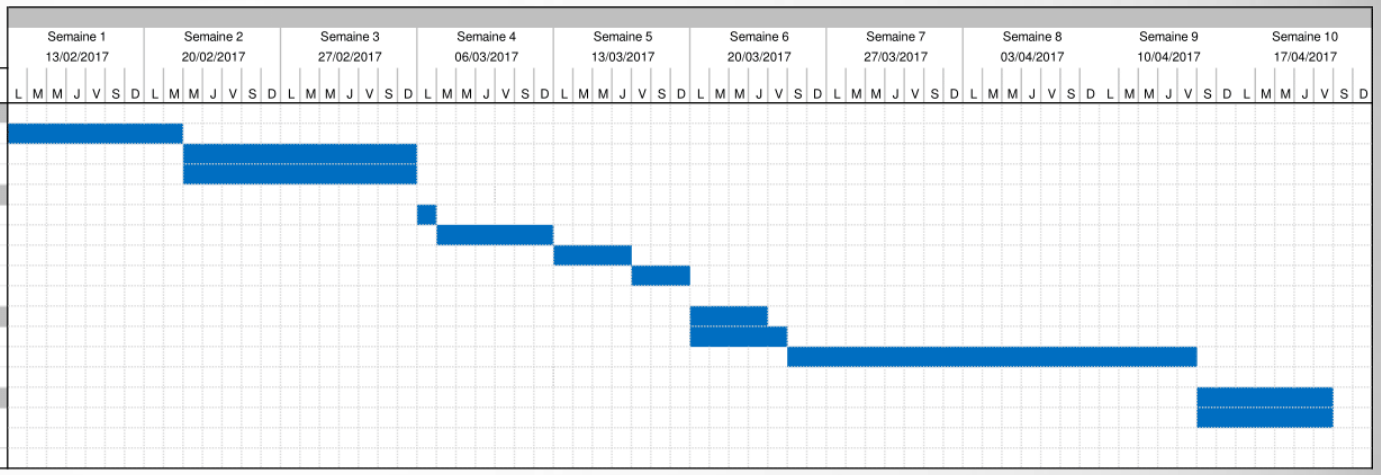
Gantt

[Nom du Projet] Caméra Zombie

AGNELOT Floryann
CARQUILLE Antoine

Date Début Projet: 13/02/2017
Semaines Passées: 1

N°	Tâche	Début	Fin	Jours Prévus	% Accompli	Jours Passés
1	**Approche du Projet**					
1.1	Réflexion sur le sujet	lun 13/02/17	mar 21/02/17	9	100%	9
1.2	Gantt	mer 22/02/17	dim 05/03/17	12	100%	12
1.3	Freemind	mer 22/02/17	dim 05/03/17	12	100%	12
2	**Approche du Matériel**					
2.1	Mise en fonction des 2 Rpi	lun 06/03/17	lun 06/03/17	1	100%	1
2.2	Mise en fonction de la Caméra IP	mar 07/03/17	dim 12/03/17	6	100%	8
2.3	Mise en fonction de la borne Linksys	mar 07/03/17	dim 12/03/17	6	100%	3
2.4	Communication entre les 2 Rpi	lun 13/03/17	jeu 16/03/17	4	100%	2
2.5	Mise en place du serveur	ven 17/03/17	dim 19/03/17	3	100%	3
3	**Réalisation du Projet**					
3.1	Création du site web	lun 20/03/17	jeu 23/03/17	4	100%	4
3.2	Création du script limite de requêtes	lun 20/03/17	ven 24/03/17	5	100%	5
3.3	Création du virus	sam 25/03/17	ven 14/04/17	21	100%	21
4	**Finalisation du Projet**					
4.1	Implémentation virus sur Caméra IP	sam 15/04/17	ven 21/04/17	7	100%	7
4.2	Attaque DoS sur le site	sam 15/04/17	ven 21/04/17	7	100%	7
4.3	Affichage limite de requêtes	sam 15/04/17	ven 21/04/17	7	100%	7



