A vendre.

Analyse de fichier audio

Énoncé

L'équipe d'investigation numérique a récupéré ce fichier audio lors de recherches sur le disque dur externe d'un suspect et soupçonne que son émetteur ait caché une information à l'intérieur. Celle-ci pourrait être très importante pour la suite de l'enquête, à vous de la retrouver.
PS : La police « Code 128 Normal » peut vous être utile.

Réalisation

- Création de l'image sous Paint.

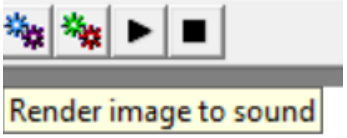
Génération du mot de passe choisi sous forme de code barre, puis sauvegarder l'image en JPG.

Bien joué ! Le mot de passe est : yxEeRn2q

Mot de passe en clair

Image du mot de passe sous forme de code barre

- Génération d'un son par rapport à l'image avec Coagula.



Option à utiliser

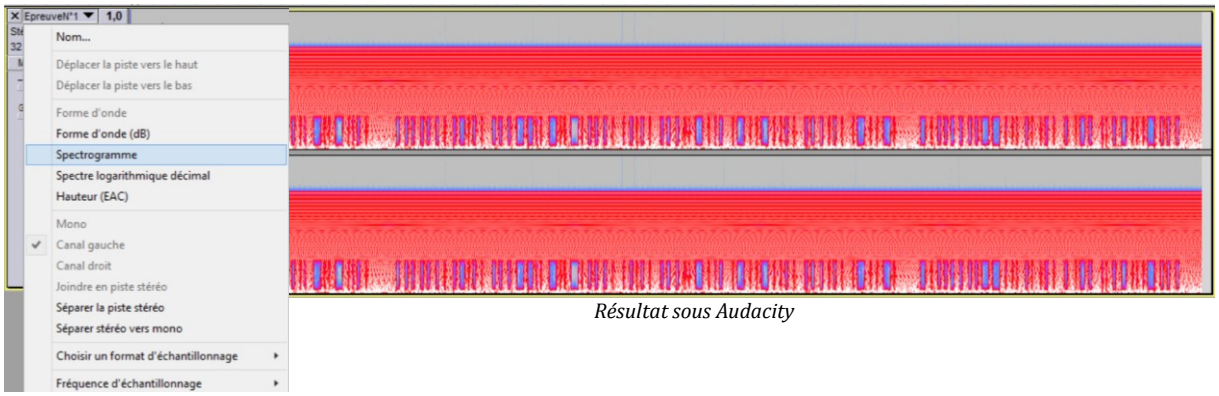
Génération du son sur l'image sous Coagula

- Enregistrer ce dernier : *EpreuveN°1.wav*

Solution

Pour résoudre cette épreuve, il suffit d'ouvrir le document .wav avec Audacity ou encore Sonic Visualiser. A ce stade le spectre parait normal, le mot de passe n'apparait pas au premier coup d'œil.

Avec Audacity, pour faire apparaitre l'image contenu dans le fichier son et ainsi le mot de passe, il suffit de sélectionner l'option « Spectrogramme ».



Résultat sous Audacity

A l'aveugle.

Analyse d'une image.

Énoncé

L'équipe d'investigation numérique a récupéré ce fichier image lors de la saisie d'un échange entre deux suspects et soupçonne que son émetteur ait caché une information à transmettre au récepteur de la plus haute importance à l'intérieur. Celle-ci pourrait être très utile pour la poursuite de l'enquête, à vous de la retrouver.

