PROJET MALWARE

2024

Rédigé et audité par :
MESSAOUDI Ilyasse

Table des matières

Rappel du cadrage

Contexte

Vous venez tout juste d'être embauché en tant qu'analyste forensique dans le CERT ESDCorp, l'une des écoles dont vous surveillez les réseaux a été frappé par un Ransomware.

Selon la direction de l'école, un fichier critique permettant de renouveler la certification RNCP du diplôme a été chiffré !
En effet, depuis quelques jours les attaques de ransomware semblent cibler particulièrement les secteurs liés à l'éducation.

Le niveau de l'attaquant qui est présumé responsable de cette attaque semble assez faible et votre manager espère que l'attaquant aura fait des erreurs vous permettant de parvenir à déchiffrer les fichiers

Chain of Custody

Chain of Custody en investigation forensic est un processus qui assure l'intégrité et la traçabilité des preuves numériques de leur collecte à leur présentation.

Elle documente chaque manipulation des preuves, incluant qui les a collectées, quand, comment et où elles ont été stockées.
Ce processus est essentiel pour garantir l'admissibilité des preuves en justice et prévenir toute altération qui pourrait compromettre leur fiabilité.

Acquisition

L'acquisition des données implique la collecte d'informations de manière rigoureuse pour garantir qu'elles sont représentatives de la réalité sans modification

De ce fait, les données ont été fournies par l'équipe du CERT ESDCorp, garantissant leur intégrité et leur authenticité dès la source.

Préservation

Pour préserver les données, j'ai utilisé une machine virtuelle sous Kali Linux.

Cette configuration permet d'utiliser des outils spécialisés tels que Wireshark et Volatility.
Pour garantir que les artefacts restent authentiques et non altérés, je vérifie la valeur du hash MD5.

Elément	HASH MD5
ZIP contenant les éléments ci-dessous (infected.zip)	0c11d4d6a6eaa801aca427cacafb8400
Dump mémoire (dump.raw)	7708f8b427f3f4399d76e4fbdc5dffc7
secret.zip.encrypted (fichier chiffré)	db5c3c3b82e431bb16a3dc13facc82a2
Capture réseau (capture.pcapng)	2d63728280a0e4a95f46f68d21f3bdd9

Réalisation

IOC identifiées

IOC,Indicator of Compromise (=indicateur de compromission), est un indice ou un artefact utilisé pour détecter des signes de compromission dans un système informatique.
Cela peut être une signature viral, une adresse IP ou même un hash de fichier malveillant.

Informations du système à analyser

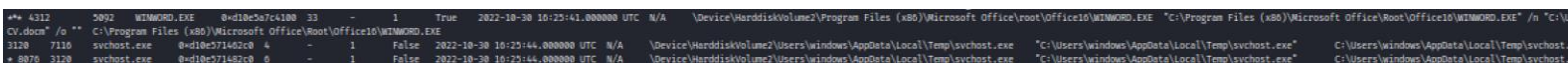
Nom de l'hôte:	WIN01
Nom du système d'exploitation:	Microsoft Windows 10 Professionnel
Version du système:	10.0.19045 N/A version 19045
Fabricant du système d'exploitation:	Microsoft Corporation
Configuration du système d'exploitation:	Station de travail membre
Type de version du système d'exploitation:	Multiprocessor Free
Propriétaire enregistré:	windows
Organisation enregistrée:	
Identificateur de produit:	00330-80000-00000-AA583
Date d'installation originale:	02/06/2022, 23:44:06
Heure de démarrage du système:	30/10/2022, 23:50:28
Fabricant du système:	innotek GmbH
Modèle du système:	VirtualBox
Type du système:	x64-based PC
Processeur(s):	1 processeur(s) installé(s).
	[01] : AMD64 Family 25 Model 33 Stepping 0
AuthenticAMD ~3793 MHz	
Version du BIOS:	innotek GmbH VirtualBox, 01/12/2006
Répertoire Windows:	C:\Windows

Étant donné que nous avons un dump d'une machine Windows et des informations sur le système, j'utilise l'outil Volatility pour obtenir un ensemble d'informations, à partir desquelles je peux extraire des IOC.

En examinant les processus présents dans le dump, j'ai identifié deux processus suspects.

Tout d'abord, le processus avec l'ID 4312, qui exécute WINWORD.EXE avec un fichier nommé CV.docm.

Le nom et l'extension du fichier suggèrent qu'il pourrait s'agir d'un processus illégitime, possiblement une macro malveillante fonctionnant en arrière-plan.