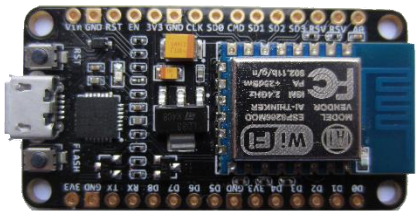
IV. Matériel

1. Choix du Matériel

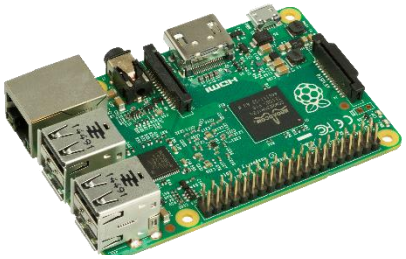
a. Le Microcontrôleur

Pour la réalisation de ce projet, les possibilités étaient diverses et nombreuses. Notamment d'un point de vue matériel. Aujourd'hui, les microcontrôleurs se démocratisent de plus en plus, et continuent de se développer. Ces petits circuits imprimés, davantage compacts et performants possèdent chacun leurs caractéristiques et leurs fonctionnalités. Ce marché est de plus en plus abordable d'un point de vue d'utilisation mais également d'un point de vue budgétaire.
En parallèle, les composants et accessoires combinables avec ces micro-ordinateurs ne sont pas en reste.

Arduino UNO



ESP8266



Raspberry Pi

Parmi tous les microcontrôleurs présents sur le marché, nous avons dû faire un choix alliant efficacité, facilité d'utilisation et à budget raisonnable. Nous avons donc opté pour un Raspberry Pi 1 Model B. Nous avons fait ce choix, car il présente de nombreuses qualités :

- Possibilité d'y installer un OS
- Possibilité de connecter des accessoires USB
- Possibilité de le relier à un réseau filaire (port Ethernet)
- Possibilité de projeter une image sur un écran (port HDMI)
- Possibilité d'ajouter des accessoires CSI

b. La Borne Wifi

Afin de pouvoir recréer un réseau local, nous avons eu besoin d'un routeur Wifi, qui a pour but d'interconnecter les Raspberry Pi ainsi que des ordinateurs. Tout comme les microcontrôleurs, le marché des routeurs Wifi est très vaste, mais une marque se distingue et s'impose comme leader du marché : Linksys. C'est pourquoi nous avons choisi une de leur borne : la Linksys WRT54GL. Cette borne est capable d'interconnecter de façon filaire 4 périphériques finaux, elle est également capable d'interconnecter des périphériques Wifi. La borne est configurable via un site Internet qui est préinstallé dessus.

Borne Linksys WRT54GL

c. Les Accessoires

Pour ce qui est des accessoires, nous avons décidé de choisir une caméra IP compatible avec le Raspberry Pi : le Module Raspberry Pi Camera Rev 1.3.
Afin de pouvoir relier en Wifi le Raspberry Pi à la borne Linksys, il nous a fallu choisir un accessoire permettant ceci. Nous avons choisi une clé Wifi USB TP-Link TL-WN725N qui est plus compacte qu'une antenne Wifi. Afin d'installer le système d'exploitation permettant l'utilisation du Raspberry Pi, nous avons choisi une carte SD classique de 16 GB.

Raspberry Pi Camera Rev

Clé Wifi TP-Link TL-WN725N

Carte SD

2. Mise en Place du Matériel

a. La Caméra IP

Le but du projet étant de prendre le contrôle d'une caméra IP, il nous a fallu simuler une caméra IP avec le Raspberry Pi et le module caméra. Une fois ceci fait et en état de marche, il nous fallait récupérer les images et les vidéos prises par la caméra. Plusieurs possibilités s'offraient à nous :

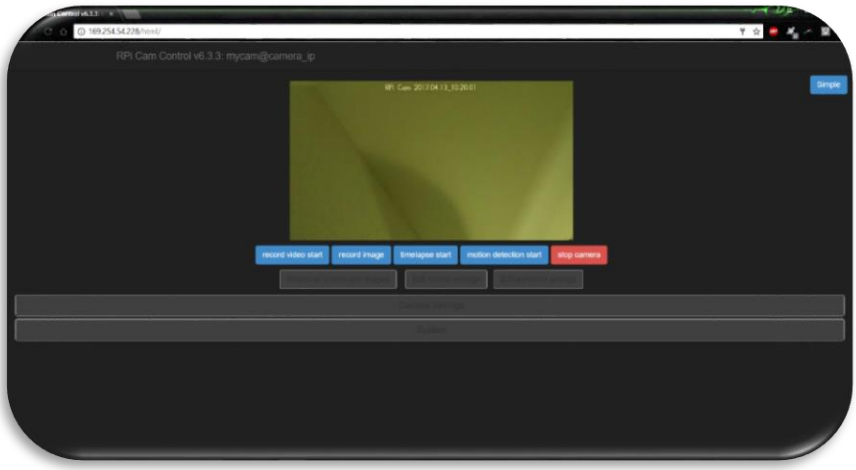
- Récupérer les images grâce à une commande intégrée dans le Raspberry Pi : "raspistill" pour la capture d'images et "raspivid" pour la capture de vidéos. Les données étaient ensuite stockées en mémoire sur le Raspberry Pi. Afin de les récupérer, le seul moyen était d'utiliser le logiciel Filezilla, permettant de téléverser les données du Raspberry Pi sur un ordinateur. Cette méthode n'a pas été retenue, car elle était trop longue et trop fastidieuse.