Réalisation

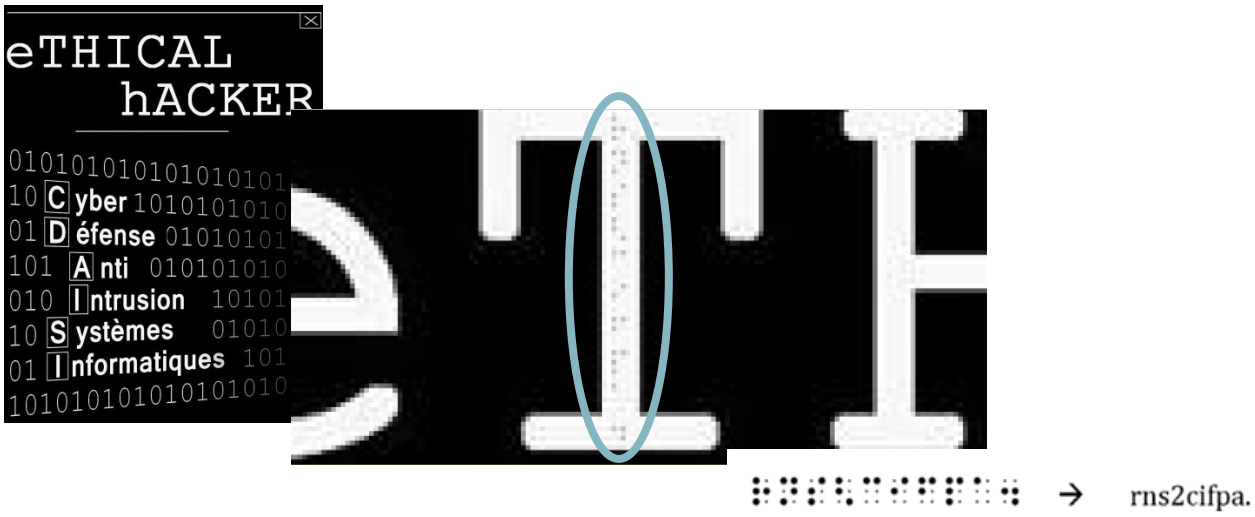
La réalisation de cette épreuve est simple, il suffit d'ouvrir l'image choisie avec le logiciel The Gimp (gratuit, Linux/Windows). Ensuite, il faut agrandir l'image autant que souhaité et se positionner à l'endroit voulu. A ce stade il ne reste plus qu'à choisir la police « braille » et d'écrire le mot de passe choisi. Enfin, il est nécessaire de jouer avec la taille et la couleur des caractères pour camoufler le mot de passe.

De plus, pour masquer davantage le mot de passe nous pouvons ajouter un peu de transparence dans l'image. Pour cela, il faut ajouter un canal alpha qui correspond à la transparence, à l'endroit voulu (au niveau du mot de passe).

Solution

Pour la résolution de l'épreuve, si le canal alpa n'est pas mis en place il suffit d'ouvrir l'image avec n'importe quelle visionneuse d'image et de l'agrandir afin de retrouver la partie dans laquelle le mot de passe est inscrit. Puis ensuite, il est juste nécessaire de traduire les caractères braille par les caractères alphanumériques correspondants.

Si le canal alpha est mis en place il faut utiliser des logiciels qui ne gèrent pas la transparence pour pouvoir visualiser le mot de passe. Sous Windows nous pouvons utiliser Paint, et en équivalent sous Linux nous avons KolourPaint. Ensuite, il suffit de traduire les caractères brailles avec les caractères alphanumériques correspondants.



Allo ?!

Analyse d'une capture réseau.

Énoncé

Cette conversation a été enregistrée lors d'une discussion entre deux personnes surveillées par les services de police, celle-ci contient des informations primordiales pour la poursuite de l'enquête : la date, l'heure et le lieu de leur prochaine attaque (Exemple de saisie : Lille20/01/10h).

Réalisation

Pour réaliser cette épreuve, il a été nécessaire de mettre en place préalablement un serveur de téléphonie IP (VoIP), ici nous avons choisi de mettre en place un serveur Asterisk sur une machine Linux Debian. Sur ce serveur nous avons également implémenté le protocole SRTP (Secure Real-Time Protocol), c'est-à-dire un protocole qui va permettre de chiffrer la voix.

Ensuite nous avons mis en place deux clients qui hébergeront des softphones, nous avons alors choisi des clients Windows 8 sur lesquels nous avons installé Blink ou Linphone puisqu'ils prennent en charge le protocole SRTP.