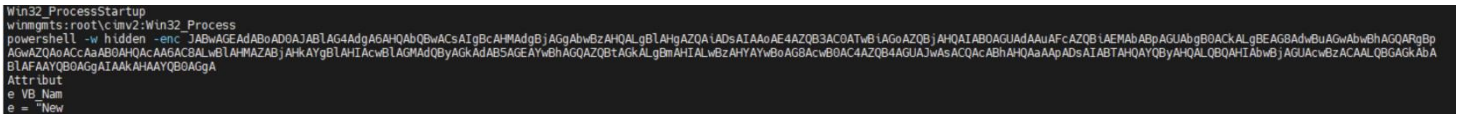


Dans un second temps, le service nommé svchost.exe, qui semble légitime, est présent. Cependant, le fait que ce programme soit situé dans un chemin de destination semblant très suspect et indique une possible activité malveillante, c:\users\windows\appdata\local\temp\svchost.exe.

L’idée est simple : l'attaquant vise à masquer son attaque en usurpant le nom du processus légitime svchost.exe, qu'il utilise pour se faire passer pour un service essentiel du système.

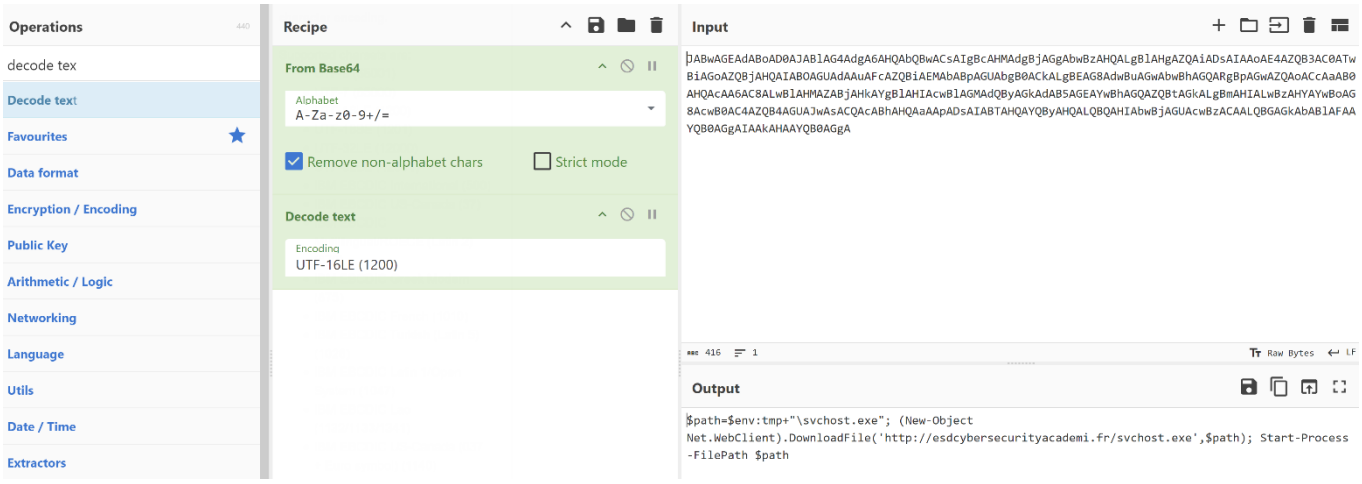
Maintenant que nous savons qu'il est nécessaire de dumper le fichier nommé CV.docm, je vais le récupérer pour en extraire le code source.

Une fois cela fait, voici l'ensemble des chaînes de caractères extraites du processus associé à winword.exe.

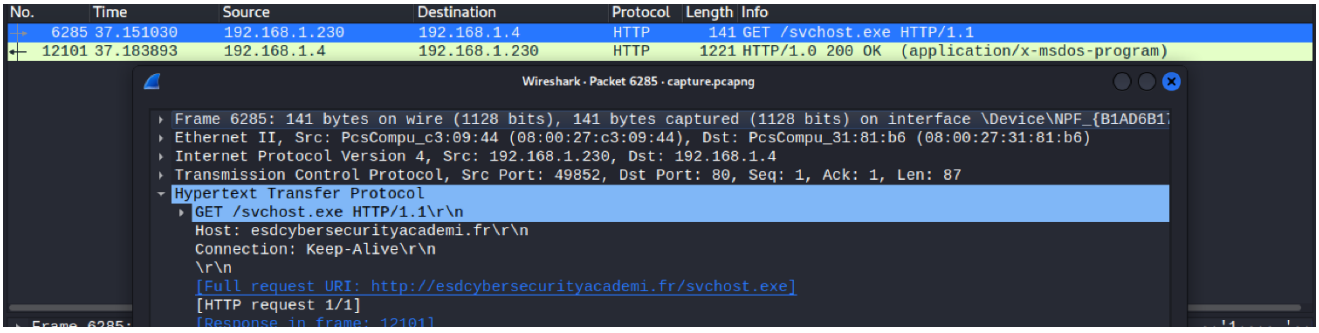


La commande « powershell -w hidden -enc » attire mon attention, car elle indique la présence d'un payload encodé.

