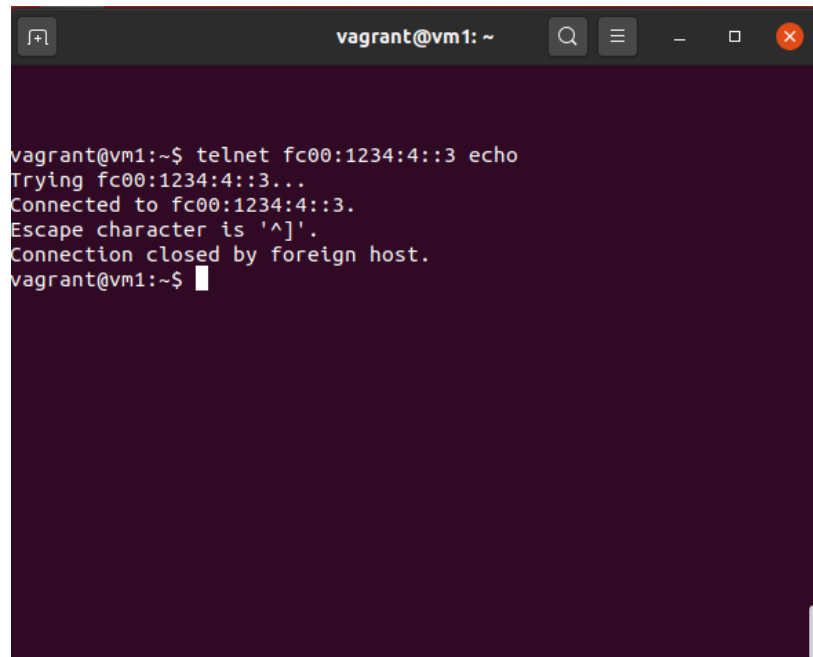


# Rapport projet de Réseaux

## I. Configuration réseau

### 1.1 Topologie et adressage



```
vagrant@vm1: ~  
vagrant@vm1:~$ telnet fc00:1234:4::3 echo  
Trying fc00:1234:4::3...  
Connected to fc00:1234:4::3.  
Escape character is '^]'.  
Connection closed by foreign host.  
vagrant@vm1:~$
```

Figure 1 Serveur echo VM1 vers VM3 ipv4



```
vagrant@vm1: ~  
vagrant@vm1:~$ sudo telnet fc00:1234:4::36 echo  
Trying fc00:1234:4::36...  
Connected to fc00:1234:4::36.  
Escape character is '^]'.  
test  
test  
concluant pour vm3-6  
concluant pour vm3-6
```

Figure 2 Serveur echo VM1 vers VM3 ipv6

4. Il est possible de n'écouter qu'en ipv6 sur le serveur echo de VM3. Pour cela, lors de la mise en place du serveur avec l'outil *Inetutils*, il faut configurer le serveur uniquement en tcp6.

```
VM3-6 # update-inetd --add "echo stream tcp6 nowait nobody internal"
```

## II. L'interface virtuelle TUN

### 2.1 Création de l'interface

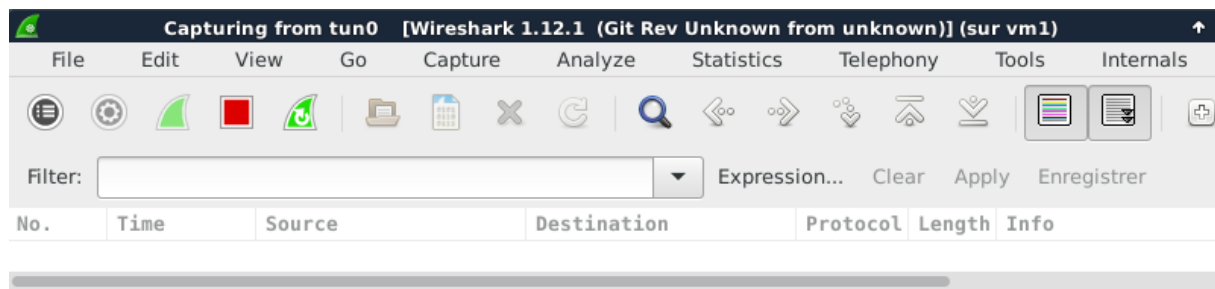
La fonction *createTun()* présente dans la bibliothèque *lftun* permet de créer un tunnel nommé *tun0*.