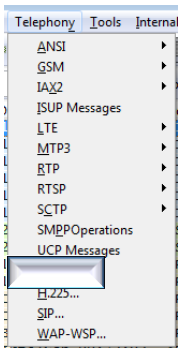
Enfin, nous avons effectué une conversation entre deux individus dans laquelle on retrouvera le mot de passe recherché. Cette conversation sera alors capturée et enregistrée grâce au logiciel Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.17	212.83.184.57	UDP	46	Source port: sip Destination port: sip
2	0.000174000	IntelCor_29:e5:9b	Broadcast	ARP	42	Who has 192.168.1.13? Tell 192.168.1.17
3	0.258455000	192.168.1.17	89.2.0.1	DNS	72	Standard query 0xa8ad NAPTR sip2sip.info
4	0.261313000	192.168.1.17	212.83.184.57	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=0a15) [Reassembled in #5]
5	0.261323000	192.168.1.17	212.83.184.57	SIP/XML	833	Request: PUBLISH sip:6001@212.83.184.57 |
6	0.261955000	192.168.1.17	212.83.184.57	SIP	560	Request: REGISTER sip:212.83.184.57 |
7	0.262805000	192.168.1.17	212.83.184.57	SIP	707	Request: SUBSCRIBE sip:6001@212.83.184.57 |
8	0.290979000	212.83.184.57	192.168.1.17	SIP	624	Status: 401 Unauthorized (0 bindings) |
9	0.291215000	192.168.1.17	212.83.184.57	SIP	720	Request: REGISTER sip:212.83.184.57 |
10	0.292258000	212.83.184.57	192.168.1.17	SIP	544	Status: 489 Bad Event |
11	0.293733000	212.83.184.57	192.168.1.17	SIP	622	Status: 401 Unauthorized |
12	0.293947000	192.168.1.17	212.83.184.57	SIP	872	Request: SUBSCRIBE sip:6001@212.83.184.57 |
13	0.295314000	89.2.0.1	192.168.1.17	DNS	227	Standard query response 0xa8ad NAPTR 30 100 s NAPTR 10 100 s NAPTR 50 100 s
14	0.298155000	192.168.1.17	89.2.0.1	DNS	82	Standard query 0x7ecd SRV _sip._udp.sip2sip.info
15	0.305855000	212.83.184.57	192.168.1.17	SIP	710	Request: NOTIFY sip:25109746@192.168.1.17:51273 |
16	0.305979000	192.168.1.17	212.83.184.57	SIP	574	Status: 200 OK |
17	0.308782000	212.83.184.57	192.168.1.17	SIP	645	Status: 200 OK (1 bindings) |
18	0.309904000	212.83.184.57	192.168.1.17	SIP	602	Status: 200 OK |
19	0.310253000	212.83.184.57	192.168.1.17	SIP	710	Request: NOTIFY sip:86597132@192.168.1.17:51310 |
20	0.310785000	192.168.1.17	212.83.184.57	SIP	574	Status: 200 OK |
21	0.319499000	89.2.0.1	192.168.1.17	DNS	119	Standard query response 0x7ecd SRV 100 100 5060 proxy.sipthor.net
22	0.966364000	IntelCor_29:e5:9b	Netgear_cc:a4:24	ARP	42	Who has 192.168.1.1? Tell 192.168.1.17
23	0.966481000	IntelCor_29:e5:9b	Broadcast	ARP	42	Who has 192.168.1.13? Tell 192.168.1.17
24	0.968052000	Netgear_cc:a4:24	IntelCor_29:e5:9b	ARP	42	192.168.1.1 is at 00:1f:33:cc:a4:24
25	1.964158000	IntelCor_29:e5:9b	Broadcast	ARP	42	Who has 192.168.1.13? Tell 192.168.1.17

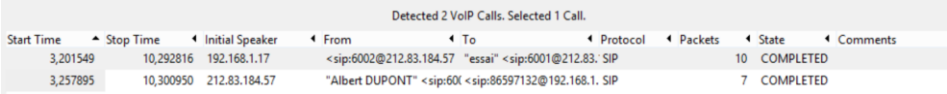
Capture Wireshark

Solution

Pour résoudre cette épreuve, il suffit de reconstituer la conversation. Pour cela, dans un premier temps il est nécessaire d'ouvrir la capture de la conversation avec le logiciel Wireshark.
Ensuite, Wireshark propose une option spécifique pour la VoIP, celle –ci va nous permettre d'écouter la conversation, il faut alors sélectionner « Telephony → VoIP Calls ».



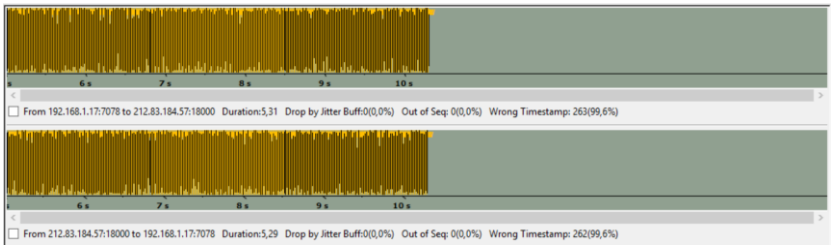
Une fois lancé, celle-ci va retrouver dans la capture les différentes conversations (s'il y en a plusieurs).



