Forensics

Agenda

- Introduction/review
- What is forensics?
- Steganography:
    - Files structure/formats & metadata
    - Files carving/extracting
    - Image steganography
- Network capture analysis
    - Wireshark and how to read packets
- Memory Forensics (little glance)

Introduction/Review

Data encoding

Plaintext: CSC{flag}
Hexadecimal (Base-16): 4353437b666c61677d0d0a
Base64 (and other bases): Q1NDe2ZsYWd9DQo=
Binary: 01000011 01010011 01000011 01111011 01100110 01101100 01100001 01100111 01111101 00001101 00001010

Introduction/Review

Web requests

Method: GET/POST/PUT (etc…)

Request headers: includes information about the web request, sometimes even cookies or credentials, IP address, and other info.

Request body: Includes HTML code, PHP, JS, Media, POST forms, etc..

Common attacks: Directory brute forcing, malicious objects, etc.

Introduction/Review

Common Linux utilities

- file, strings, and other basic Linux commands
- Installing packages/tools
- Basic info about networking protocols
- Programing/scripting language (Python)

GOOGLING + COMMON SENSE ☺

Forensics

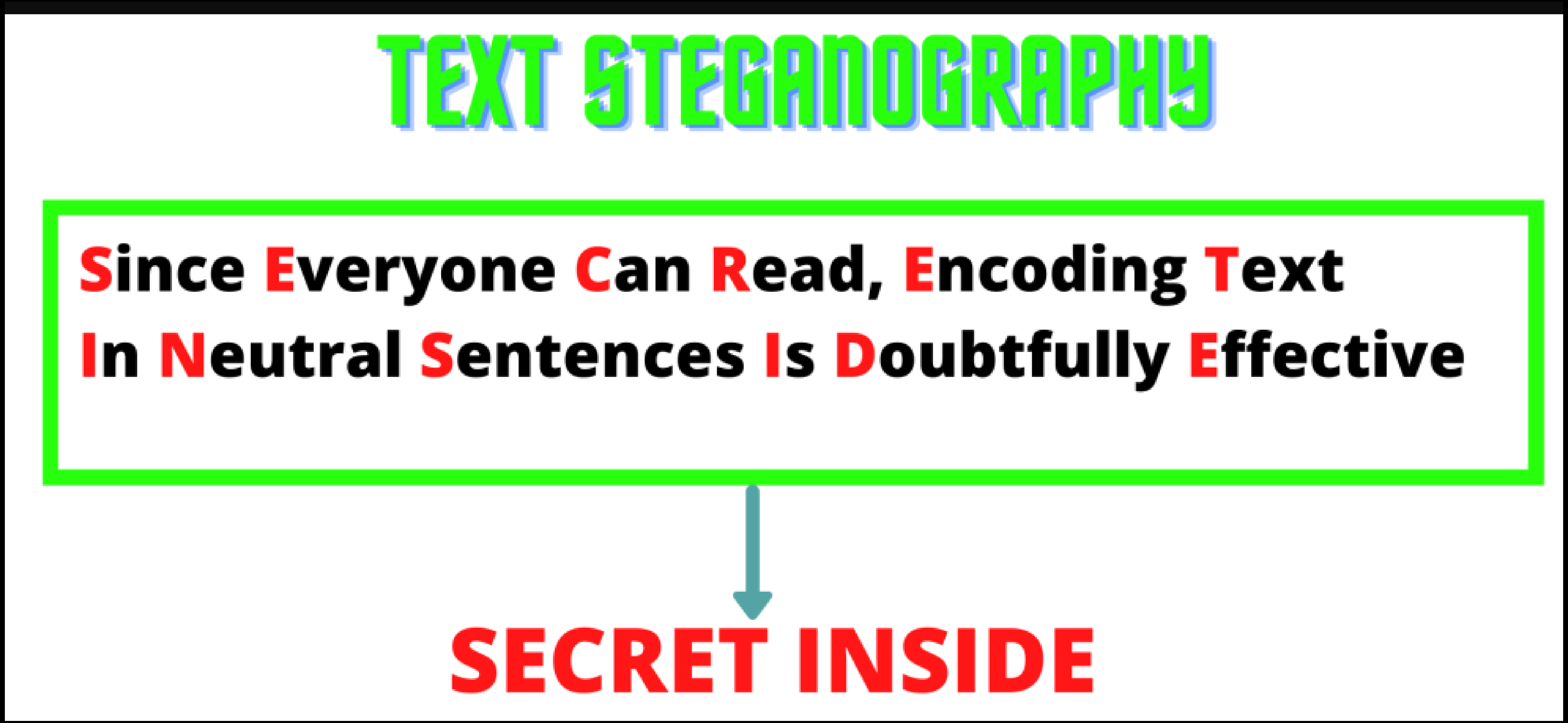
Digital forensics refers to investigating digital evidence created on a machine.