### 2.2 Configuration de l'interface

1. Le script Bash *configure-tun.sh* présent dans le dossier partage permet de configurer *tun0* avec l'adresse *fc00:1234:ffff::1*. Le masque adéquat ici est 64.

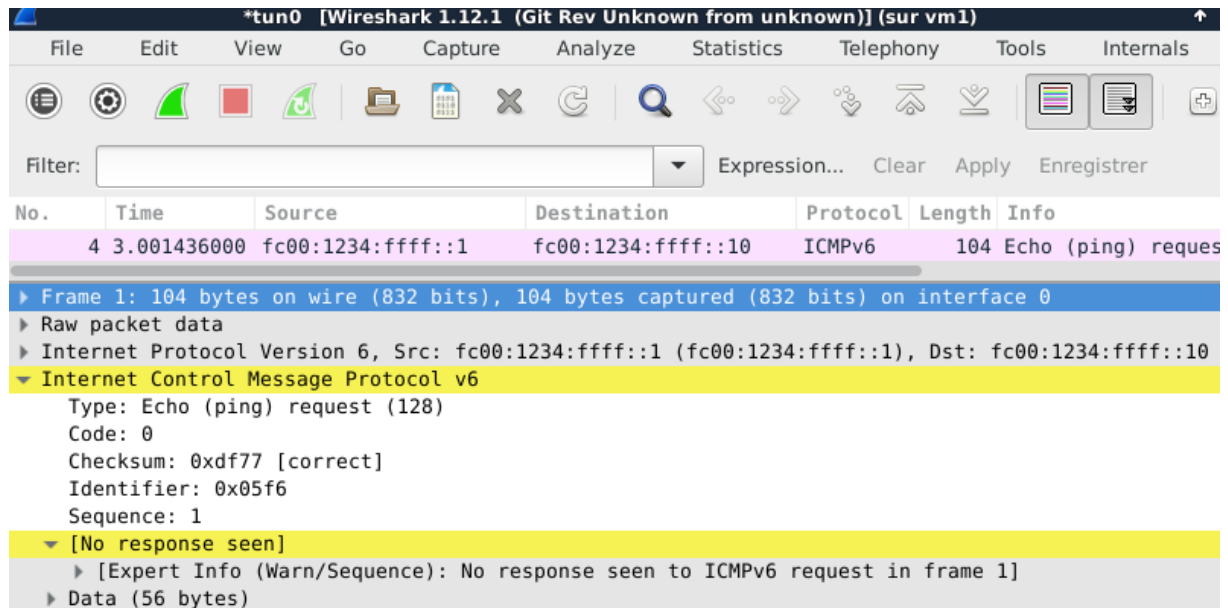
2. Les modifications de routage sont à faire sur VM1-6. Il faut lui enlever l'interface réseau qui le liait à VM2-6. De plus, Pour pouvoir accéder à LAN4-6, la VM1-6 devra passer par VM1 en mettant en *gateway* l'adresse de *tun0*.

3.



On remarque que Wireshark ne capte aucune trame alors que les envois de paquets vers *tun0* se font sans problèmes. Quand une interface se ping elle-même, les paquets passent dans loopback ce qui explique l'absence de paquet dans Wireshark.

4.



On constate ici la présence de trames qui indiquent l'échec d'envoi des paquets vers `fc00:1234:ffff::10`. Cela s'explique par le fait que l'envoi passe par l'interface `tun0`. L'adresse requise étant inexistante, il est normal de recevoir ce type de trame sur Wireshark.

## 2.3 Récupération des paquets

1. La fonction `rwPacket()` présente dans la bibliothèque `lftun` sert à écrire dans un descripteur de fichier tout le flux lu par le descripteur de fichier de `tun0`.

2.