Detected 2 VoIP Calls. Selected 1 Call.

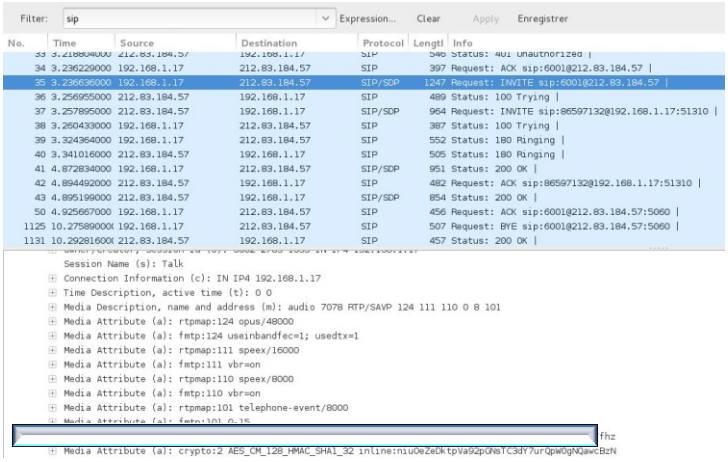
Start Time	Stop Time	Initial Speaker	From	To	Protocol	Packets	State	Comments
3,201549	10,292816	192.168.1.17	<sip:6002@212.83.184.57	"essai" <sip:6001@212.83.	SIP	10	COMPLETED	
3,257895	10,300950	212.83.184.57	"Albert DUPONT" <sip:60(	<sip:86597132@192.168.1.	SIP	7	COMPLETED	

Il suffit alors de sélectionner la conversation qui nous intéresse, et de lancer « Player », puis « Decode » et on sélectionne les cases « From ».



Cependant, on constate que la lecture fait ressortir des grésillements, plutôt que les paroles normalement enregistrées : il va donc falloir déchiffrer la voix.

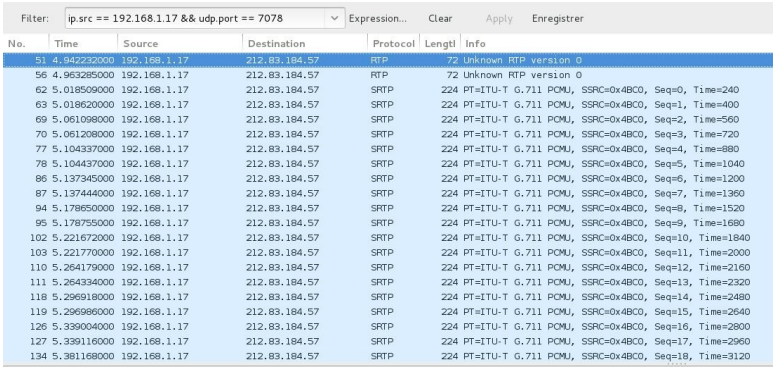
Ainsi, pour déchiffrer la voix il faut dans un premier temps trouver la clé avec laquelle la capture a été chiffrée. Pour cela, on applique un filtre dans Wireshark pour ne récupérer que les informations du protocole SIP.



Une fois le filtre appliqué on récupère la clé qui va nous permettre de déchiffrer la conversation : dJyCdeYx2Cd5oqCqZsrkw+Cdl+tGn4xRFFF9Yfhz.

Maintenant nous allons utiliser l'outil srtp-decrypt afin de générer un fichier en hexadécimal qui nous servira pour déchiffrer le protocole SRTP.

Pour commencer on applique un filtre pour isoler le protocole SRTP, on observe la présence d'un paquet RTP, celui-ci correspond à un paquet envoyé par le softphone Linphone par défaut.