In the real cyber security applications, it’s mostly focused on analyzing logs, memory dumps, and disk images from real computers.

In CTFs, it’s focused on steganography, memory and disk analysis, sometimes android analysis, and even some reverse engineering.

Steganography

Steganography refers to hiding a message inside another message.

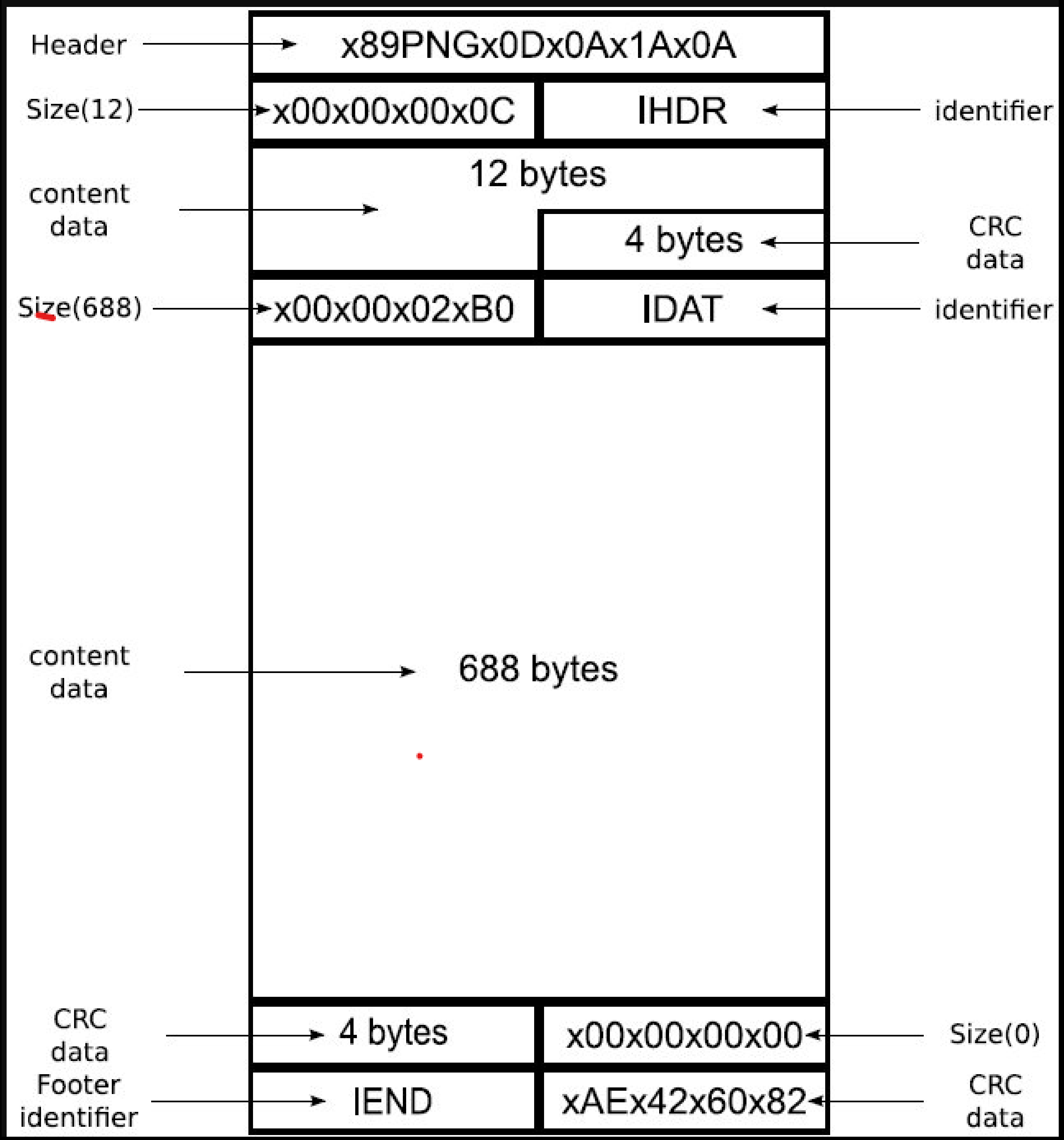


TEXT STEGANOGRAPHY

Since **E**veryone **C**an **R**ead, **E**ncoding **T**ext
In **N**eutral **S**entences **I**s **D**oubtfully **E**ffective

SECRET INSIDE

Steganography



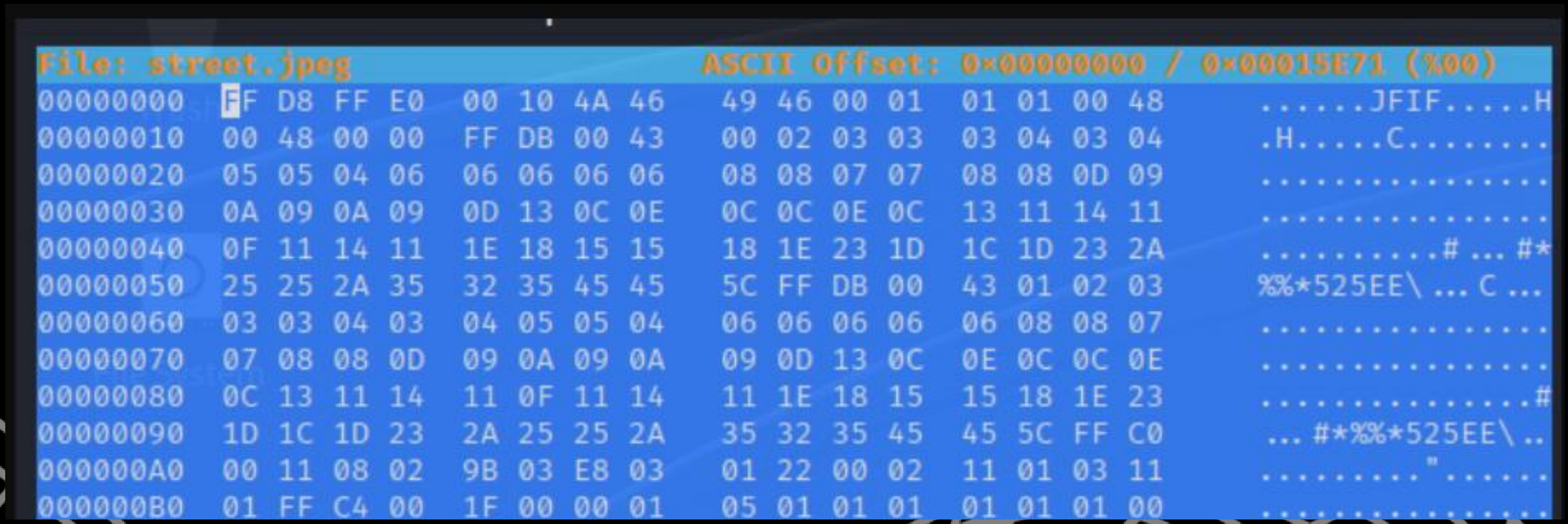
How are files structured?