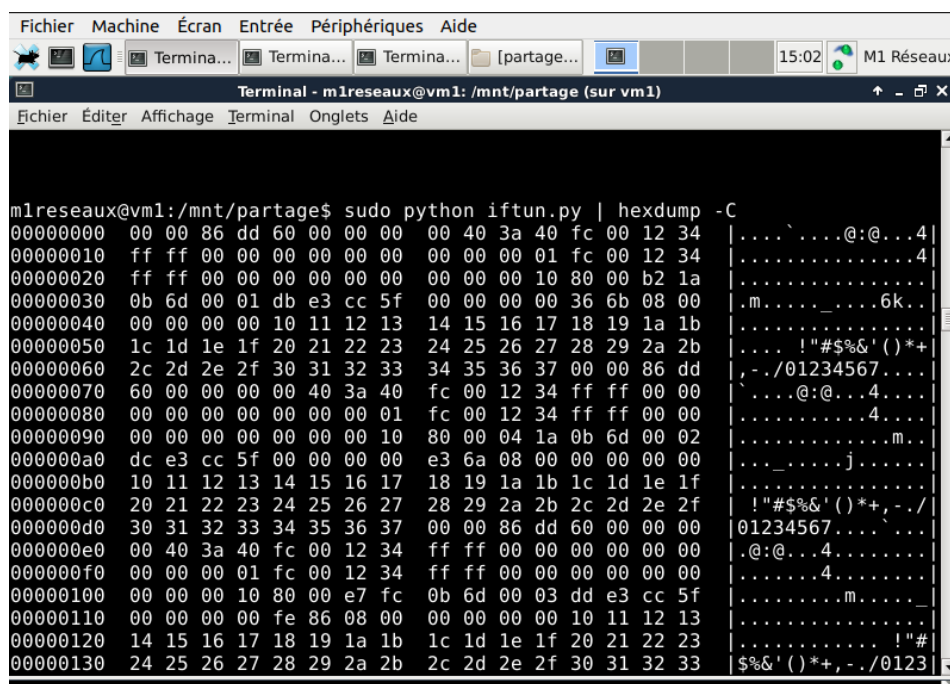


Figure 3 sortie du programme `rwPacket()` sur l'adresse `fc00:1234:ffff::10`

3. Le ping sur l'adresse même du *tun0* n'affiche rien.

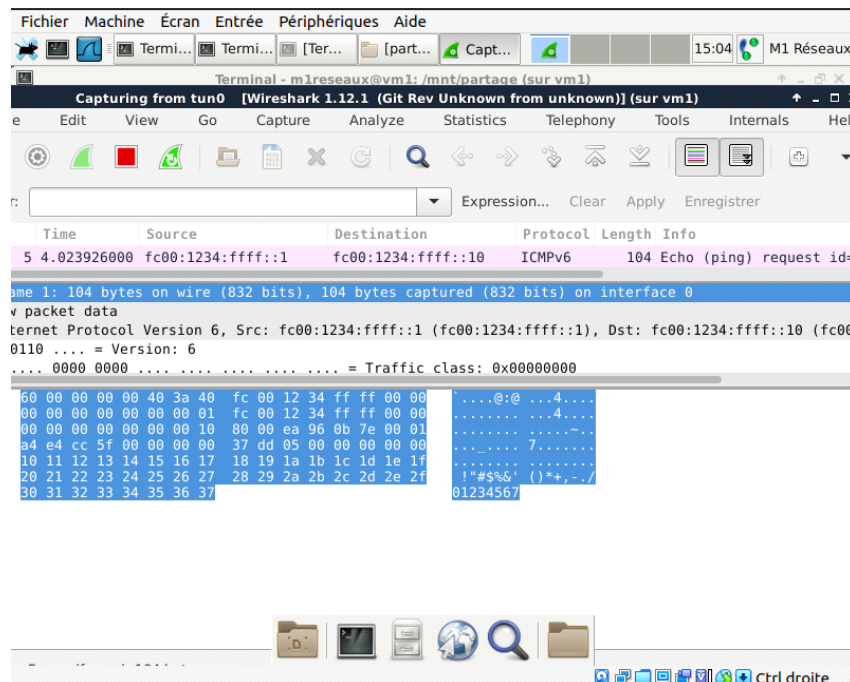


Figure 4 capture Wireshark lors du `rwPacket()` sur l'adresse `fc00:1234:ffff::10`

Le type de trafic constaté est de l'ICMPv6. On remarque que les frames capturés par Wireshark ainsi que le retour de la fonction `rwPacket` dans la sortie standard affiche les mêmes séries de caractères.

