

## **TP 3 Analyse du Malware**



**Salima Zribi & Nourhen Messaadi & Aliya Maatit**

## Introduction

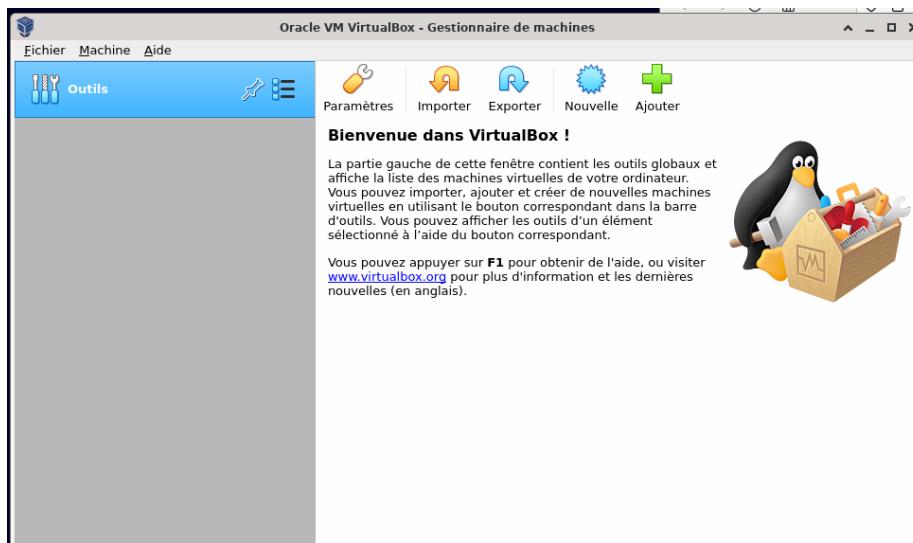
Dans ce TP, notre objectif était de mettre en place un environnement contrôlé permettant d'analyser le comportement de logiciels malveillants sans danger pour notre machine hôte. Pour cela, nous avons utilisé deux machines virtuelles : une Kali Linux pour l'analyse et une Windows 7 comme machine victime. Nous avons également configuré INetSim afin de simuler un faux Internet, ce qui nous a permis d'intercepter toute tentative de communication du malware.

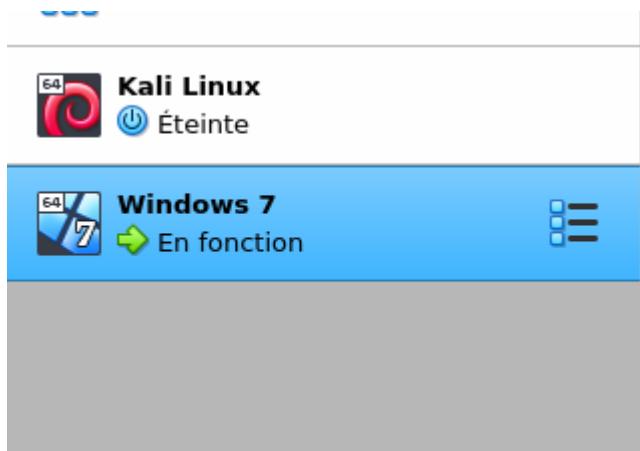
Après avoir mis en place le réseau, le partage de fichiers et la configuration DNS, nous avons téléchargé un échantillon de malwares et nous les avons exécutés sur la machine Windows tout en surveillant leur activité réseau avec Wireshark. L'objectif était d'observer leurs comportements et d'identifier les techniques utilisées pour communiquer, analyser l'environnement ou tenter d'établir une connexion avec un serveur externe.

## 1. Configuration de l'Environnement d'Analyse

### 1.1. Installation des Machines Virtuelles

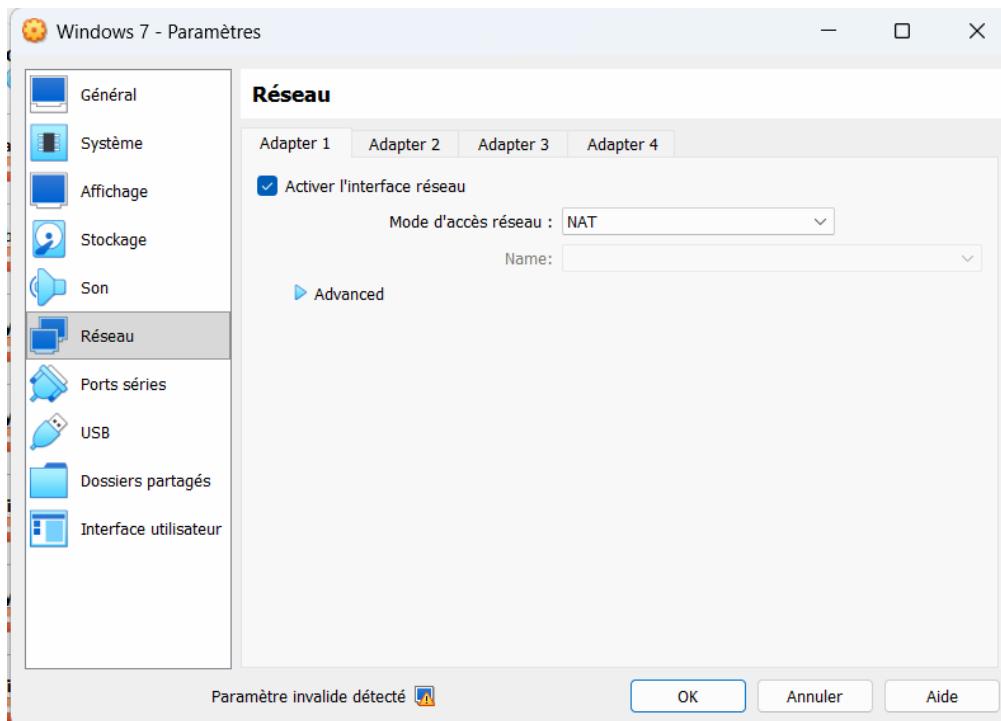
- Oracle VM VirtualBox utilisée comme hyperviseur
- Kali Linux : machine d'analyse
- Windows 7 : machine victime

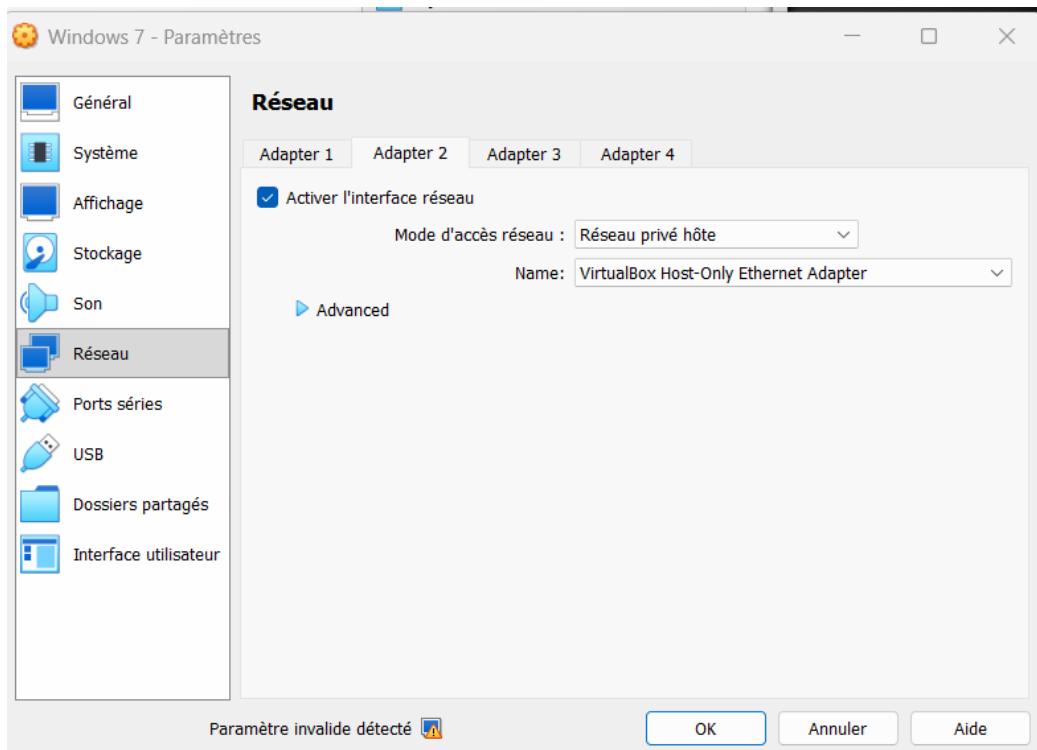




## 1.2. Configuration Réseau

- Mode NAT configuré sur les deux machines
- Adaptateur 2 en mode "Réseau privé hôte"
- Interface : VirtualBox Host-Only Ethernet Adapter