- Une deuxième solution a été trouvée, installer un package Linux : "motion". Le principe est simple, lorsque la caméra détecte un mouvement, celle-ci lance un « démon » qui lance le package "motion" et enregistre ainsi une vidéo. Cette solution n'a également pas été retenue, car le package "motion" n'était pas compatible avec la caméra IP choisie.

- Une autre méthode, étudiée lors d'un TP du module M1101 Initiation aux Réseaux d'Entreprises ; consistait à installer un package Linux "webcam". Puis après édition des fichiers de configuration, les images vidéo étaient retransmises sur un site web créé préalablement. A nouveau, cette méthode n'a pas été retenue, car le package n'était pas compatible avec le module de caméra IP.

- Par la suite, nous avons retrouvé une nouvelle solution. Pour ce faire, il nous a fallu télécharger une archive ”mjpeg-streamer”. Le principe était le même que celui énoncé précédemment, à la seule différence que le déclenchement de la capture vidéo se faisait manuellement. Cette méthode pour les mêmes raisons que les autres n’a pas été retenue.

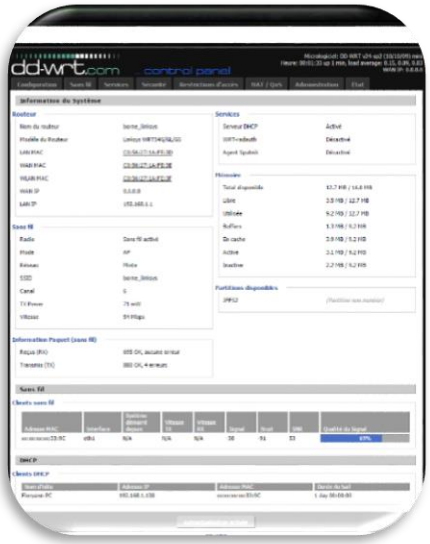
- Finalement, nous avons trouvé à nouveau une solution. Il existe sur la documentation officielle Linux Embarquée, une archive Github : ”Rpi-Cam-Web-Interface”, développée par Silvan Melchior. Celle-ci est en réalité un programme qui installe un serveur web Apache afin de pouvoir visionner le rendu obtenu par le module caméra. Le programme lance un démon qui va allumer automatiquement la caméra IP et retransmettre le rendu sur le site web local. Depuis l’interface web, il est possible de régler divers paramètres tels que la modification de la résolution, la création de timelapse ou encore éteindre ou rebooter la caméra.



Interface Web de la Caméra IP (Rpi-Cam-Web-Interface)