Steganography

A secret message could be hidden in different types of files. Let’s see how files work and how we could identify them.

First of all, almost all files have something called signature. A signature is a sequence of bytes at the beginning of the file.

We can see those bytes using a hex editor.



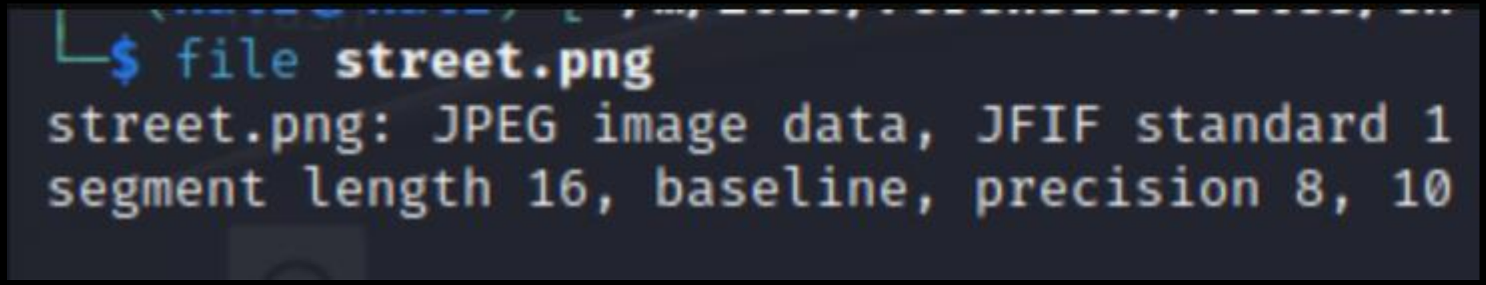
```
File: street.jpeg                                ASCII Offset: 0×00000000 / 0×00015E71 (%00)
00000000  FF D8 FF E0  00 10 4A 46  49 46 00 01  01 01 00 48   ......JFIF.....H
00000010  00 48 00 00  FF DB 00 43  00 02 03 03  03 04 03 04   .H.....C........
00000020  05 05 04 06  06 06 06 06  08 08 07 07  08 08 0D 09   ................
00000030  0A 09 0A 09  0D 13 0C 0E  0C 0C 0E 0C  13 11 14 11   ................
00000040  0F 11 14 11  1E 18 15 15  18 1E 23 1D  1C 1D 23 2A   ..........#...#*
00000050  25 25 2A 35  32 35 45 45  5C FF DB 00  43 01 02 03   %%*525EE\...C...
00000060  03 03 04 03  04 05 05 04  06 06 06 06  06 08 08 07   ................
00000070  07 08 08 0D  09 0A 09 0A  09 0D 13 0C  0E 0C 0C 0E   ................
00000080  0C 13 11 14  11 0F 11 14  11 1E 18 15  15 18 1E 23   ...............#
00000090  1D 1C 1D 23  2A 25 25 2A  35 32 35 45  45 5C FF C0   ...#*%%*525EE\..
000000A0  00 11 08 02  9B 03 E8 03  01 22 00 02  11 01 03 11   ........."......
000000B0  01 FF C4 00  1F 00 00 01  05 01 01 01  01 01 01 00   ................
```

Steganography

Getting the file signature will help us get what type of file it is. However, the simplest way to know the type of file is by looking at its extension, but the extension isn't always correct.

The command `file` shows us the type of file based on the signature, if the signature isn't corrupt.

Notice how the command says "JPEG" even if the extension is ".png".