## Configuration IP Statique

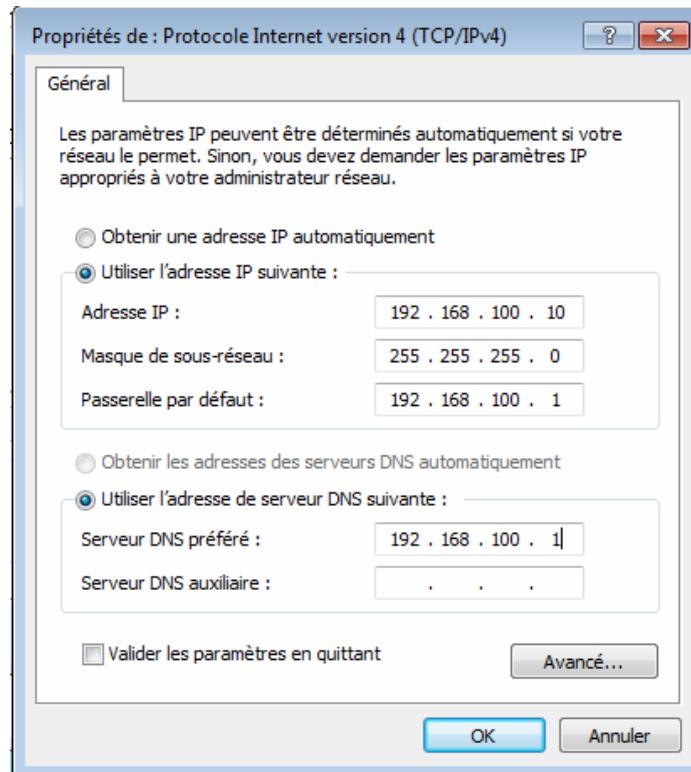
Kali Linux :

```
salima@kali: ~
Session Actions Éditer Vue Aide
    valid_lft forever preferred_lft forever
(salima@kali)-[~]
$ ip addr add 192.168.100.1/24 dev eth1
RTNETLINK answers: Operation not permitted
(salima@kali)-[~]
$ sudo ip addr add 192.168.100.1/24 dev eth1
[sudo] Mot de passe de salima :
(salima@kali)-[~]
$ sudo ip link set eth1 up
(salima@kali)-[~]
$ ip addr show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:4c:84:74 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.1/24 brd 192.168.100.255 scope global dynamic noprefixroute eth1
        valid_lft 443sec preferred_lft 443sec
        inet 192.168.100.1/24 scope global eth1
            valid_lft forever preferred_lft forever
        inet6 fe80::a00:27ff:fe4c:8474/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
(salima@kali)-[~]
$
```

→ Adresse configurée : 192.168.100.1/24

Windows 7 :

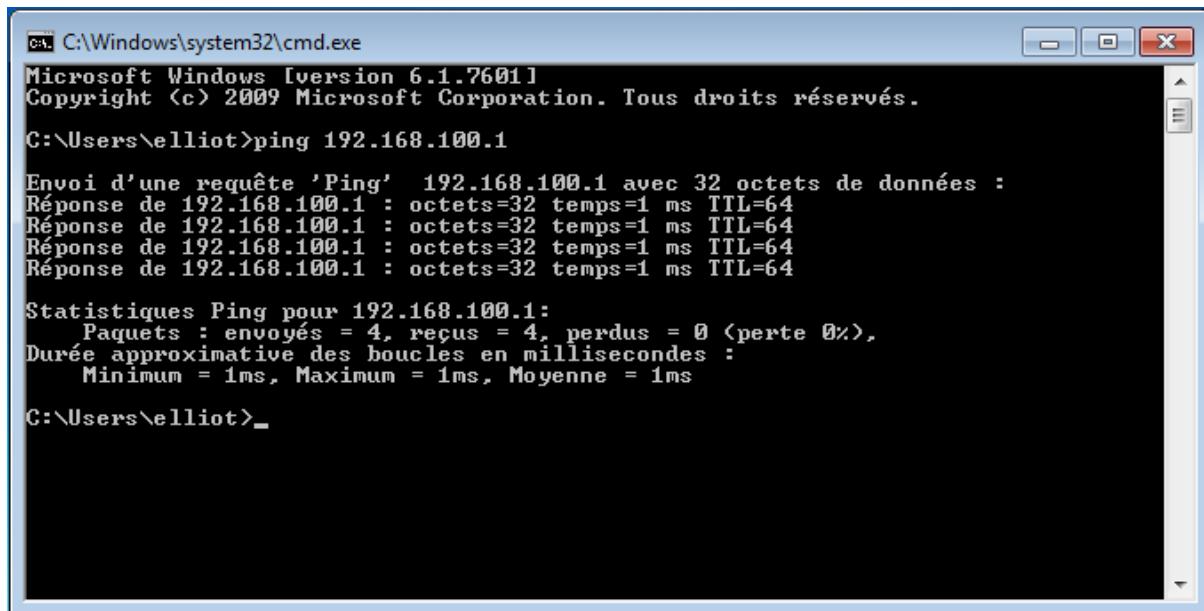
- IP : 192.168.100.10
- Masque : 255.255.255.0
- Passerelle : 192.168.100.1
- DNS : 192.168.100.1



## Tests de Connectivité

Ping depuis Windows vers Kali

Les tests de connectivité montrent que la communication entre les deux machines fonctionne correctement. Depuis Windows, les paquets ICMP envoyés vers Kali (192.168.100.1) sont tous reçus avec un temps moyen d'environ 1 ms, ce qui confirme que Windows atteint bien la machine d'analyse.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [version 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. Tous droits réservés.

C:\Users\elliot>ping 192.168.100.1

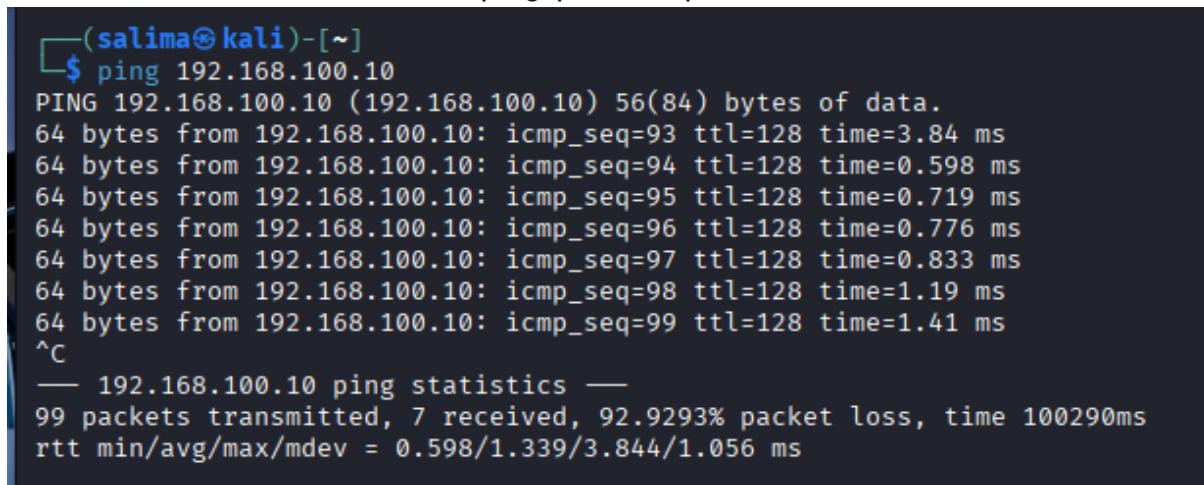
Envoi d'une requête 'Ping' à 192.168.100.1 avec 32 octets de données :
Réponse de 192.168.100.1 : octets=32 temps=1 ms TTL=64