b. La Borne Wifi et le Réseau Local

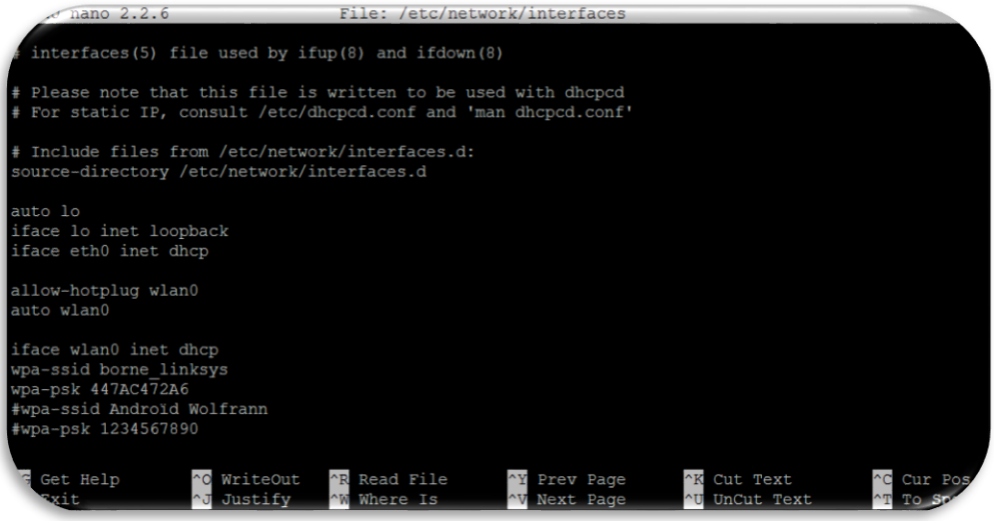
Maintenant que la caméra IP est en place, il nous a fallu reproduire un réseau local, dans lequel apparaîtrait 2 Raspberry Pi : un simulant la caméra IP, et l’autre simulant l’ordinateur de l’administrateur réseau. Pour ce faire, comme expliqué précédemment, nous avons choisi la borne Linksys WRT54GL. Pour pouvoir la configurer, il faut se rendre sur le site web prévu à cet effet ; le site web est en réalité intégré à la borne et accessible depuis un navigateur web à l’adresse ”192.168.1.1”.
Depuis cette interface, nous avons pu renommer la borne ainsi que le nom du réseau wifi ; sécuriser l’accès wifi par un mot de passe et activer des protocoles tels que Telnet ou SSH afin de pouvoir communiquer avec la borne.



Interface de Configuration de la Borne Linksys (DD-WRT Control Panel)

c. Connexion des Raspberry Pi à la Borne Wifi

Enfin, il faut pouvoir relier les Raspberry Pi à la borne Linksys. Pour ce faire, nous avons utilisé un dongle wifi : TP-Link TL-WN725N. Son utilisation est simple, il faut le brancher sur un port USB du Raspberry Pi. Ensuite, il faut modifier un fichier présent dans le Raspberry Pi : /etc/network/interfaces ; et y renseigner le nom et le mot de passe du nouveau réseau et l'adresse IP à utiliser. Une fois cela fait, le dongle Wifi est fonctionnel.

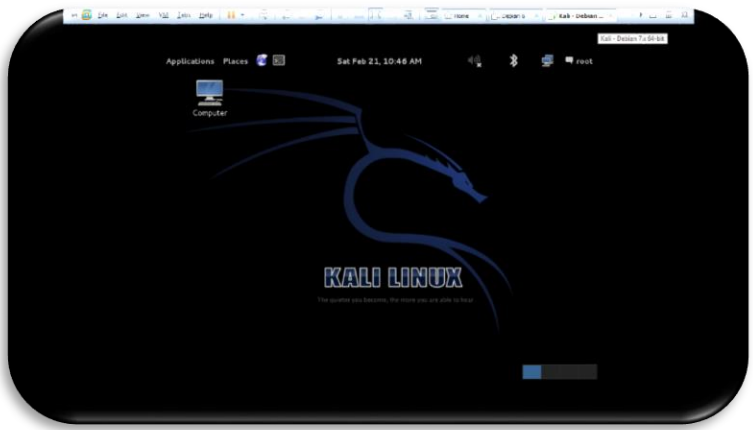


Fichier de Configuration du Dongle Wifi (/etc/network/interfaces)

V. Réalisation du Déni de Service

1. Kali Linux et Ses Outils

Ayant la conviction que Linux était un système d'exploitation beaucoup plus basé sur le hacking que celui de Windows, nous avons, en premier lieu, décidé d'installer sur un ordinateur une machine virtuelle Kali, basée sur Ubuntu. Une fois l'installation de la VM achevée, non à notre grande surprise mais avec joie nous avons pu constater, qu'en effet, cet OS regorge d'outils nécessaires aux tests de sécurité, notamment à ceux d'intrusions.