Filter: ip.src == 192.168.1.17 && udp.port == 7078 — Expression... Clear Apply Enregistrer

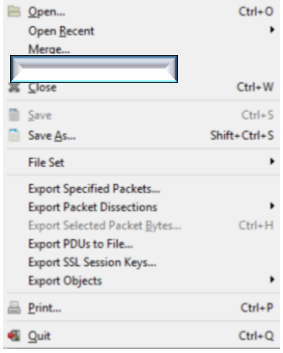
No.	Time	Source	Destination	Protocol	Length	Info
51	4.942232000	192.168.1.17	212.83.184.57	RTP	72	Unknown RTP version 0
56	4.963285000	192.168.1.17	212.83.184.57	RTP	72	Unknown RTP version 0
62	5.018509000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=0, Time=240
63	5.018620000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=1, Time=400
69	5.061098000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=2, Time=560
70	5.061208000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=3, Time=720
77	5.104337000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=4, Time=880
78	5.104437000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=5, Time=1040
86	5.137345000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=6, Time=1200
87	5.137444000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=7, Time=1360
94	5.178650000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=8, Time=1520
95	5.178755000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=9, Time=1680
102	5.221672000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=10, Time=1840
103	5.221770000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=11, Time=2000
110	5.264179000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=12, Time=2160
111	5.264334000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=13, Time=2320
118	5.296918000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=14, Time=2480
119	5.296986000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=15, Time=2640
126	5.339004000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=16, Time=2800
127	5.339116000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=17, Time=2960
134	5.381168000	192.168.1.17	212.83.184.57	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x4BC0, Seq=18, Time=3120

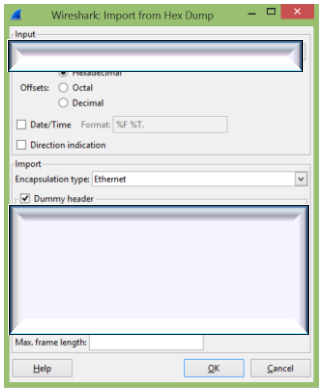
Une fois le filtre appliqué on enregistre les paquets dans une nouvelle capture (file → export specified packets … → .pcap), celle –ci correspondra à la conversation venant d'un des deux émetteurs.

Ensuite, pour générer le fichier hexadécimal nous allons utiliser l'outil srtp-decrypt qui va prendre en paramètre la clé de chiffrement, la seconde capture en entrée et un fichier texte en sortie.

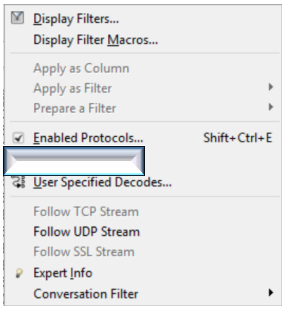
```
root@kali:~/Desktop# cd srtp-decrypt-master/
root@kali:~/Desktop/srtp-decrypt-master# ./srtp-decrypt -k dJyCdeYx2Cd5oqCqZsrkw
+Cdl+tGn4xRFFF9Yfhz < /root/Desktop/rtpOKDecrypt.pcap > /root/Desktop/rtpOK.txt
```

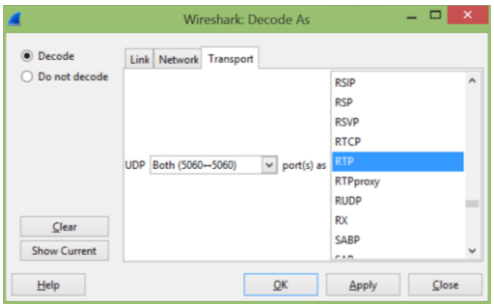
La commande précédente a donc généré un fichier dans lequel on retrouve des informations en hexadécimale qui correspondent à des données de déchiffrement. Ce fichier sera alors ré-implémenté dans Wireshark pour déchiffrer les paquets en SRTP.



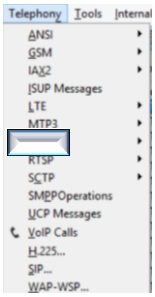


Une fois le fichier implémenté dans Wireshark, les paquets SRTP deviennent des paquets UDP, pour qu'ils soient reconnus en RTP il faut le spécifier.

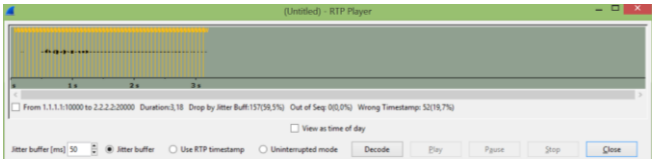




Enfin, il ne reste plus qu'à écouter la conversation qui est maintenant audible puisqu'elle n'est plus chiffrée.
Pour cela, les étapes sont à peu près identiques aux précédentes, sauf qu'il faut choisir l'option RTP dans le menu Telephony.



Ainsi, après avoir sélectionné la conversation puis l'avoir analysé, nous pouvons lancer le player. On obtient le spectrogramme de la voix et nous pouvons alors jouer le son. On retrouve donc Dijon12/05/14h.