Statistiques Ping pour 192.168.100.1:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 1ms, Maximum = 1ms, Moyenne = 1ms

C:\Users\elliot>_
```

Ping depuis Kali vers Windows :

Dans l'autre sens, Kali arrive aussi à joindre Windows (192.168.100.10). Les réponses ICMP sont bien reçues, avec des temps très faibles. Le taux de perte affiché est dû à l'arrêt manuel du ping, pas à un problème réseau réel.



```
[salima@kali)-[~]
$ ping 192.168.100.10
PING 192.168.100.10 (192.168.100.10) 56(84) bytes of data.
64 bytes from 192.168.100.10: icmp_seq=93 ttl=128 time=3.84 ms
64 bytes from 192.168.100.10: icmp_seq=94 ttl=128 time=0.598 ms
64 bytes from 192.168.100.10: icmp_seq=95 ttl=128 time=0.719 ms
64 bytes from 192.168.100.10: icmp_seq=96 ttl=128 time=0.776 ms
64 bytes from 192.168.100.10: icmp_seq=97 ttl=128 time=0.833 ms
64 bytes from 192.168.100.10: icmp_seq=98 ttl=128 time=1.19 ms
64 bytes from 192.168.100.10: icmp_seq=99 ttl=128 time=1.41 ms
^C
--- 192.168.100.10 ping statistics ---
99 packets transmitted, 7 received, 92.9293% packet loss, time 100290ms
rtt min/avg/max/mdev = 0.598/1.339/3.844/1.056 ms
```

les deux machines sont correctement configurées dans le réseau Host-Only, et la connectivité bidirectionnelle est opérationnelle, ce qui est essentiel pour analyser le trafic du malware via INetSim.

## Déploiement des Outils d'Analyse

installation d'inetsim

```
(salima@kali)-[~]
$ sudo apt update
Réception de : 1 http://kali.download/kali kali-rolling InRelease [34,0 kB]
Réception de : 2 http://kali.download/kali kali-rolling/main amd64 Packages [20,9 MB]
Réception de : 3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [52,6 MB]
Réception de : 4 http://kali.download/kali kali-rolling/contrib amd64 Packages [114 kB]
Réception de : 5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [259 kB]
Réception de : 6 http://kali.download/kali kali-rolling/non-free amd64 Packages [187 kB]
Réception de : 7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [894 kB]
Réception de : 8 http://kali.download/kali kali-rolling/non-free-firmware amd64 Packages [11,7 kB]
Réception de : 9 http://kali.download/kali kali-rolling/non-free-firmware amd64 Contents (deb) [28,5 kB]
75,1 Mo réceptionnés en 16s (4 691 ko/s)
1178 paquets peuvent être mis à jour. Exécutez « apt list --upgradable » pour les voir.
```

```
(salima@kali)-[~]
$ sudo apt install inetsim
inetsim est déjà la version la plus récente (1.3.2+dfsg.1-1).
inetsim passé en « installé manuellement ».
Sommaire :
  Mise à niveau de : 0. Installation de : 0Supprimé : 0. Non mis à jour : 117
8
```

## configuration du dns sur inetsim

```
GNU nano 8.6                               /etc/inetsim/inetsim.conf *
dns-service = on
dns_ip =192.168.100.1
```

Dans cette configuration, nous avons activé le service DNS d'INetSim (dns-service = on) et nous lui avons attribué l'adresse IP 192.168.100.1, qui correspond à la machine Kali.

Cela permet de forcer la machine Windows à envoyer toutes ses requêtes DNS vers INetSim, au lieu d'aller vers un vrai serveur Internet.

Dès que le malware tente de résoudre un nom de domaine, c'est INetSim qui répond. Cela nous permet de capturer et observer entièrement son comportement réseau en toute sécurité.

```
#####
service_bind_address=192.168.56.10
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
#service_bind_address 10.10.10.1
```

Ici, on a configuré INetSim pour qu'il écoute sur l'adresse 192.168.56.10, qui correspond à l'IP de notre Kali sur le réseau Host-Only. En faisant ça, on s'assure que tous les services simulés (DNS, HTTP, etc.) sont disponibles pour la machine Windows.

```
#####
dns_bind_port 53
#
# Port number to bind DNS service to
#
# Syntax: dns_bind_port <port number>
#
### Default: 53
```

Dans cette partie, on a laissé le port DNS par défaut : le port 53. Comme c'est le port standard utilisé pour la résolution DNS, Windows considère INetSim comme un vrai serveur DNS.

```
#####
# dns_default_ip
#
# Default IP address to return with DNS rep
#
# Syntax: dns_default_ip <IP address>
#
# Default: 127.0.0.1
#
dns_default_ip 192.168.56.10

#####
^G Aide      ^O Écrire      ^F Chercher
^X Quitter   ^R Lire fich.  ^V Remplacer
```

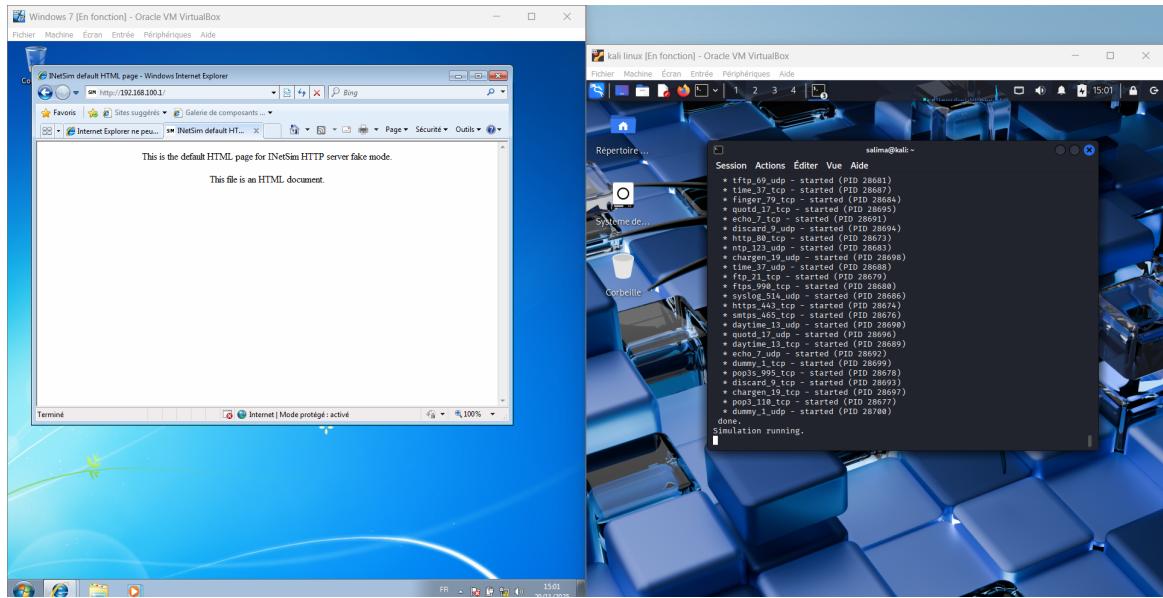
Avec ce paramètre, on a défini l'IP que renverra INetSim pour toutes les requêtes DNS. En mettant 192.168.56.10, ça veut dire que n'importe quel domaine sera redirigé vers Kali. Ça nous permet de garder le malware complètement enfermé dans notre réseau et de surveiller toutes ses tentatives de communication.