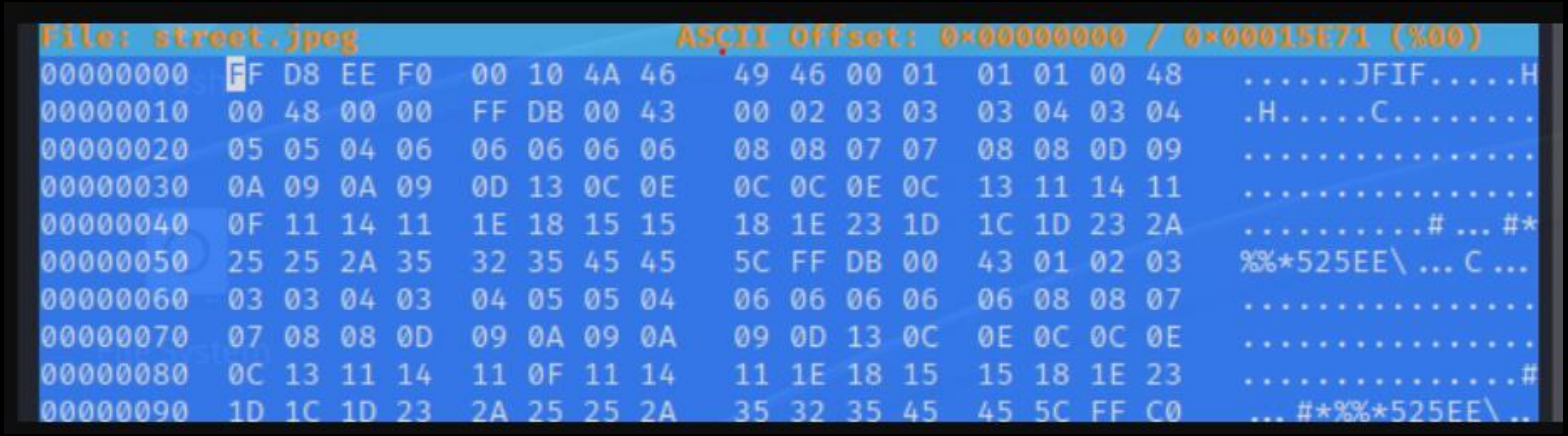
```
└─$ file street.png
street.png: JPEG image data, JFIF standard 1
segment length 16, baseline, precision 8, 10
```

Steganography

We know the file signature, then what?
- Correctly identifying files
- Fixing corrupted files

https://en.wikipedia.org/wiki/List_of_file_signatures

The signature was modified at the third byte. The original signature starts with FF D8 FF E0, not FF D8 EE F0.