4. l'option `IFF_NO_PI` permet de ne rajouter aucune information sur les paquets envoyés (protocole, flags ...). Si l'on ajoute cette option, les paquets reçus seront plus léger.

## III. Un tunnel simple pour IPv6

### 3.1 Redirection du trafic entrant

1. La fonction `ext_out` se situe dans la bibliothèque *extrémité*. Elle permet de lancer un serveur en écoute sur le port renseigné.

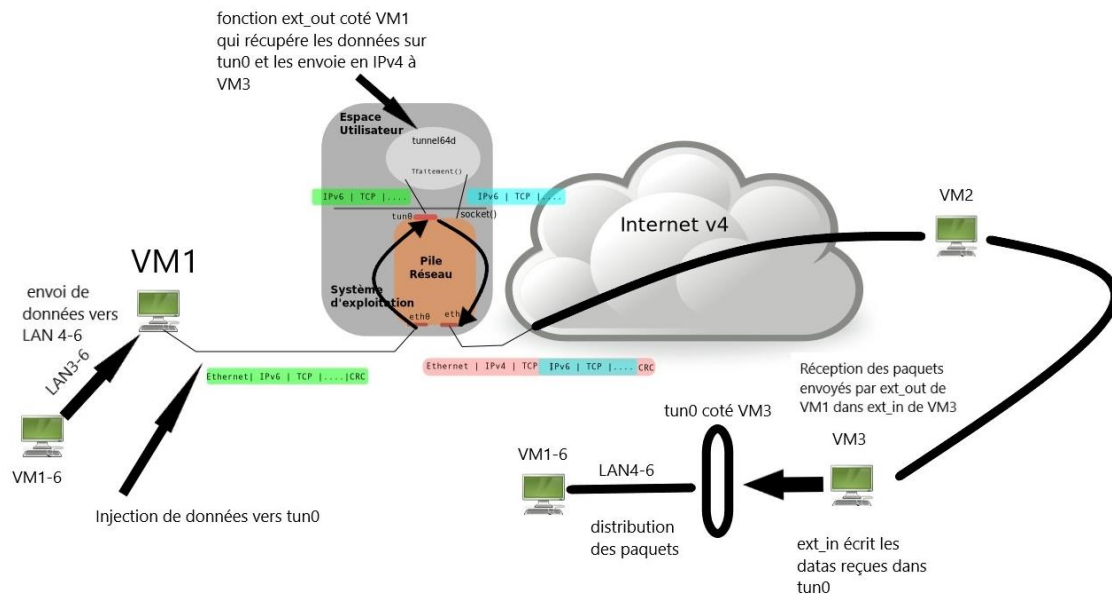
2. La fonction `ext_in` se situe elle aussi dans la même bibliothèque. Elle permet d'établir une connexion avec un serveur en lui indiquant son adresse et le port adéquat.

3. On dispose un serveur sur VM3 et un client sur VM1 en attribuant les bonnes adresses avec les fonctions `ext_out` et `ext_in` citées auparavant.



2. L'utilisation de Thread a été nécessaire pour pouvoir mettre en place la bidirectionnalité, les communications sont donc bien asynchrones.

### 3.4 Mise en place du tunnel entre VM1 et VM3



### 3.5 Mise en place du tunnel entre VM1 et VM3 : Système

1. Nous avons mis en place l'exécutable tunnel64d qui lie un fichier de configuration entré en paramètre et configure un tunnel ainsi qu'un serveur et un client sur l'appareil où il est exécuté. Nous avons donc deux fichiers de configuration à disposition dans le dossier partagé l'un pour VM1 et l'autre pour VM6.