```
# Default: none
#
dns_static www.liglab.fr 192.168.56.10
#dns_static ns1.foo.com 10.70.50.30
#dns_static ftp.bar.net 10.10.20.30

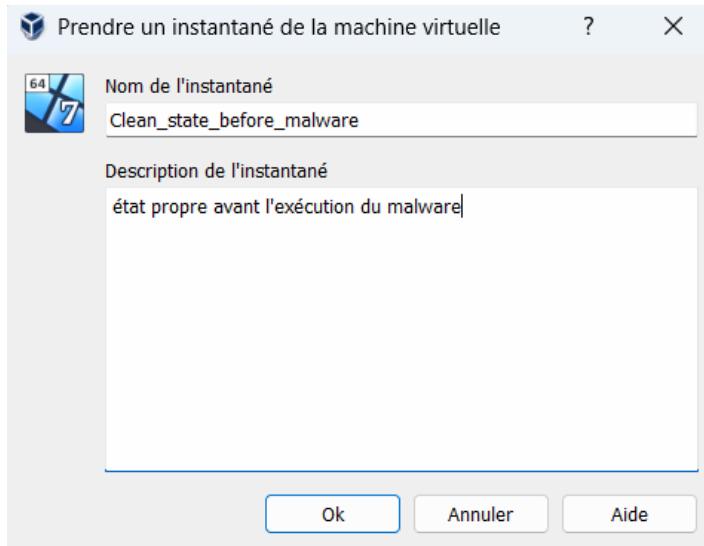
#####
# dns_version
#
# DNS version
#
# Syntax: dns_version <version>
```

Ici, on a ajouté une entrée dns\_static pour que [www.liglab.fr](http://www.liglab.fr) pointe directement vers l'IP de Kali (192.168.56.10). Ça nous permet de vérifier qu'INetSim répond bien aux requêtes DNS et que Windows redirige correctement le trafic vers notre faux serveur.

Fonctionnement d'inetsim sur windows en tapant l'adresse ip de la machine hôte dans le navigateur windows



On a créé un instantané de la machine Windows avant de lancer le malware.  
Ça nous permet de garder un état propre, pour pouvoir revenir en arrière facilement si l'échantillon casse le système ou laisse des traces.  
C'est notre point de sauvegarde pour refaire les tests sans tout reconfigurer à chaque fois.



### création du dossier partagé entre les deux vms

Ici, on a installé Samba sur Kali pour pouvoir créer un dossier partagé accessible depuis Windows. C'est grâce à ça qu'on peut transférer les malwares depuis Kali vers la machine victime sans passer par Internet. L'installation se fait sans erreur, donc la partie partage est bien prête.

```
Session Actions Éditer Vue Aide
└─(salima㉿kali)-[~]
$ sudo apt update
[sudo] Mot de passe de salima :
Atteint : 1 http://http.kali.org/kali kali-rolling InRelease
Tous les paquets sont à jour.

└─(salima㉿kali)-[~]
$ sudo apt install samba -y
samba est déjà la version la plus récente (2:4.23.3+dfsg-1).
samba passé en « installé manuellement ».
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
  amass-common          libudfread0
  gir1.2-girepository-2.0  libwireshark18
  libarmadillo14         libwiretap15
  libbluray2             libwsutil16
  libbson-1.0-0t64        libx264-164
  libdisplay-info2        libyelp0
  libgdal37               python3-bluepy
  libgeos3.14.0           python3-click-plugins
  libgirepository-1.0-1   python3-gpg
  libinstpatch-1.0-2      python3-kismetcapturebtgeiger
  libjs-jquery-ui          python3-kismetcapturefreaklabszigbee
  libjs-underscore         python3-kismetcapturertl433
  libmongoc-1.0-0t64      python3-kismetcapturertladsb
  libnet1                  python3-kismetcapturetlamr
  libobjc-14-dev           python3-protobuf
  libplacebo349            python3-xlutils
```

Dans cette capture, on vérifie que le service Samba (smbd) est bien en cours d'exécution. Le statut indique active (running), ce qui veut dire que le partage réseau fonctionne correctement et que Windows pourra accéder aux fichiers qu'on mettra dans le dossier partagé.

```
└─(salima㉿kali)-[~]
$ sudo systemctl status smbd
● smbd.service - Samba SMB Daemon
  Loaded: loaded (/usr/lib/systemd/system/smbd.service; disabled; preset:>)
  Active: active (running) since Thu 2025-11-20 21:44:44 CET; 2min 46s ago
  Invocation: 6d6a2d90e4604ce0ac9cc9dd57585823
    Docs: man:smbd(8)
          man:samba(7)
          man:smb.conf(5)
  Main PID: 5918 (smbd)
    Status: "smbd: ready to serve connections ... "
     Tasks: 3 (limit: 2116)
    Memory: 23.6M (peak: 23.8M)
      CPU: 262ms
     CGroup: /system.slice/smbd.service
             └─5918 /usr/sbin/smbd --foreground --no-process-group
                 ├─5922 "smbd: notifyd" .
                 ├─5923 "smbd: cleanupd"
                 └─5924 "smbd: main" .
```

nov. 20 21:44:43 kali systemd[1]: Starting smbd.service - Samba SMB Daemon ...
nov. 20 21:44:44 kali update-apparmor-samba-profile[5912]: grep: /etc/apparm...
nov. 20 21:44:44 kali update-apparmor-samba-profile[5915]: diff: /etc/apparm...
nov. 20 21:44:44 kali systemd[1]: Started smbd.service - Samba SMB Daemon.

Ici, on a créé un petit fichier test\_safe.txt dans le dossier malware\_quarantine pour vérifier que notre partage Samba fonctionne bien. On l'a généré directement depuis Kali, puis on a listé le contenu du dossier pour confirmer qu'il est bien enregistré.

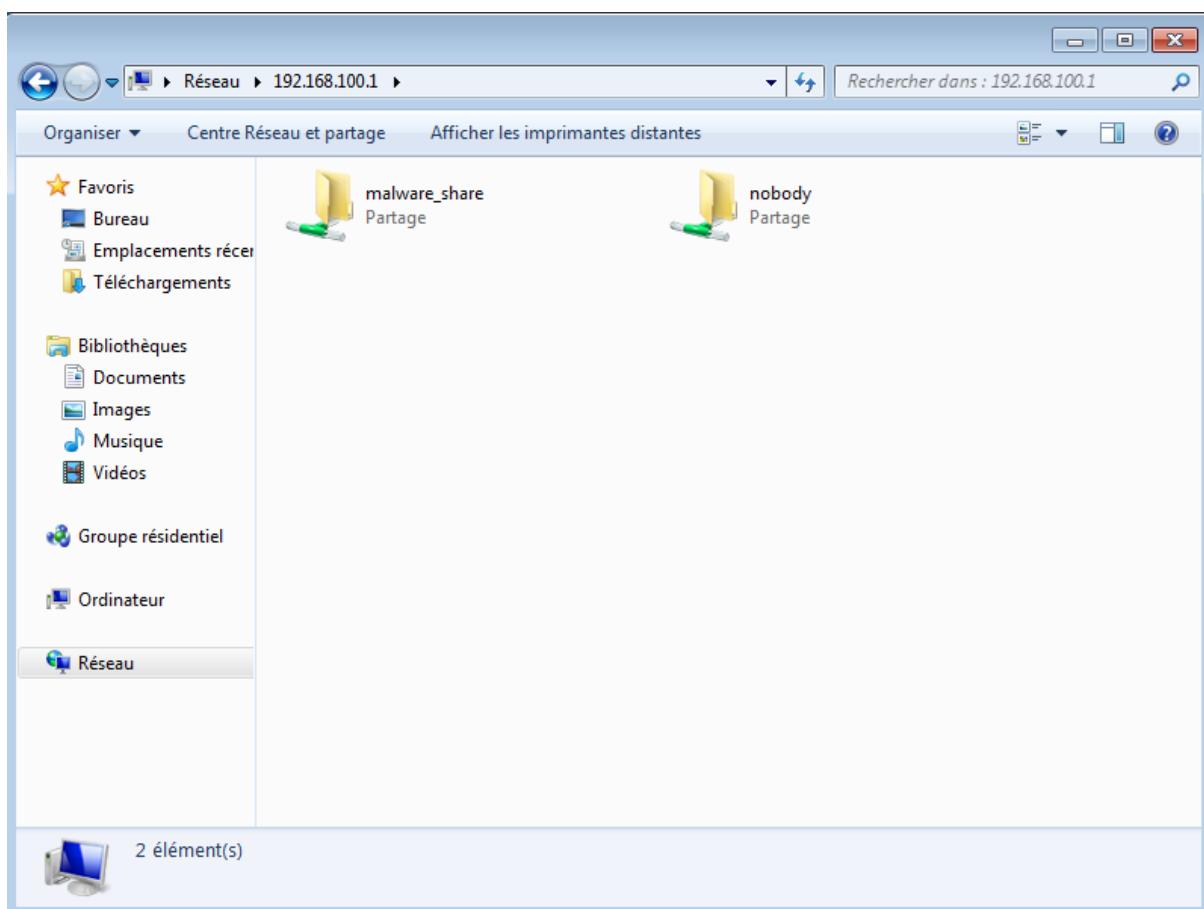
L'idée : avant de transférer les vrais malwares, on s'assure que Windows pourra voir les fichiers sans problème