Il est facile de décoder ce payload en utilisant l'outil CyberChef pour retrouver la forme initiale du code.



En lien avec la capture Wireshark fournie lors de l'enquête, nous pouvons clairement observer la trace réseau correspondant à ce téléchargement. Cela confirme que le fichier malveillant svchost.exe a été récupéré depuis l'URL spécifiée.



Désormais, on comprend que le fichier CV.docm en cours d'exécution contient un code PowerShell encodé qui télécharge et exécute un fichier malveillant nommé svchost.exe.

En creusant dans les processus, svchost.exe pid 8076 charge une dll se nommant python.

```
┌──(kali㉿kali)-[~/Desktop/artefacts]
└─$ python3 ../volatility3/vol.py -f dump.raw windows.dlllist --pid 8076 | grep 'python'
8076resssvchost.exe     0×7ffe36010000an0×3c1000ished   python37.dll    C:\Users\windows\AppData\Local\Temp\_MEI31202\python37.dll      2022-10-30 16:25:44.000000 UTC  Disabled
8076    svchost.exe     0×7ffe36a20000  0×f000  python3.DLL     C:\Users\windows\AppData\Local\Temp\_MEI31202\python3.DLL        2022-10-30 16:25:44.000000 UTC  Disabled
```

Identifications des malwares

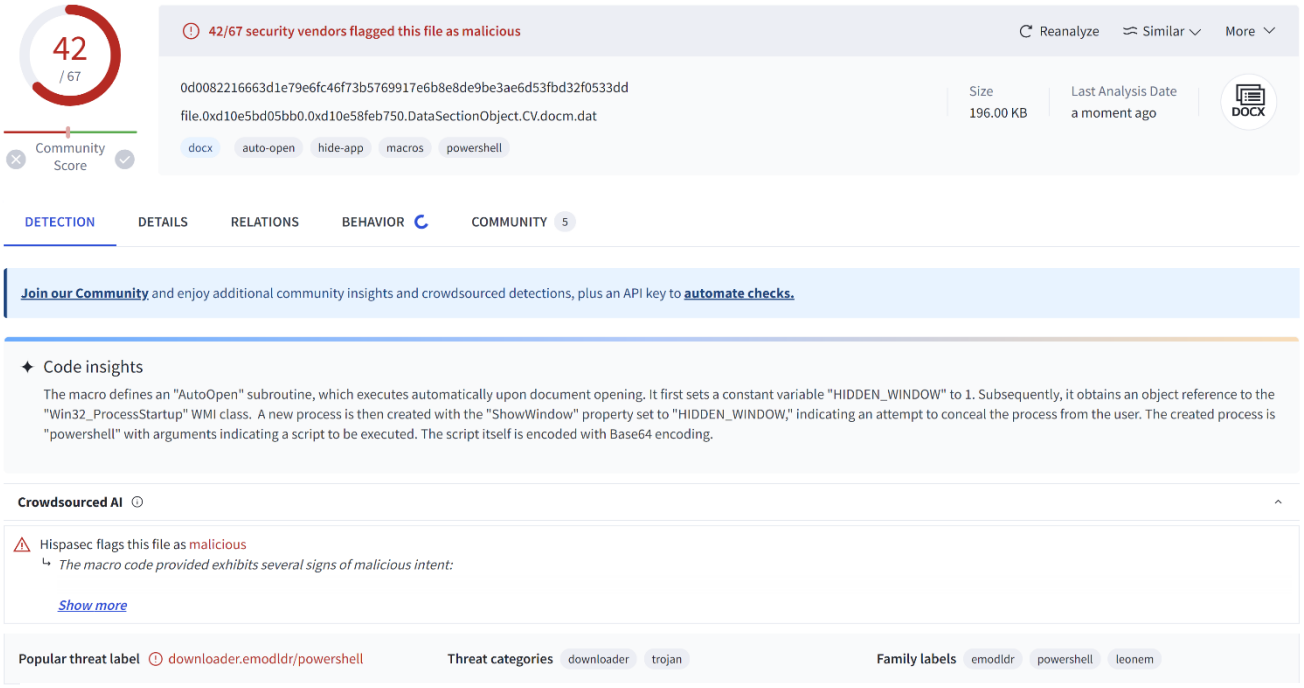
Comme mentionné précédemment, nous avons identifié deux processus potentiellement malveillants.
Le premier est cv.docm, un downloader qui, lorsqu'il est ouvert, active une macro qui télécharge et exécute un fichier exécutable. Comme observé dans la capture Wireshark, le fichier téléchargé est svchost.exe.

Le second processus suspect est ce même svchost.exe, qui utilise la technique du Masquerading en usurpant un processus légitime, bien que ses actions soient malveillantes, utilisant notamment une dll illégitime se nommant python.