Identifie-Moi.

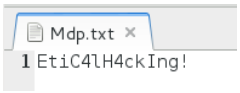
Analyse d'une capture réseau.

Énoncé

Lors d'une enquête d'investigation cet échange a été saisi, celui-ci contient le mot de passe pour déchiffrer les disques durs saisis précédemment au domicile du suspect. Pouvez-vous nous aider à le retrouver ?

Réalisation

Pour réaliser cette épreuve, il suffit d'utiliser une machine Linux et la commande Hping3. Avec celle-ci, il faut avant tout créer un fichier texte contenant le mot de passe choisi.



Fichier contenant le mot de passe.

Puis, à l'aide de la commande hping il faut générer le nombre de paquet voulu dans lesquels sera placé le mot de passe divisé en plusieurs parties. Enfin, il suffit d'enregistrer l'échange grâce à Wireshark.

```
root@kali:~/Desktop# hping3 --icmp --count 4 --data 4 --file Mdp.txt 192.168.1.1
4
HPING 192.168.1.14 (eth0 192.168.1.14): icmp mode set, 28 headers + 4 data bytes
[main] memlockall(): Success
Warning: can't disable memory paging!
len=46 ip=192.168.1.14 ttl=128 id=3370 icmp_seq=0 rtt=1.4 ms
len=46 ip=192.168.1.14 ttl=128 id=3371 icmp_seq=1 rtt=0.8 ms
len=46 ip=192.168.1.14 ttl=128 id=3372 icmp_seq=2 rtt=0.9 ms
len=46 ip=192.168.1.14 ttl=128 id=3373 icmp_seq=3 rtt=1.5 ms

--- 192.168.1.14 hping statistic ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.8/1.2/1.5 ms
```

Commande et échange ICMP.

De façon à rendre cette épreuve plus compliqué nous avons réalisé une seconde version toujours basée sur l'échange ICMP mais avec le mot de passe placé dans un autre champ de la trame.
Pour réaliser cette épreuve, il suffit d'utiliser une machine Linux et le Framework Python ainsi que l'interpréteur spécialisé réseau Scapy.

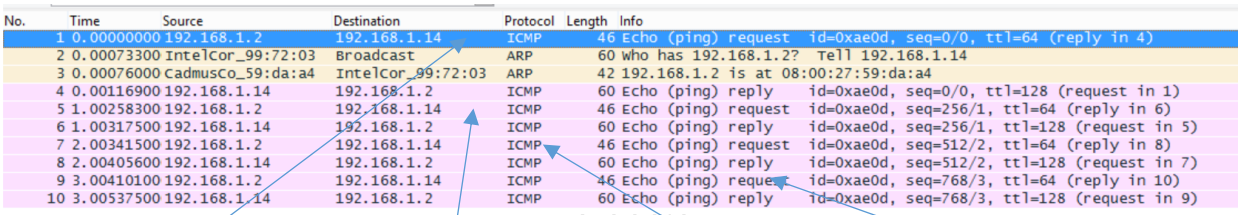
A l'aide de Python et Scapy on réalise un script qui va envoyer des ping sur le réseau à une adresse donnée (ici www.google.fr). Dans celui-ci on entre une chaine de caractère qui correspondra au mot de passe à trouver, cette chaine sera alors décomposée caractère par caractère qui seront transformés en hexadécimal et introduit au niveau du champ ID.

```
#! /usr/bin/python

#Importe toutes les bibliotheques scpay dans python
from scapy.all import *
conf.verb=0 #n'affiche pas d'info de debogage a l'ecran

chaine="EtiC4lH4ckIng!"
p=IP(dst="www.google.fr")/ICMP()
for c in chaine:
        p[IP].id=ord(c)
        send(p)
```

Script Python/Scapy

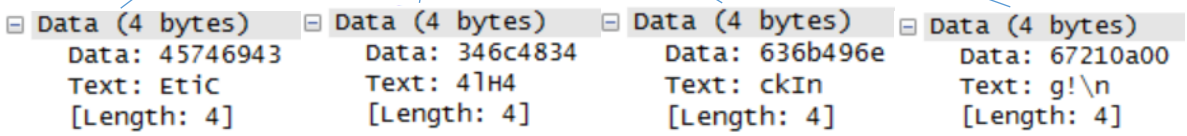
Solution

Pour la résolution de cette épreuve (version 1), il suffit d'ouvrir le document fournit avec le logiciel Wireshark. Ensuite, il nous faut l'étudier, paquet par paquet. On remarque ainsi plusieurs paquets ICMP, ceux-ci contiennent une série de caractères.