```
(salima㉿kali)-[~]
└─$ echo "Ceci est un test sécurisé" >/home/salima/malware_quarantaine/test_safe.txt

(salima㉿kali)-[~]
└─$ ls -la /home/salima/malware_quarantaine/
total 12
drwx—— 2 salima salima 4096 20 nov. 21:49 .
drwx—— 18 salima salima 4096 20 nov. 21:38 ..
-rw-rw-r-- 1 salima salima 28 20 nov. 21:49 test_safe.txt
```

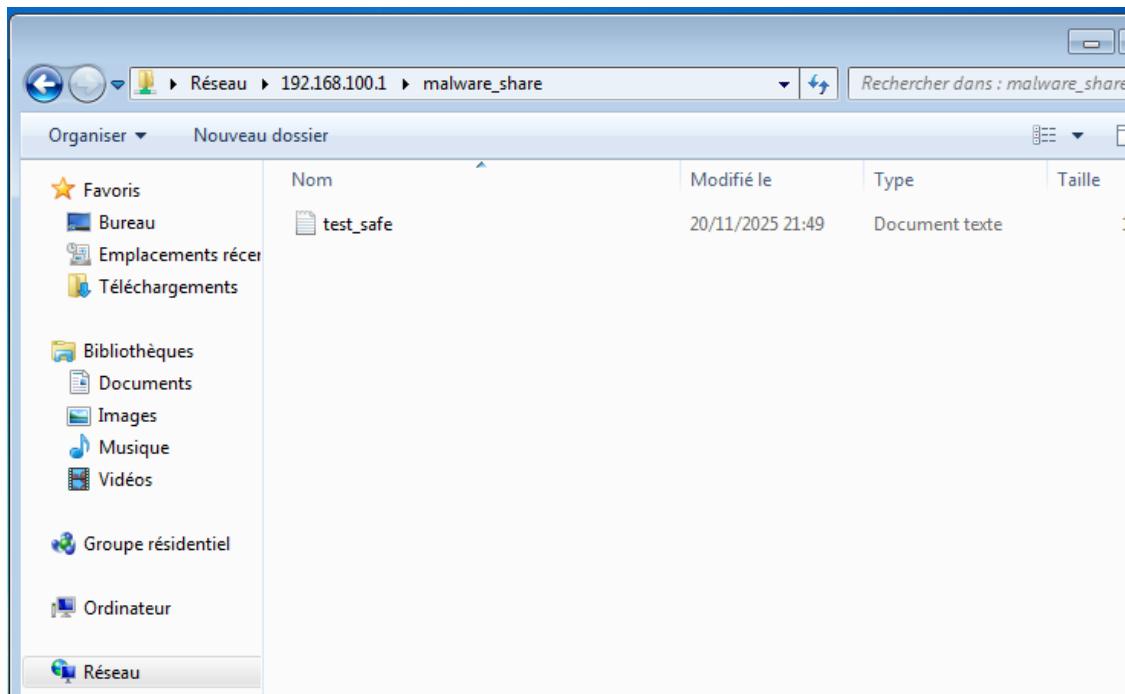
Dans cette étape, on vérifie que Windows arrive bien à voir le dossier partagé hébergé sur Kali. En allant sur l'adresse \\192.168.100.1, on voit apparaître les deux partages Samba, dont malware\_share, celui qu'on a créé pour transférer nos échantillons.

Conclusion : la connexion au partage fonctionne, donc Windows pourra récupérer les fichiers malwares qu'on déposera dans ce dossier.



on voit que Windows accède correctement au dossier partagé malware\_share et qu'il affiche bien le fichier test\_safe qu'on avait créé depuis Kali.

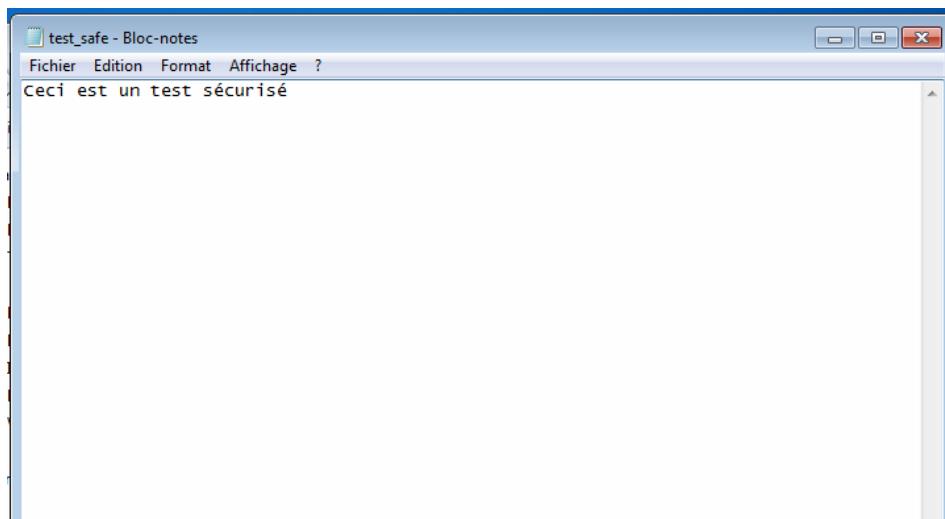
Ça confirme que le partage Samba fonctionne dans les deux sens : Kali peut déposer des fichiers, et Windows peut les lire sans problème.



On a ouvert le fichier `test_safe.txt` directement depuis Windows pour vérifier qu'il a bien été transmis depuis Kali.

Comme on le voit, le contenu affiché ("Ceci est un test sécurisé") correspond exactement à ce qu'on avait créé sur Kali.

Le partage Samba fonctionne parfaitement et Windows peut lire sans problème les fichiers que nous déposerons — on est donc prêts à transférer les malwares.



## Téléchargement du malware sur Kali

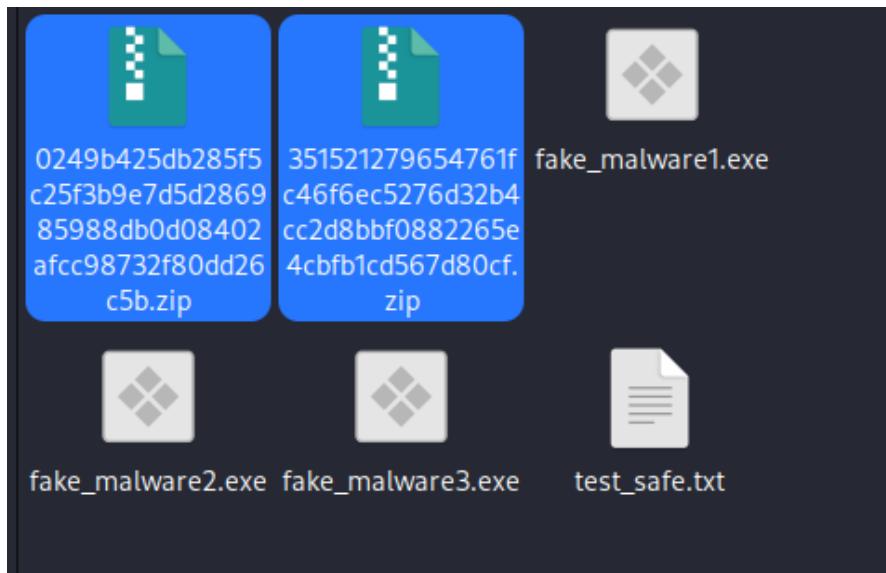
On est allés sur le site MalwareBazaar pour récupérer nos échantillons de malware. Comme on voit sur la capture, la plateforme nous demande d'abord de vérifier que nous ne sommes pas un robot avant d'autoriser le téléchargement.

The screenshot shows the MalwareBazaar homepage. At the top, there is a navigation bar with links for 'Browse', 'Upload', 'Hunting Alerts', 'Access Data', 'FAQ', 'About', and 'Login'. Below the navigation bar, there is a search bar and a logo for 'from ABUSE.ch | SPAMHAUS'. A prominent 'Open application menu' button is located in the top right corner. The main content area displays a CAPTCHA challenge: 'Please confirm that you are not a robot by clicking on the checkbox below'. It includes a checkbox labeled 'I am human', the 'hCaptcha' logo, and links for 'Privacy' and 'Terms'. At the bottom of the page, there is a footer bar with social media icons for X, LinkedIn, and others, and links for 'Terms and Conditions', 'Terms of Use', 'Privacy Policy', and 'Cookie Policy'.

Sur cette capture, on voit la liste des échantillons disponibles sur MalwareBazaar : chaque malware est affiché avec son hash, son type (.exe / .elf), sa signature et ses tags (ex : RAT, trojan, stealer...). À partir de cette liste, on a choisi plusieurs fichiers à télécharger pour les analyser plus tard dans la machine Windows.

Date (UTC)	SHA256 hash	Type	Signature	Tags	Reporter	DL
2025-11-20 22:11	<a href="#">bb82bbd8e3c463aa3ab...</a>	<input type="checkbox"/> exe		<a href="#">exe</a>	smica83	<a href="#">Download</a>
2025-11-20 22:10	<a href="#">4154a81334fa6da11138...</a>	<input type="checkbox"/> exe	<a href="#">PureLogsStealer</a>	<a href="#">exe</a> <a href="#">PureLogsStealer</a>	abuse_ch	<a href="#">Download</a>
2025-11-20 21:42	<a href="#">8d62a94ba04ccb38cf0...</a>	<input checked="" type="checkbox"/> elf	<a href="#">Mirai</a>	<a href="#">elf</a> <a href="#">mirai</a> <a href="#">upx-dec</a>	abuse_ch	<a href="#">Download</a>
2025-11-20 21:41	<a href="#">0249b425db285f5c25f3...</a>	<input type="checkbox"/> exe		<a href="#">exe</a> <a href="#">Ransomware</a> <a href="#">trojan</a> <a href="#">Updater</a>	Anonymous	<a href="#">Download</a>
2025-11-20 21:41	<a href="#">2daf43fb7e635c338f3c3...</a>	<input checked="" type="checkbox"/> elf	<a href="#">Mirai</a>	<a href="#">elf</a> <a href="#">mirai</a> <a href="#">UPX</a>	abuse_ch	<a href="#">Download</a>
2025-11-20 21:40	<a href="#">351521279654761fc46f...</a>	<input type="checkbox"/> exe	<a href="#">njrat</a>	<a href="#">exe</a> <a href="#">NJRAT</a> <a href="#">RAT</a>	abuse_ch	<a href="#">Download</a>
2025-11-20 21:38	<a href="#">dafcbc6f3f3f8ddcbddc9...</a>	<input checked="" type="checkbox"/> elf	<a href="#">Mirai</a>	<a href="#">elf</a> <a href="#">mirai</a>	abuse_ch	<a href="#">Download</a>

Sur cette capture, on voit les fichiers malwares qu'on a téléchargés depuis MalwareBazaar. Ils arrivent d'abord sous forme d'archives ZIP protégées par le mot de passe "infected", puis on les extrait pour obtenir les exécutables (fake\_malware1.exe, fake\_malware2.exe, etc.). On a placé tous ces fichiers dans notre dossier de quarantaine pour pouvoir ensuite les transférer vers la machine Windows via le partage Samba.



### vérification du bon fonctionnement d'inetsim

On a démarré INetSim avec systemctl start inetsim, puis on a utilisé netstat pour vérifier qu'il écoute bien sur les ports nécessaires.

Comme on le voit, plusieurs services (HTTP, DNS, SMTP, FTP, etc.) sont en mode LISTEN sur l'adresse de Kali.

=>INetSim est bien actif et prêt à simuler un faux Internet.

Ça veut dire que lorsque le malware essayera de communiquer, toutes ses requêtes passeront par nous.