Malware du fichier CV

Après avoir effectué un dump du processus malveillant lié à cv.docm, j'ai récupéré le fichier cv.docm pour en extraire et comparer son hash sur la plateforme VirusTotal. Cela m'a permis d'évaluer la gravité du fichier et de déterminer sa nature malveillante.

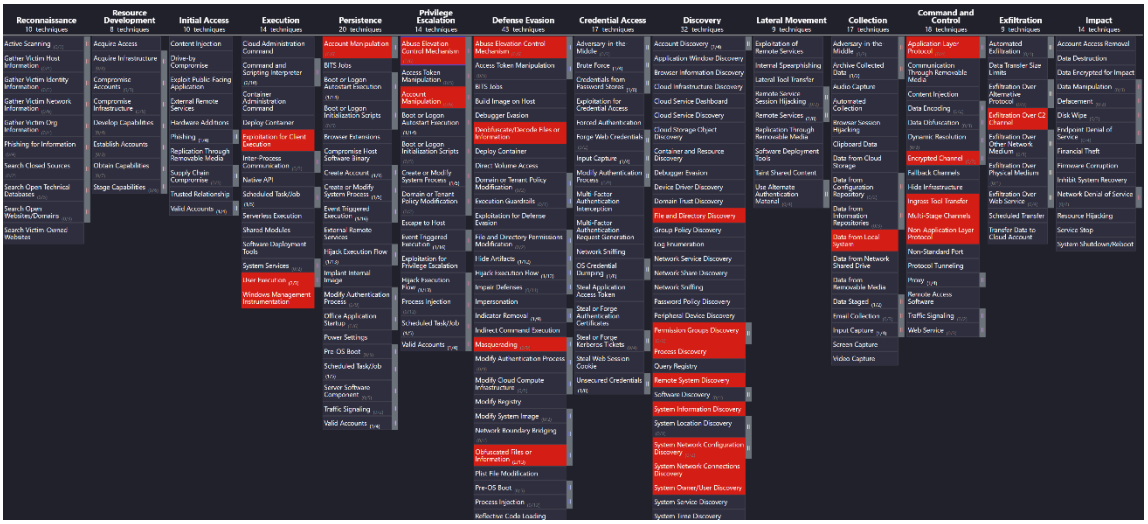
```
kali@kali:~/Desktop$ ls cv/ | grep 'docm'
file.0×d10e5bd05bb0.0×d10e58feb750.DataSectionObject.CV.docm.dat
file.0×d10e5bd05bb0.0×d10e59d95260.SharedCacheMap.CV.docm.vacb

kali@kali:~/Desktop$ md5sum cv/file.0×d10e5bd05bb0.0×d10e58feb750.DataSectionObject.CV.docm.dat
5fd8f2b5f9a6f4cff487bbc1ee5fdc3c  cv/file.0×d10e5bd05bb0.0×d10e58feb750.DataSectionObject.CV.docm.dat
```



https://www.virustotal.com/gui/file/0d0082216663d1e79e6fc46f73b5769917e6b8e8de9be3ae6d53fbd32f0533dd

TTP CV.docm

Avec l'outil MITRE ATT&CK Navigator on peut créer notre contexte, ici un exemple pour le malware cv.docm.