Bureau Kali Linux

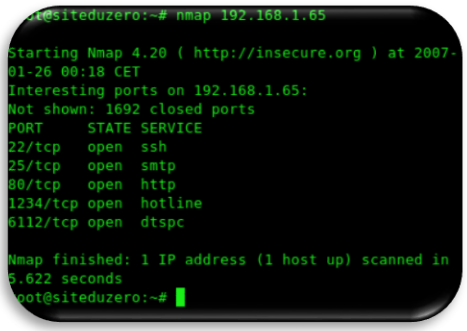
a. Metasploit

Tout d'abord, nous nous sommes intéressés à un outil phare, celui des hackers qui veulent prendre le contrôle d'une machine quelconque à distance : Metasploit Framework. Nous rappelons que le but de notre projet à cet instant était de pouvoir contrôler la caméra IP depuis une autre machine distante. Oui mais voilà : nous nous sommes vite rendu compte que ce logiciel ne pouvait pas être utile dans notre projet. En effet, Metasploit Framework requiert une action de la victime comme l'ouverture d'une pièce jointe dans un mail pour débuter une session sur la machine du hacker. Une fois ceci fait, celui-ci possède un contrôle total sur l'ordinateur distant. Or, ceci n'est pas possible avec la caméra IP puisqu'elle n'effectue aucune action de ce type.

Metasploit

b. Nmap

Après quelques recherches, nous avons découvert un outil qui était de base sur la machine Kali Linux et qui pouvait être assez intéressant. Celui-ci se nomme Nmap. Cet outil permet de prendre connaissance du réseau auquel nous sommes connectés. Nous pouvons faire plusieurs actions, comme mapper tous les hôtes actifs étant dans le même réseau que le nôtre et ainsi récupérer le nom des machines, l'adresse MAC et l'adresse IP de celles-ci. Cette commande Nmap peut prendre un certain nombre d'arguments. Dans notre cas, nous lui indiquons l'interface de sortie et le réseau. Pour un côté plus réaliste, nous avons également indiqué une fausse adresse IP source, c'est-à-dire que l'adresse qui fait des requêtes ARP dans l'ensemble du réseau qui devrait être la nôtre ne l'est pas. Nous découvrons donc le réseau et prenons connaissance de l'adresse IP de notre victime, la caméra IP.



Nmap

Une fois l'adresse IP découverte, nous utilisons une deuxième fois la commande Nmap mais avec des arguments différents. Le but maintenant est d'en savoir plus sur la caméra IP. Nous gardons une adresse IP faussée et indiquons celle de notre victime. Après l'exécution de cette commande, nous obtenons tous les ports ouverts sur celle-ci. Un port nous intéresse tout particulièrement : le port 22, qui correspond à SSH. Désormais, il faut effectuer une attaque de mots de passe sur ce port pour espérer prendre le contrôle de la caméra IP.

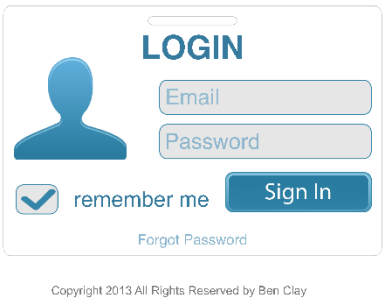
c. Hydra

Après nous être penché brièvement sur Ettercap nous avons opté pour Hydra, qui est également une commande pouvant prendre des arguments en compte et un outil directement installé sur Linux. Hydra permet de cracker un identifiant et un mot de passe grâce à des dictionnaires : un pour le login et un autre pour le mot de passe. Nous voulons obtenir le login et le mot de passe pour effectuer une connexion SSH. Il nous faut donc à ce moment-là créer les dictionnaires.

d. Crunch

Nous avons tenté avec la commande Crunch, qui encore une fois est un outil Linux, de faire ceci. Celle-ci crée un dictionnaire selon les caractères que nous lui donnons, et écrit dans un document texte toutes les combinaisons possibles entre ceux-ci. Cette commande est très intéressante et très puissante, mais dans le cadre de notre projet il n'y avait pas d'utilité à créer un fichier de 15 GO et possédant 1 milliard de lignes pour obtenir un mot de passe, surtout que celui de la caméra IP serait probablement un mot de passe d'usine ou par défaut. Par ailleurs nous avons dû installer une deuxième fois la machine Kali Linux car le document texte de 15 GO a tout simplement rempli tout l'espace mémoire de la VM, rendant le démarrage de celle-ci impossible.

2. Récupération du Mot de Passe



Fenêtre Login

Il nous fallait tout de même deux dictionnaires, c'est pourquoi nous les avons directement créés. Ceux-ci contiennent tous les identifiants et mots de passe simples et par défaut, autrement dit si la caméra IP a « admin » comme login et de même pour le mot de passe, Hydra va réussir à nous transmettre les deux informations recherchées dans un court laps de temps.