```
(salima㉿kali)-[~/malware_quarantine]
$ sudo systemctl start inetsim

(salima㉿kali)-[~/malware_quarantine]
$ sudo netstat -tlnp | grep inetsim
tcp        0      0 192.168.100.1:6667      0.0.0.0:*          LISTEN
  46923/inetsim_irc_6
tcp        0      0 192.168.100.1:995      0.0.0.0:*          LISTEN
  46919/inetsim_pop3s
tcp        0      0 192.168.100.1:990      0.0.0.0:*          LISTEN
  46921/inetsim_ftps_
tcp        0      0 192.168.100.1:37       0.0.0.0:*          LISTEN
  46928/inetsim_time_
tcp        0      0 192.168.100.1:9       0.0.0.0:*          LISTEN
  46934/inetsim_disc
tcp        0      0 192.168.100.1:13      0.0.0.0:*          LISTEN
  46930/inetsim_dayti
tcp        0      0 192.168.100.1:1       0.0.0.0:*          LISTEN
  46940/inetsim_dummy
tcp        0      0 192.168.100.1:7       0.0.0.0:*          LISTEN
  46932/inetsim_echo_
tcp        0      0 192.168.100.1:25      0.0.0.0:*          LISTEN
  46916/inetsim_smtp_
tcp        0      0 192.168.100.1:19      0.0.0.0:*          LISTEN
  46938/inetsim_charg
tcp        0      0 192.168.100.1:17      0.0.0.0:*          LISTEN
  46936/inetsim_quotd
```

Dans cette capture, on vérifie l'état du service INetSim avec systemctl status inetsim. On voit qu'il est active (running), ce qui confirme qu'il fonctionne correctement.

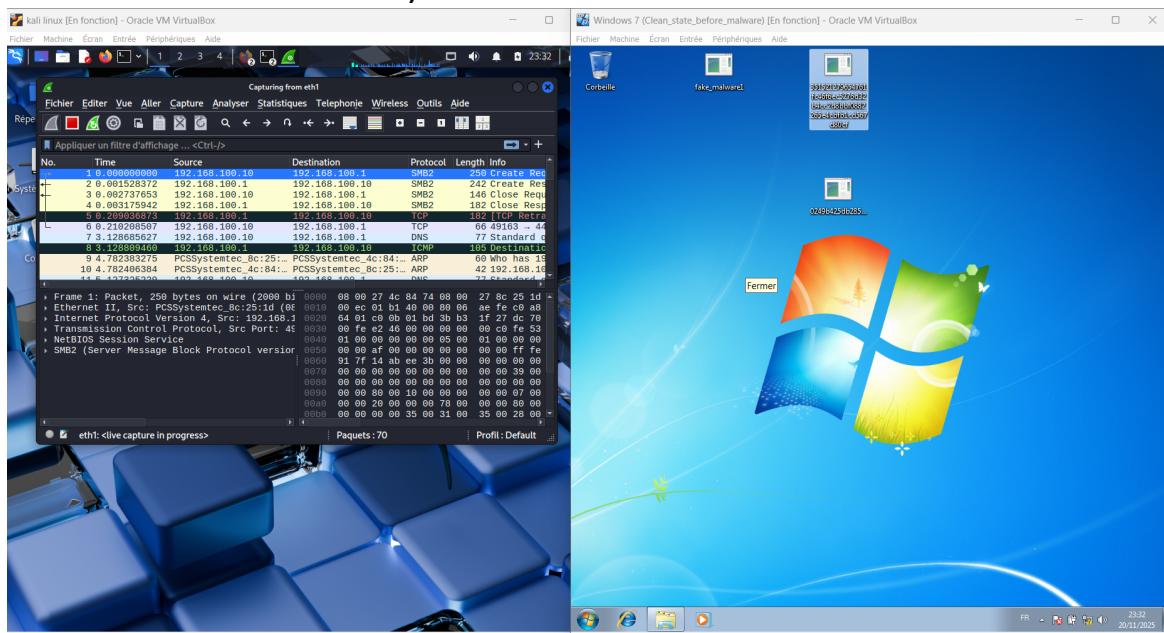
Juste en dessous, la liste des modules actifs apparaît : HTTP, HTTPS, SMTP, FTP, DNS, etc.