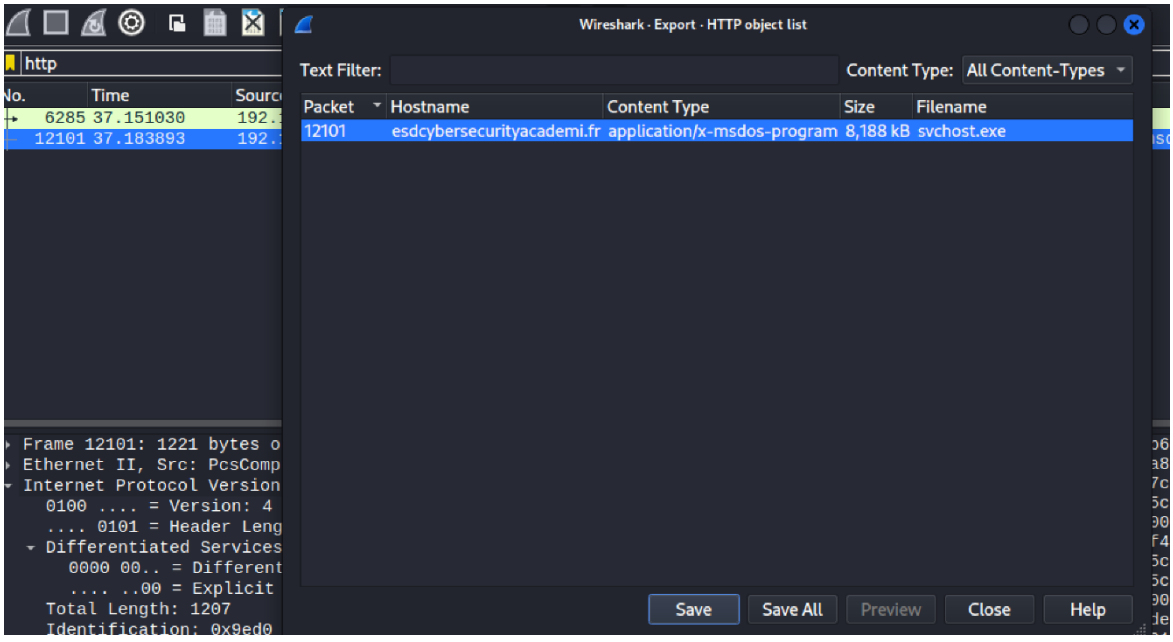


Cette analyse permet d'améliorer la détection des menaces en utilisant les techniques MITRE ATT&CK.

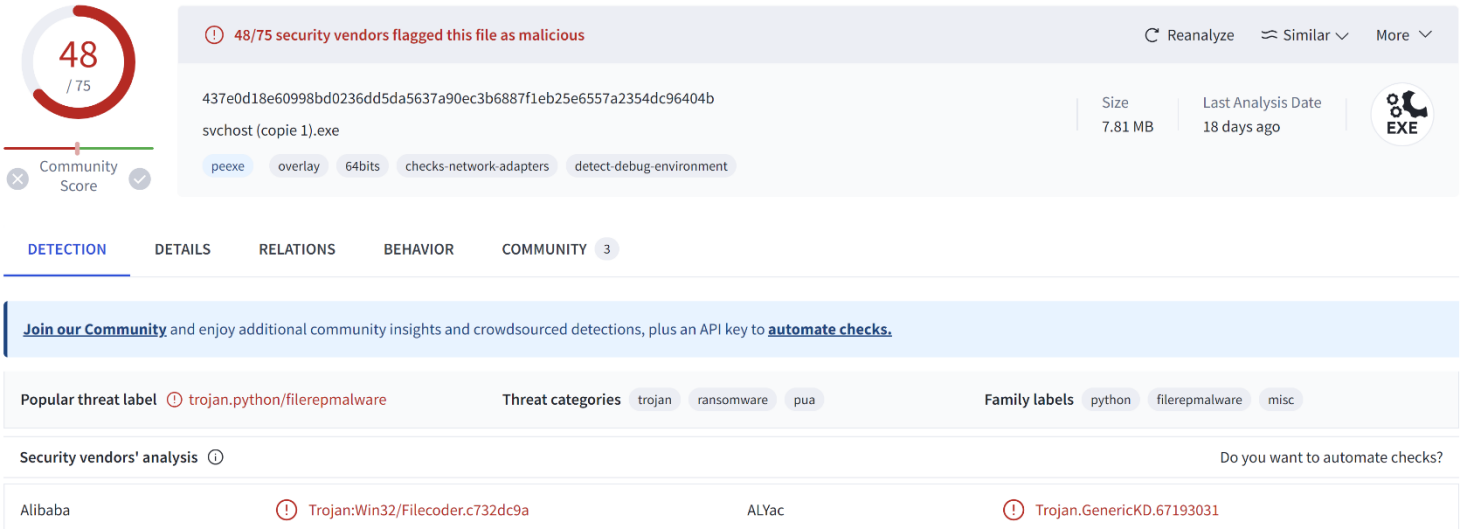
En utilisant ce cadre comme référence, les investigateurs, comme nous, peuvent améliorer leur posture de sécurité en se concentrant sur les techniques les plus pertinentes pour leurs environnements et en proposant des stratégies de réponse ciblées aux clients.

Malware du service svchost.exe

Pour analyser la potentielle malveillance, comme avec le fichier précédent, je vais extraire le fichier svchost.exe, récupérer son hash, et le vérifier sur VirusTotal.