3. SSH et Scripts

a. SSH

Actuellement nous avons déjà réussi à avoir l'adresse IP de la caméra, l'état de son port SSH ainsi que les informations pour s'y connecter. Maintenant nous allons envoyer par l'intermédiaire de GFTP, un logiciel de transfert de fichiers, un script Bash et un script Python. Nous reviendrons plus tard sur ceux-ci. En transférant ces deux scripts, GFTP va en quelque sorte préparer la caméra IP pour à terme faire un déni de service sur le serveur qui est nous le rappelons le but du projet.

Enfin, il nous fallait un moyen de se connecter à la caméra IP via SSH et d'exécuter le script Bash cité un peu plus haut automatiquement. Pour permettre cela nous avons utilisé la commande sshpass qui se connecte en SSH sur la caméra après avoir indiqué comme arguments l'adresse IP, le nom d'utilisateur et le mot de passe de celle-ci.

Logo SSH

SSH

Les huit derniers paragraphes expliquent d'une manière compréhensive, nous l'espérons, la façon dont le payload principal de notre projet fonctionne. Toutes les commandes s'exécutent sans intervention du hacker, mise à part le transfert par GFTP. Nous allons maintenant vous expliquer ce que fait le script Python, et dans un dernier temps le script Bash lancé par le payload principal.

b. Script Python

Ce script Python va créer un paquet SYN. En d'autres termes les 44 lignes qui le compose sont la configuration même du paquet. Celui-ci est paramétré pour utiliser le protocole TCP et envoyer des requêtes sur le port 80 (HTTP) de la victime. L'adresse IP source et le port source ne sont jamais les mêmes d'un paquet à un autre.
Nous ajoutons que quand une requête SYN est reçue sur le port 80 du serveur, celui-ci stocke les informations du paquet mais n'en renvoie aucun à l'adresse source, qui bien sûr n'existe pas, et par conséquent le serveur ne fait pas la connexion en trois étapes du protocole TCP. Il garde tout de même des informations en mémoire. Quand un paquet est envoyé sur le réseau, une ligne nous confirmant un envoi réussi s'affiche dans le terminal du hacker.

Logo Python

Nous allons faire une attaque SYN flood : elle consiste à submerger le serveur en envoyant des paquets SYN très rapidement et en grand nombre. A l'issu de cette attaque, elle provoquera le déni de service du serveur ciblé. A cet instant, nous avons créé un paquet SYN mais il faudrait l'envoyer en boucle afin de DOS notre victime.

c. Script Bash

C'est là que le script Bash intervient : il va tout simplement dire à la caméra IP d'exécuter le script Python plusieurs fois et dans le même temps. Celle-ci va donc envoyer des paquets SYN au serveur afin de faire un déni de service sur celui-ci. Nous avons donc transformé la caméra IP en caméra zombie.

Logo Bash

Pour résumer, le payload principal lancé depuis la machine Linux va prendre connaissance des informations nécessaires et relatives au contrôle à distance de la caméra IP, lui transférer le script Bash ainsi que le script Python et dire à celle-ci d'exécuter le script Bash tout juste envoyé. La caméra IP va donc l'exécuter, et envoyer des paquets de synchronisation (SYN) en boucle sur le port 80 du serveur. Celui-ci ne pourra pas gérer le nombre incalculable de requêtes et sa mémoire va se retrouver pleine. Au bout d'un certain temps, le site web du serveur devient inaccessible : nous avons provoqué le déni de service.

VI. Problèmes Rencontrés

Avant de faire une conclusion de ce qui a été évoqué auparavant, nous tenons à vous partager quelques problèmes que nous avons rencontrés au cours du projet. Pour la plupart, ceux-ci nous ont ralentis mais je pense que d'un côté ils nous ont fait comprendre pourquoi cela ne fonctionnait pas et ont donc une fin instructive.

Tout d'abord, le codage du paquet SYN. Ce script est certes trouvable sur Internet, néanmoins plusieurs versions existent et pas une d'entre elles ne fonctionnait directement. Il a donc fallu que nous modifiions ce script, nous trouvons ça plutôt intéressant car désormais nous savons comment est codé ce paquet en question. Celui-ci que nous croyons compliquer, ne comporte en fait, qu'une quarantaine de lignes ! (Et compréhensibles). Nous avons donc configuré le script pour qu'il effectue ce que nous recherchions.