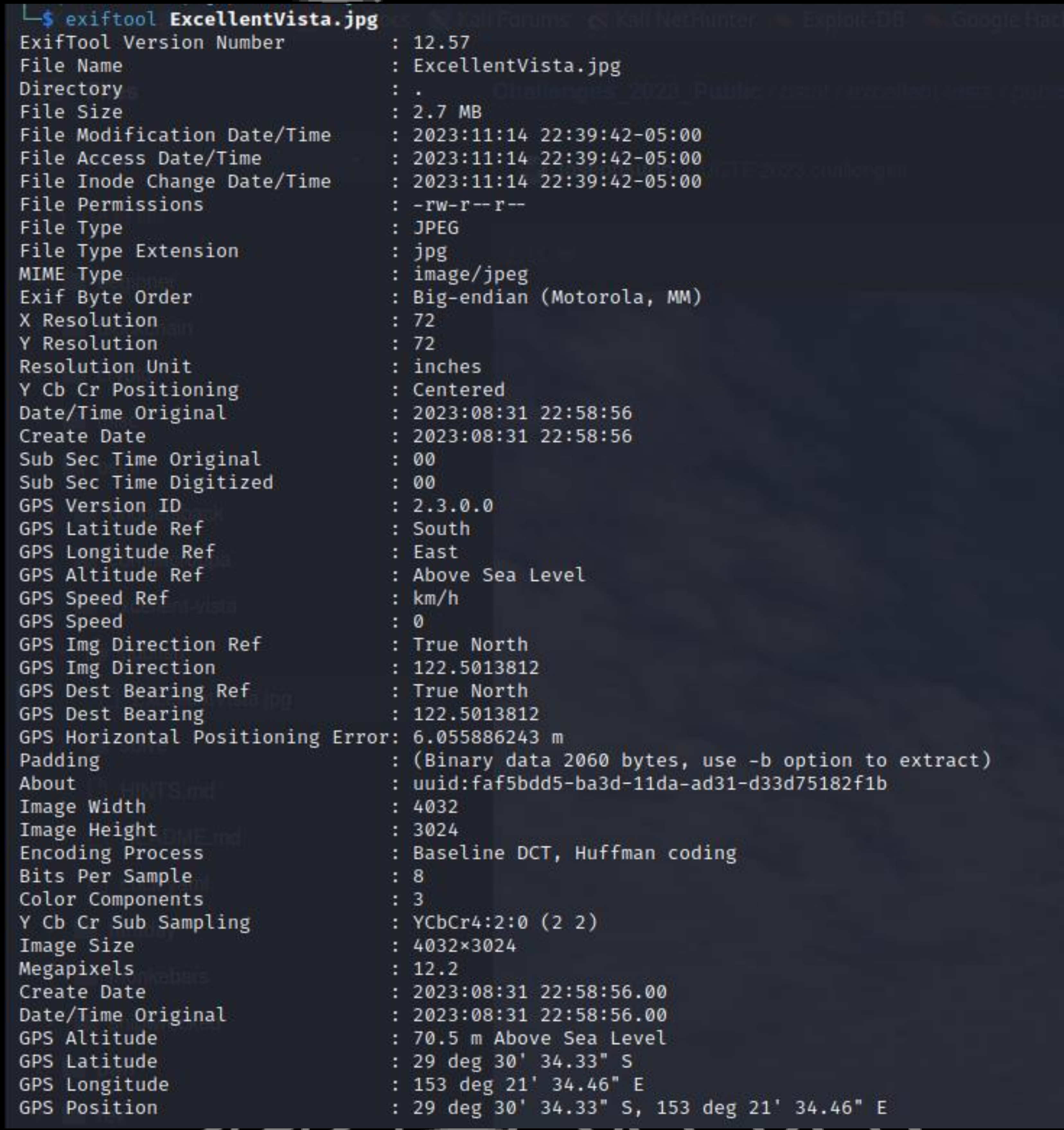
```
File: street.jpeg                          ASCII Offset: 0x00000000 / 0x00015E71 (%00)
00000000  FF D8 EE F0  00 10 4A 46  49 46 00 01  01 01 00 48   ......JFIF.....H
00000010  00 48 00 00  FF DB 00 43  00 02 03 03  03 04 03 04   .H.....C........
00000020  05 05 04 06  06 06 06 06  08 08 07 07  08 08 0D 09   ................
00000030  0A 09 0A 09  0D 13 0C 0E  0C 0C 0E 0C  13 11 14 11   ................
00000040  0F 11 14 11  1E 18 15 15  18 1E 23 1D  1C 1D 23 2A   ..........#...#*
00000050  25 25 2A 35  32 35 45 45  5C FF DB 00  43 01 02 03   %%*525EE\...C...
00000060  03 03 04 03  04 05 05 04  06 06 06 06  06 08 08 07   ................
00000070  07 08 08 0D  09 0A 09 0A  09 0D 13 0C  0E 0C 0C 0E   ................
00000080  0C 13 11 14  11 0F 11 14  11 1E 18 15  15 18 1E 23   ...............#
00000090  1D 1C 1D 23  2A 25 25 2A  35 32 35 45  45 5C FF C0   ...#*%%*525EE\..
```

Metadata

Metadata is data about data. Metadata is usually attached with photos, videos, documents, etc. This could help us get the flag sometimes, or at least some hints about what we're dealing with.