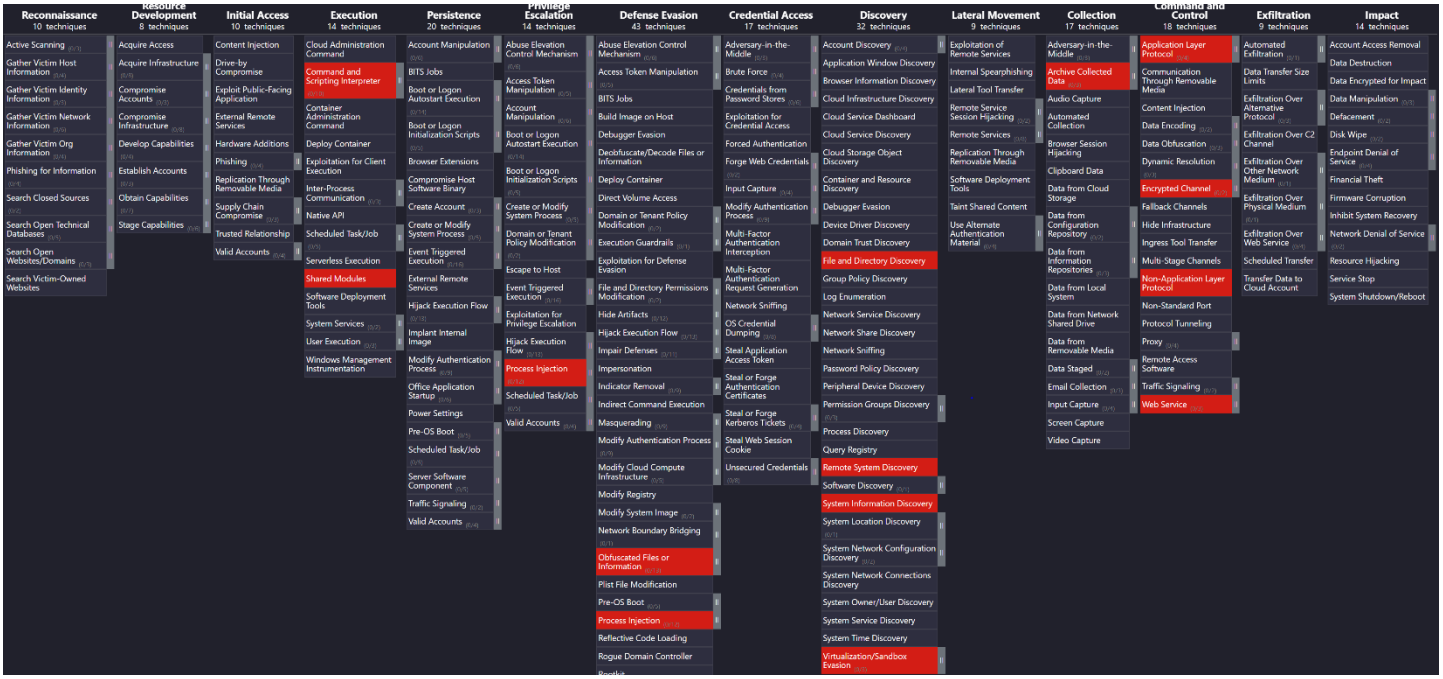
```
kali@kali:~/Desktop/artefacts$ md5sum svchost.exe
9b0ba3738c994d5850d97077d578d3bc  svchost.exe
```



https://www.virustotal.com/gui/file/437e0d18e60998bd0236dd5da5637a90ec3b6887f1eb25e6557a2354dc96404b

TTP de svchost.exe

Comme précédemment, voici une visualisation avec MITRE ATT&CK qui offre une vue détaillée des différentes attaques et techniques associées à cet exécutable.



Plan d'actions correctives

Maintenant que nous avons identifié le script Python comme responsable du ransomware et l'avons extrait du dump.
Je peux utiliser l'outil pyinstxtractor pour décompiler l'exécutable .exe, afin d'obtenir le code source.

Cela me permettra d'entamer un travail de reverse engineering et peut-être de découvrir la clé de déchiffrement pour tous les fichiers locks.