```

salima@kali: ~/malware_quarantaine
Session Actions Éditer Vue Aide
(salima@kali)-[~/malware_quarantaine]
$ sudo systemctl status inetsim
● inetsim.service - LSB: start and stop the internet simulation
  Loaded: loaded (/etc/init.d/inetsim; generated)
  Active: active (running) since Thu 2025-11-20 23:05:12 CET; 1min 51s ago
  Invocation: 088aaaf77265043a9992d14936d10f601
    Docs: man:systemd-sysv-generator(8)
  Process: 46892 ExecStart=/etc/init.d/inetsim start (code=exited, status=>
  Tasks: 29 (limit: 2116)
  Memory: 140.1M (peak: 141.8M)
    CPU: 2.964s
  CGroup: /system.slice/inetsim.service
          └─46902 inetsim_main
              ├─46914 inetsim_http_80_tcp
              ├─46915 inetsim_https_443_tcp
              ├─46916 inetsim_smtp_25_tcp
              ├─46917 inetsim_smtps_465_tcp
              ├─46918 inetsim_pop3_110_tcp
              ├─46919 inetsim_pop3s_995_tcp
              ├─46920 inetsim_ftp_21_tcp
              ├─46921 inetsim_ftps_990_tcp
              ├─46922 inetsim_tftp_69_udp
              ├─46923 inetsim_imc_6667_tcp

```

### observation du comportement du malware (le premier malware ne fonctionnait pas mais le deuxième fonctionne)



Quand on exécute le deuxième malware sur Windows (le premier ne montrait aucun signe d'activité), on voit immédiatement du trafic apparaître dans Wireshark sur Kali.

Cela confirme que le malware essaie de communiquer sur le réseau.

Voici ce qu'on observe :

- **Requêtes DNS**

Le malware commence par faire des requêtes DNS.

Ça montre qu'il essaie de contacter un domaine externe, probablement son serveur de commande (C2).

Grâce à INetSim, c'est nous qui répondons, donc on peut voir exactement quel domaine il cherche.

- **ICMP (Ping)**

On voit aussi plusieurs pings.

Le malware vérifie simplement si la connexion réseau est active.

Beaucoup de malwares font ça avant de lancer la suite de leurs actions.

- **ARP**

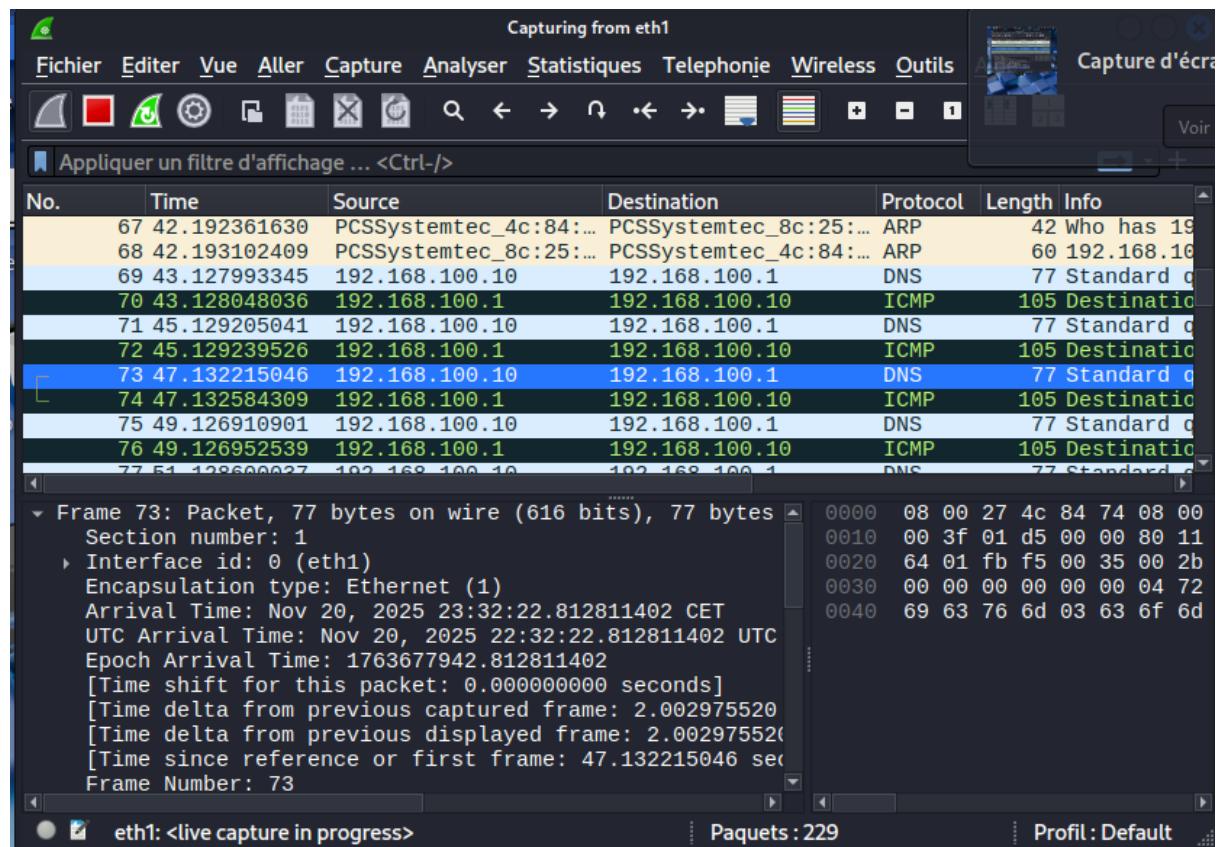
Il envoie des requêtes ARP pour voir qui est présent sur le réseau local.

Ça peut servir à repérer la passerelle, le serveur DNS, ou simplement à vérifier qu'il n'est pas isolé.

- **Trafic TCP inattendu**

On observe du trafic TCP qui ne vient pas de l'utilisateur.

C'est un bon indicateur que le malware tente d'établir une communication vers un serveur extérieur.



The screenshot shows a Wireshark interface capturing traffic from the eth1 interface. The packet list pane displays several frames, mostly DNS requests and responses, indicating network reconnaissance. The details pane shows the structure of a selected frame, which is a DNS request from 192.168.100.1 to 192.168.100.1. The bytes pane shows the raw hex and ASCII data of the selected frame.

No.	Time	Source	Destination	Protocol	Length	Info
67	42.192361630	PCSSystemtec_4c:84:...	PCSSystemtec_8c:25:...	ARP	42	Who has 192.168.100.1 on link layer? [ARP Request]
68	42.193102409	PCSSystemtec_8c:25:...	PCSSystemtec_4c:84:...	ARP	60	192.168.100.1 is at 192.168.100.1 (oui) [ARP Response]
69	43.127993345	192.168.100.10	192.168.100.1	DNS	77	Standard query to 192.168.100.1#53 [DNS Request]
70	43.128048036	192.168.100.1	192.168.100.10	ICMP	105	Destination unreachable [ICMP Echo Reply]
71	45.129205041	192.168.100.10	192.168.100.1	DNS	77	Standard query to 192.168.100.1#53 [DNS Request]
72	45.129239526	192.168.100.1	192.168.100.10	ICMP	105	Destination unreachable [ICMP Echo Reply]
73	47.132215046	192.168.100.10	192.168.100.1	DNS	77	Standard query to 192.168.100.1#53 [DNS Request]
74	47.132584309	192.168.100.1	192.168.100.10	ICMP	105	Destination unreachable [ICMP Echo Reply]
75	49.126910901	192.168.100.10	192.168.100.1	DNS	77	Standard query to 192.168.100.1#53 [DNS Request]
76	49.126952539	192.168.100.1	192.168.100.10	ICMP	105	Destination unreachable [ICMP Echo Reply]
77	51.128600027	192.168.100.10	192.168.100.1	DNS	77	Standard query to 192.168.100.1#53 [DNS Request]

Sur cette capture, on voit clairement que le malware démarre son activité réseau juste après son exécution. Les paquets ARP montrent qu'il commence par repérer les machines présentes sur le réseau, ce qui est une étape classique de reconnaissance. Ensuite, on observe des requêtes DNS envoyées vers notre machine Kali, ce qui prouve que le malware essaie de contacter un domaine externe, très probablement son serveur de commande. On voit aussi du trafic ICMP, ce qui indique qu'il teste la connectivité avant d'aller plus loin. En résumé, cette capture confirme que le malware tente d'établir une communication extérieure et qu'il analyse d'abord l'environnement réseau avant d'envoyer ses requêtes.