Après avoir mis en place les deux configurations, nous pouvons constater le résultat sur les images suivantes :

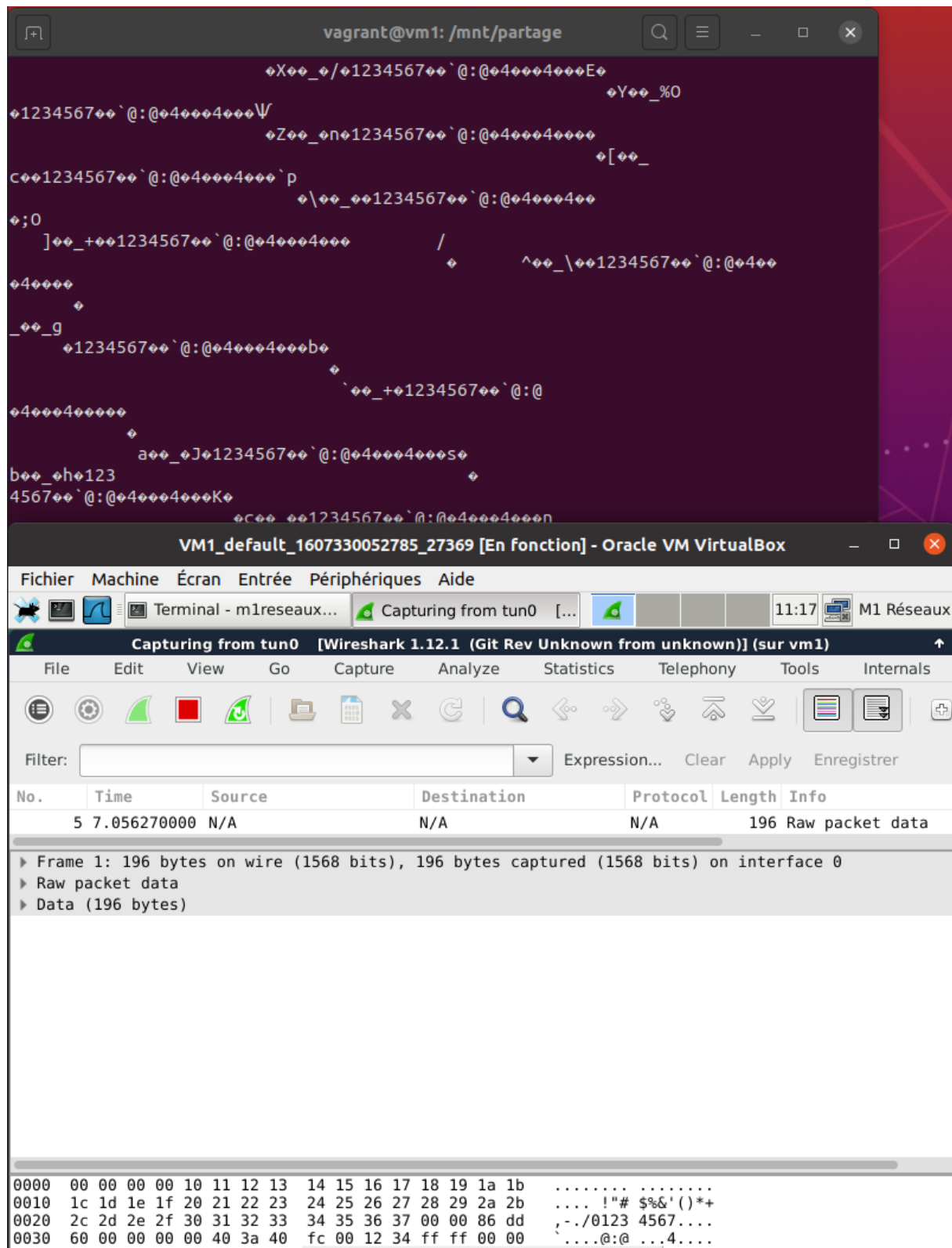


Figure 7 Réception de données sur la VM1 et affichage sur le sorite standard et Wireshark