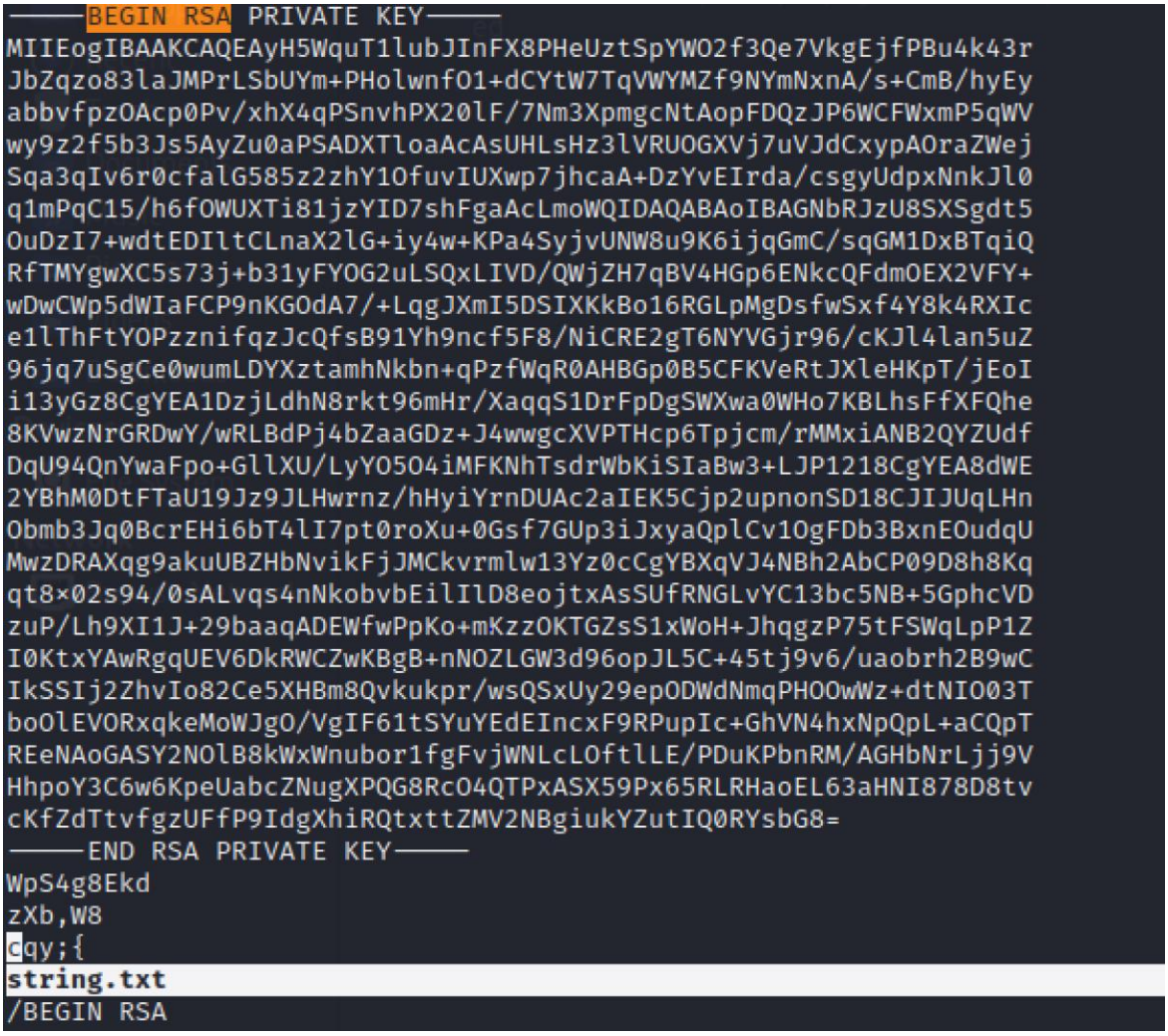
```
┌──(kali㉿kali)-[~/Desktop/artefacts]
└─$ python3 ../test/pyinstxtractor/pyinstxtractor.py svchost.exe
[+] Processing svchost.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 3.7
[+] Length of package: 7867311 bytes
[+] Found 38 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: pyi_rth_subprocess.pyc
[+] Possible entry point: encrypt.pyc
[!] Warning: This script is running in a different Python version than the one used to build the executable.
[!] Please run this script in Python 3.7 to prevent extraction errors during unmarshalling
[!] Skipping pyz extraction
[+] Successfully extracted pyinstaller archive: svchost.exe

You can now use a python decompiler on the pyc files within the extracted directory
```

Voici le code source complet du ransomware.
En examinant de plus près nous voyons que le ransomware, par son essence possède une clé privée et clé publique.

Je vais maintenant rechercher toutes les occurrences potentielles dans le dump à la recherche éventuelle d'une clé laissée en clair sur la machine.

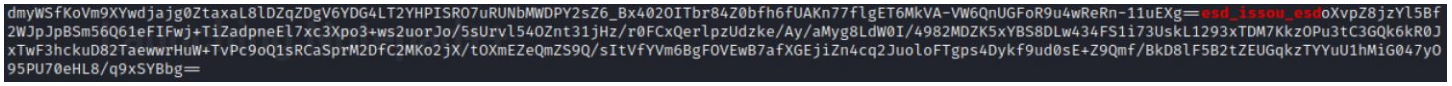
```
┌──(kali㉿kali)-[~/Desktop/artefacts/svchost.exe_extracted]
└─$ /home/kali/.local/bin/uncompyle6 encrypt.pyc
# uncompyle6 version 3.9.2
# Python bytecode version base 3.7.0 (3394)
# Decompiled from: Python 3.11.4 (main, Jun  7 2023, 10:13:09) [GCC 12.2.0]
# Embedded file name: ransomware\encrypt.py
from os import path, walk, rename
import cryptography
from cryptography.fernet import Fernet
import requests
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import utils, padding
import base64, time

def generate_secret_key():
    key = Fernet.generate_key()
    return key


def encrypt_secret_key(public_key, secret_key):
    public_key_bytes = serialization.load_pem_public_key(public_key.encode())
    encrypted_secret_key = public_key_bytes.encrypt(secret_key, padding.OAEP(mgf=padding.MGF1(algorithm=(hashes.SHA256())),
      algorithm=(hashes.SHA256()),
      label=None))
    encrypted_secret_key = base64.b64encode(encrypted_secret_key)
    return encrypted_secret_key


def encrypt_file(file, public_key):
    errors = []
    key = generate_secret_key()
    encrypted_secret_key = encrypt_secret_key(public_key, key)
    f = Fernet(key)
```

Bingo ! En cherchant dans le dump, nous avons trouvé une clé privée. Il est maintenant nécessaire de la faire correspondre avec la clé publique pour récupérer le mot de passe du fichier zip.