Autre problème plus précis et à l'intérieur même du script Python : l'adresse IP. Nous vous avions dit plus haut que chaque paquet dispose d'une adresse IP source différente d'un autre. En effet, dans la configuration de celui-ci la librairie « Random » est utilisée pour que l'adresse soit choisie aléatoirement. Oui mais nous nous sommes rendu compte qu'elle crée des adresses IP publiques : bien évidemment il fallait rester dans le domaine privé puisque nous sommes dans un réseau local. Problème réglé.

Nous avons parlé de librairie : justement celle citée au-dessus n'a pas été la seule à nous faire des siennes. La librairie Scapy, indispensable pour créer un paquet que nous voulons envoyer sur le réseau, nous a ralenti sur la fin de notre projet. En effet, celle-ci ne se situait pas à un emplacement valide dans l'arborescence Linux, ce qui donnait comme résultat que le script Python ne s'exécutait pas et aucun paquet n'était envoyé. Une fois ce problème réglé, nous n'avions plus qu'à tester la totalité de notre travail pour voir si nous avions réussi.

Pour nous le projet qui nous avait été donné, celui qui provoquait un déni de service depuis une caméra zombie, a été réalisée avec succès. Donc après des heures de recherches et de travail nous pouvons dire que notre projet fonctionne.

Nous avons tout de même une chose à rajouter : nous avons remarqué que le déni de service ne pouvait pas être possible depuis le Raspberry Pi. En toute honnêteté nous pensons que celui-ci n'est pas assez puissant pour exécuter le script Bash et envoyer les paquets rapidement. Même s'il en envoie, ce qui prouve que notre travail fonctionne réellement, le Raspberry Pi les transmet beaucoup trop lentement à la borne.

Le déni de service a donc été expérimenté depuis la machine Linux et ce fut un succès : le site web est devenu inaccessible. Bien évidemment, nous avons utilisé les mêmes scripts et les mêmes outils.

VII. Remerciements

Nous souhaitons remercier les personnes qui ont contribué à la réussite de ce projet.

Tout d'abord nos tuteurs : Mme Christine Bolou-Chiaravalli et M. Stéphane Givron grâce au suivi et à leurs conseils lorsque nous rencontrions des problèmes.

Nous souhaitons également remercier un collègue, Rémy Boucard, étudiant MMI à l'I.U.T. de Dijon ; pour son travail quant à son aide à la réalisation de l'affiche de notre projet.

Mais également nos camarades de classes qui ont pu nous proposer des solutions lorsque nous rencontrions des problèmes.

Comme évoqué dans nos remerciements, un poster a été réalisé, voici un lien permettant de le visionner : https://poster-camera-zombi.000webhostapp.com/index.html/

De même, une vidéo a été réalisée, voici le lien permettant de la visionner : https://youtu.be/rc5COhhDg7c

VIII. Sources

- Image Arduino Uno : By R.hampl - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=40167098
- Image RaspberryPi : By Evan-Amos - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=49898562
- Image Module ESP8266 : 由Vowstar - 自己的作品， CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=39348325
- Image Caméra IP : CC BY 2.0, https://www.sparkfun.com/products/retired/11868
- Image Clé Wifi : Copyright © 2017 TP-Link Technologies Co., Ltd. Tous droits réservés.
- Image Carte SD : By Rektal Trauma - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45620257
- Image Borne Linksys WRT54GL : By I, Jonathan Zander, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4964895
- Image Bureau Kali Linux : Par Arshane88 — Travail personnel, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=38517115
- Image Metasploit : By https://blackarch.org - https://blackarch.org, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=55100523
- Image Code Nmap : By Taonas - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1600669
- Image Login : By Benjamin F Clay - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=27808571
- Image Logo SSH : By Unknown - http://www.ssh.com/documents/30/SSHTectiaServer60_zOS_UserManual.pdf, Public Domain, https://commons.wikimedia.org/w/index.php?curid=40433249
- Image Dossier SSH : CC BY-SA 2.5, https://commons.wikimedia.org/w/index.php?curid=3373991
- Image Logo Python : By Benjamin Hell (en:User:Siebengang) - Python logo from http://www.python.org/community/logos/, modifications own work., Public Domain, https://commons.wikimedia.org/w/index.php?curid=34991652
- Image Logo Bash : By Justindorfman - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=53428398

Réalisé par :

Antoine CARQUILLE & Floryann AGNELOT