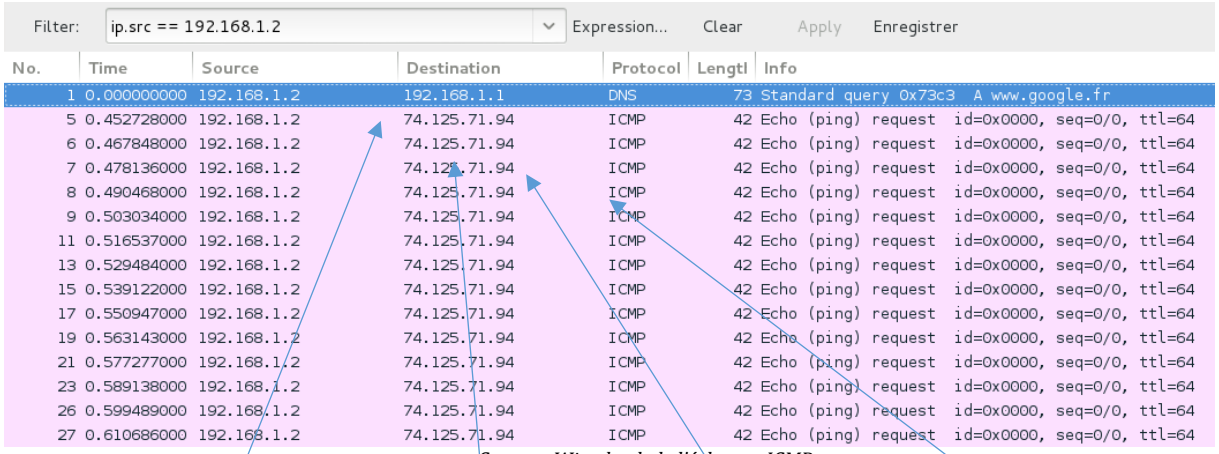
No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	192.168.1.2	192.168.1.14	ICMP	46	Echo (ping) request id=0xae0d, seq=0/0, ttl=64 (reply in 4)
2	0.00073300	IntelCor_99:72:03	Broadcast	ARP	60	Who has 192.168.1.2? Tell 192.168.1.14
3	0.00076000	CadmusCo_59:da:a4	IntelCor_99:72:03	ARP	42	192.168.1.2 is at 08:00:27:59:da:a4
4	0.00116900	192.168.1.14	192.168.1.2	ICMP	60	Echo (ping) reply id=0xae0d, seq=0/0, ttl=128 (request in 1)
5	1.00258300	192.168.1.2	192.168.1.14	ICMP	46	Echo (ping) request id=0xae0d, seq=256/1, ttl=64 (reply in 6)
6	1.00317500	192.168.1.14	192.168.1.2	ICMP	60	Echo (ping) reply id=0xae0d, seq=256/1, ttl=128 (request in 5)
7	2.00341500	192.168.1.2	192.168.1.14	ICMP	46	Echo (ping) request id=0xae0d, seq=512/2, ttl=64 (reply in 8)
8	2.00405600	192.168.1.14	192.168.1.2	ICMP	60	Echo (ping) reply id=0xae0d, seq=512/2, ttl=128 (request in 7)
9	3.00410100	192.168.1.2	192.168.1.14	ICMP	46	Echo (ping) request id=0xae0d, seq=768/3, ttl=64 (reply in 10)
10	3.00537500	192.168.1.14	192.168.1.2	ICMP	60	Echo (ping) reply id=0xae0d, seq=768/3, ttl=128 (request in 9)

Capture Wireshark de l'échange ICMP.



Reconstitution des données de l'échange.