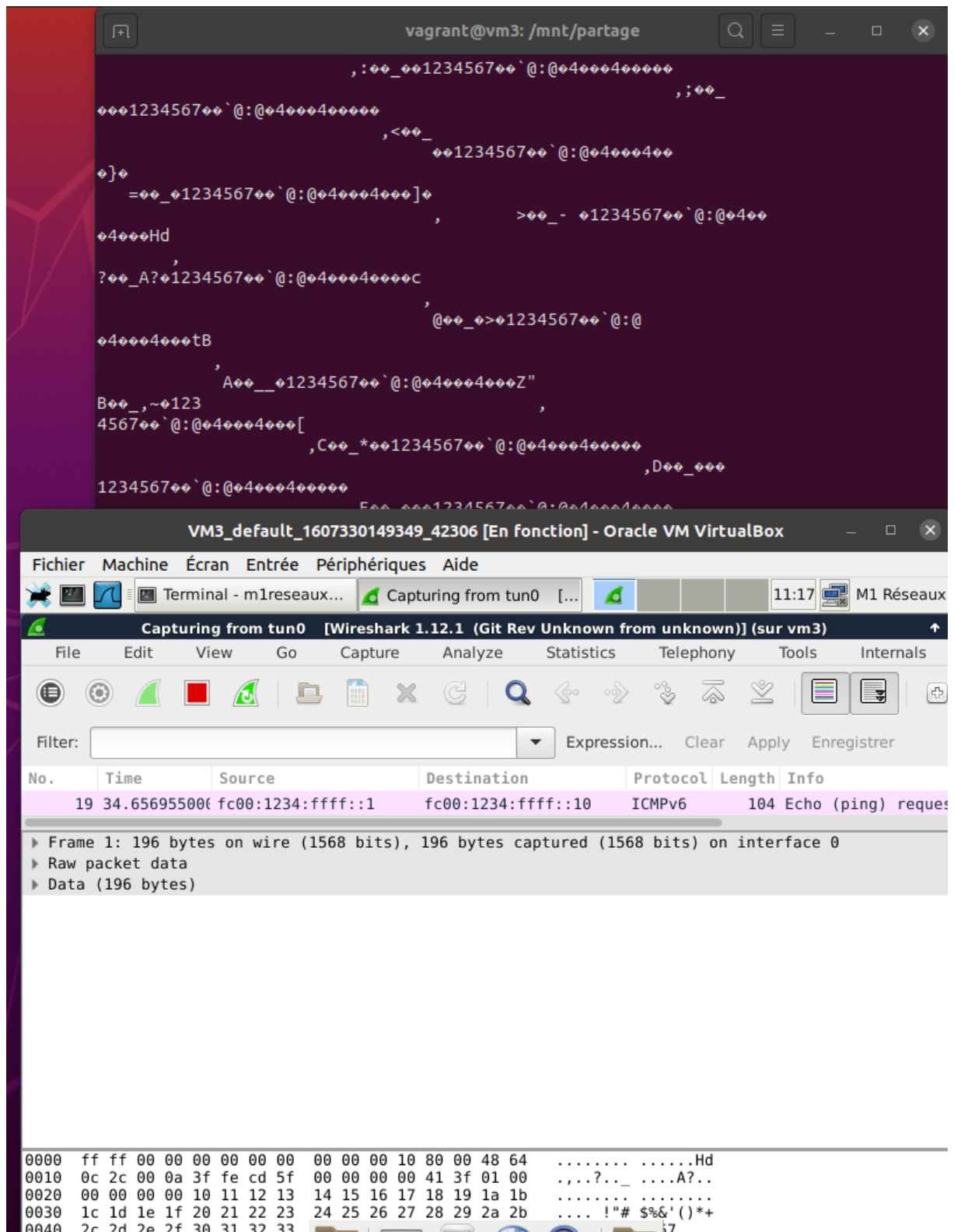


Figure 8 Réception de données sur la VM3 et affichage sur le sorite standard et Wireshark