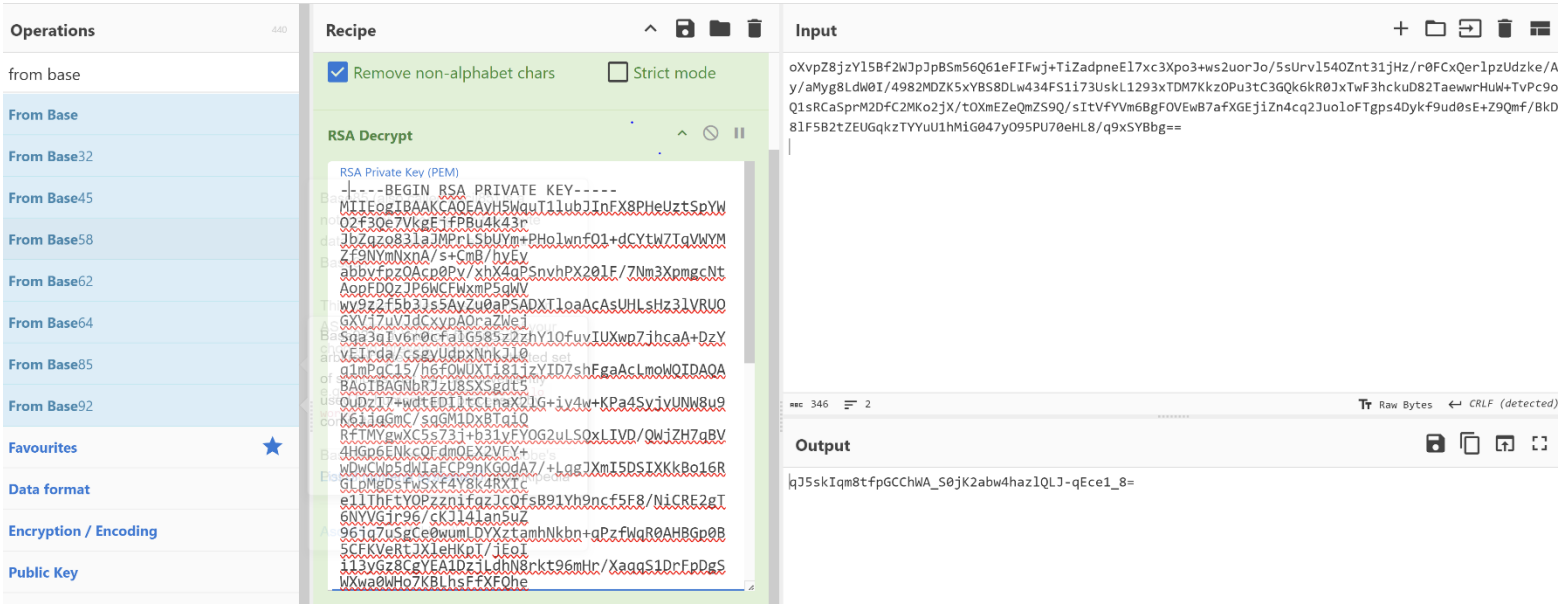


Comme l'indique le code, une ligne esd_issou_esd est inscrite juste avant la clé de chiffrement publique, qui ici est encodé en base64.

Désormais il nous reste plus qu'à faire correspondre avec la clé privée afin de récupérer le secret nécessaire au déchiffrement de l'archive.



Une fois de plus, en utilisant l'outil CyberChef, nous avons pu extraire le secret dans la zone de sortie.



Grâce à un script, nous avons pu déchiffrer l'archive et accéder à toutes les ressources qu'elle contenait.

```
 GNU nano 7.2                                                script.py
from cryptography.fernet import Fernet

with open('key.txt', 'rb') as key_file:
    key = key_file.read().strip()

chiffre = Fernet(key)

with open('secret.zip.encrypted', 'rb') as encr_file:
    encrypted_data = encr_file.read()

try:
    decrypted_data = chiffre.decrypt(encrypted_data)
    with open('secret.zip', 'wb') as decrypted_file:
        decrypted_file.write(decrypted_data)
    print("OK")
except Exception as e:
    print(f"Erreur {e}")
```

```
┌──(kali㉿kali)-[~/Desktop/artefacts]
└─$ python3 script.py
OK

┌──(kali㉿kali)-[~/Desktop/artefacts]
└─$ unzip secret.zip
Archive:  secret.zip
replace flag.png? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
  inflating: flag.png
  inflating: photo_importante.png
```

Conclusion

Deux flags s'y trouvaient, marquant la conclusion de l'investigation.