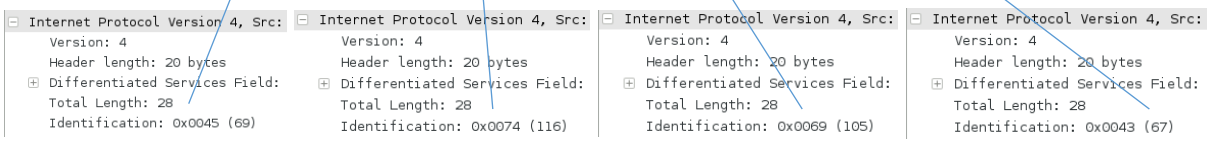
Pour la résolution de cette épreuve (version 2), il suffit d'ouvrir le document fournit avec le logiciel Wireshark. Ensuite, il nous faut l'étudier, paquet par paquet. On remarque ainsi plusieurs paquets ICMP, ceux-ci ne contiennent aucune donnée. Il faut s'intéresser à un champ particulier du paquet ICMP, le champ ID. Dans celui-ci on retrouve chacun de nos caractère en hexadécimal. Il faut donc récupérer toutes les valeurs et les traduire en ASCII pour retrouver la chaine recherchée.



Filter: ip.src == 192.168.1.2 Expression... Clear Apply Enregistrer

No.	Time	Source	Destination	Protocol	Lengtl	Info
1	0.000000000	192.168.1.2	192.168.1.1	DNS	73	Standard query 0x73c3 A www.google.fr
5	0.452728000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
6	0.467848000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
7	0.478136000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
8	0.490468000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
9	0.503034000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
11	0.516537000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
13	0.529484000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
15	0.539122000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
17	0.550947000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
19	0.563143000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
21	0.577277000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
23	0.589138000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
26	0.599489000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64
27	0.610686000	192.168.1.2	74.125.71.94	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64

Capture Wireshark de l'échange ICMP.



Nous avons choisi le mot de passe « EtiC4lH4ckIng ! ».

L'eXORciste.

Le XOR dans tous ses états

Énoncé

Lors d'un échange de données entre deux individus surveillés par les forces de l'ordre, cette image a été interceptée, elle pourrait contenir des informations utiles à l'enquête. Cependant, celle-ci est impossible à ouvrir, elle parait corrompue. Saurez-vous nous venir en aide ?

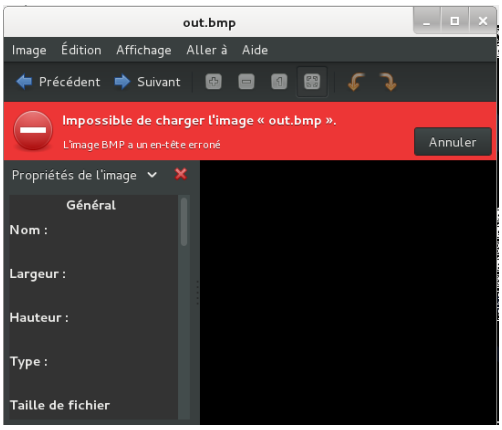
Réalisation

Pour réaliser cette épreuve, il est nécessaire d'utiliser une machine Linux (ou Windows) implémenté du langage Python.

Ensuite, il est nécessaire de réaliser une image simple sous Paint qui contiendra le mot de passe qui validera l'épreuve.

L'image va alors ensuite être chiffrée via l'utilisation de XOR, celle-ci ne sera alors plus affichable par les utilisateurs.
Pour le chiffrement avec XOR il est nécessaire de connaitre son fonctionnement, comme dans tout système cryptographique on retrouve une formule mathématique, à savoir : a XOR b = (not(a) and b) ^ (a and not(b)). Ainsi pour le chiffrement de l'image nous avons réalisé un script en Python qui reprend la formule exposée précédemment.

Image non chiffrée Image chiffrée



(impossible à ouvrir)

```
from PIL import Image
key = 'FIC2015'
cypher = bytearray(open("FIC2015.bmp", "rb").read())
for i in range(len(cypher)):
        cypher[i] ^= ord(key[i%len(key)])
        print cypher
        open("out.bmp","wb").write(cypher)
try:
        im = Image.open("out.bmp")
        im.verify()
except IOError:
        print 'rate'
else:
        print 'verification ok'
```

Script de chiffrement

Solution

Pour résoudre cette épreuve, les utilisateurs devrons réaliser un script qui permettra de déchiffrer et ainsi de faire apparaitre l'image et par la même occasion le mot de passe recherché.

```
from PIL import Image
key = 'FIC2015'
cypher = bytearray(open("FIC2015.bmp", "rb").read())
for i in range(len(cypher)):
        cypher[i] ^= ord(key[i%len(key)])
        print cypher
        open("out.bmp","wb").write(cypher)
try:
        im = Image.open("out.bmp")
        im.verify()
except IOError:
        print 'rate'
else:
        print 'verification ok'
```

Script de résolution (identique au chiffrement)

Image en cous de déchiffrement