```
Internet Protocol Version 4, Src: 192.168.100.10, Dst: 192.168.100.1
User Datagram Protocol, Src Port: 64501, Dst Port: 53
Domain Name System (query)
  Transaction ID: 0x0ffc
  Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  Queries
    rony.publicvm.com: type A, class IN
```

On voit que le malware envoie une requête DNS pour résoudre le domaine rony.publicvm.com. Cela confirme que le programme essaie de contacter un serveur externe, probablement son serveur de commande (C2). Comme la requête est envoyée vers notre machine Kali (port 53), c'est INetSim qui intercepte et répond. Cette étape montre donc le comportement principal du malware : tenter d'établir une communication externe via un domaine spécifique.

Donc au cours de cette attaque on observe à travers wireshark des requêtes dns et icmp répétitives avec des requêtes arp

en analysant ces requêtes de près on constate que le domaine consulté est [rony.publicvm.com](http://rony.publicvm.com)

1. Redirection locale : rony.publicvm.com → 192.168.100.10
2. Scan réseau via ARP → reconnaissance de l'environnement
3. Test de connectivité via ICMP → vérification réseau
4. Redirection Locale → rony.publicvm.com → 192.168.100.10

l'attaque commence à 23:32:22 quelques secondes après l'exécution du malware

## Résultats de l'Analyse

Dans nos tests, le premier malware n'a montré aucune activité réseau, donc il semble non fonctionnel dans notre environnement.

En revanche, le deuxième malware a généré du trafic dès son exécution, ce qui confirme qu'il essaie réellement de communiquer.

Nous avons identifié trois comportements importants :

- le malware tente une communication avec son serveur C2 en utilisant des requêtes DNS

- il réalise un scan du réseau local via ARP et ICMP pour comprendre son environnement
- il utilise ensuite une redirection locale, ce qui montre qu'il essaye de contourner l'absence d'Internet réel

En résumé, le deuxième échantillon adopte un comportement typique d'un malware actif cherchant à établir une connexion externe, tandis que le premier ne présente aucune activité observable.

### Modification du domaine sur inetsim

Dans cette étape, on a ajouté dans INetSim une entrée dns\_static pour que le domaine rony.publicvm.com, utilisé par le malware, soit automatiquement redirigé vers l'adresse 192.168.100.1 (notre Kali).

Grâce à cette redirection, même si le malware essaie d'aller sur son vrai serveur, c'est INetSim qui répond à sa place.

#  
# Static mappings for DNS  
#  
# Syntax: dns\_static <fqdn hostname> <IP address>  
#  
# Default: none  
#  
dns\_static rony.publicvm.com 192.168.100.1  
#dns\_static ns1.foo.com 10.70.50.30  
#dns\_static ftp.bar.net 10.10.20.30

No.	Time	Source	Destination	Protocol	Length	Info
323	246.981181975	PCSSystemtec_4c:84:74	PCSSystemtec_8c:25:00	ARP	42	192.168.100.1
324	248.0766911166	192.168.100.10	192.168.100.1	DNS	77	Standard query
325	248.076720575	192.168.100.1	192.168.100.10	ICMP	105	Destination unreachable
326	250.078478523	192.168.100.10	192.168.100.1	DNS	77	Standard query
327	250.078509422	192.168.100.1	192.168.100.10	ICMP	105	Destination unreachable
328	252.076706288	192.168.100.10	192.168.100.1	DNS	77	Standard query
329	252.076739355	192.168.100.1	192.168.100.10	ICMP	105	Destination unreachable
330	253.216027688	PCSSystemtec_4c:84:74	PCSSystemtec_8c:25:00	ARP	42	Who has 192.168.100.1
331	253.217286239	PCSSystemtec_8c:25:00	PCSSystemtec_4c:84:74	ARP	60	192.168.100.1
332	254.076504998	192.168.100.10	192.168.100.1	DNS	77	Standard query
333	254.076542249	192.168.100.1	192.168.100.10	ICMP	105	Destination unreachable

On observe encore du trafic généré par le malware après son exécution.

Les lignes ARP sont particulièrement importantes : on voit que la machine Windows (adresse MAC commençant par PCSSystemtec) envoie des requêtes "Who has 192.168.100.1 ?". Cela veut dire qu'elle cherche l'adresse physique (MAC) de la machine

Kali, qui joue le rôle de serveur DNS/Internet simulé. Ce comportement est typique d'un malware qui vérifie la présence du serveur qu'il veut contacter.

Juste au-dessus, on retrouve aussi du DNS et du ICMP, ce qui montre que le malware continue son cycle normal :

- reconnaître le réseau → trouver sa passerelle → résoudre son domaine → tester la connexion.

## Conclusion

À la fin de notre analyse, nous avons constaté que le premier malware testé ne produisait aucune activité notable. En revanche, le second a montré un comportement réseau clair et répétitif : requêtes DNS vers un domaine spécifique, pings ICMP pour vérifier la connectivité, et requêtes ARP pour analyser le réseau local. Grâce à INetSim, nous avons pu rediriger tous ses essais de communication et ainsi observer son fonctionnement sans aucun risque.

Cette expérience nous a permis de mieux comprendre la manière dont un malware tente d'interagir avec son environnement et comment des outils comme INetSim et Wireshark peuvent être utilisés pour analyser ces comportements de façon sécurisée. Le TP nous a également montré l'importance d'un environnement d'analyse bien isolé et bien configuré pour éviter toute infection réelle.