Example: We can see this image with its coordinates.

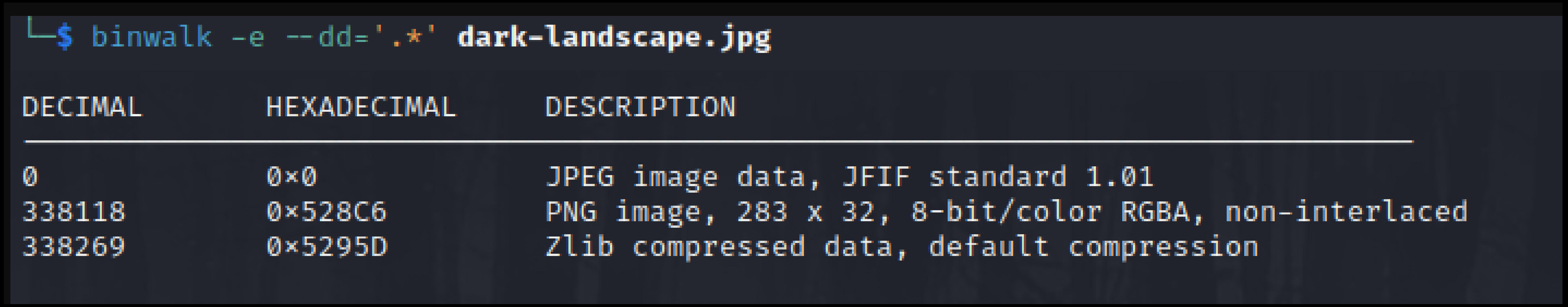
Which city was this image taken in?



```
└─$ exiftool ExcellentVista.jpg
ExifTool Version Number         : 12.57
File Name                       : ExcellentVista.jpg
Directory                       : .
File Size                       : 2.7 MB
File Modification Date/Time     : 2023:11:14 22:39:42-05:00
File Access Date/Time           : 2023:11:14 22:39:42-05:00
File Inode Change Date/Time     : 2023:11:14 22:39:42-05:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
Exif Byte Order                 : Big-endian (Motorola, MM)
X Resolution                    : 72
Y Resolution                    : 72
Resolution Unit                 : inches
Y Cb Cr Positioning             : Centered
Date/Time Original              : 2023:08:31 22:58:56
Create Date                     : 2023:08:31 22:58:56
Sub Sec Time Original           : 00
Sub Sec Time Digitized          : 00
GPS Version ID                  : 2.3.0.0
GPS Latitude Ref                : South
GPS Longitude Ref               : East
GPS Altitude Ref                : Above Sea Level
GPS Speed Ref                   : km/h
GPS Speed                       : 0
GPS Img Direction Ref           : True North
GPS Img Direction               : 122.5013812
GPS Dest Bearing Ref            : True North
GPS Dest Bearing                : 122.5013812
GPS Horizontal Positioning Error: 6.055886243 m
Padding                         : (Binary data 2060 bytes, use -b option to extract)
About                           : uuid:faf5bdd5-ba3d-11da-ad31-d33d75182f1b
Image Width                     : 4032
Image Height                    : 3024
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 4032x3024
Megapixels                      : 12.2
Create Date                     : 2023:08:31 22:58:56.00
Date/Time Original              : 2023:08:31 22:58:56.00
GPS Altitude                    : 70.5 m Above Sea Level
GPS Latitude                    : 29 deg 30' 34.33" S
GPS Longitude                   : 153 deg 21' 34.46" E
GPS Position                    : 29 deg 30' 34.33" S, 153 deg 21' 34.46" E
```

File carving/extracting

Sometimes some files are hidden inside other files. We can use the tool foremost (or binwalk if you want) to extract those files.

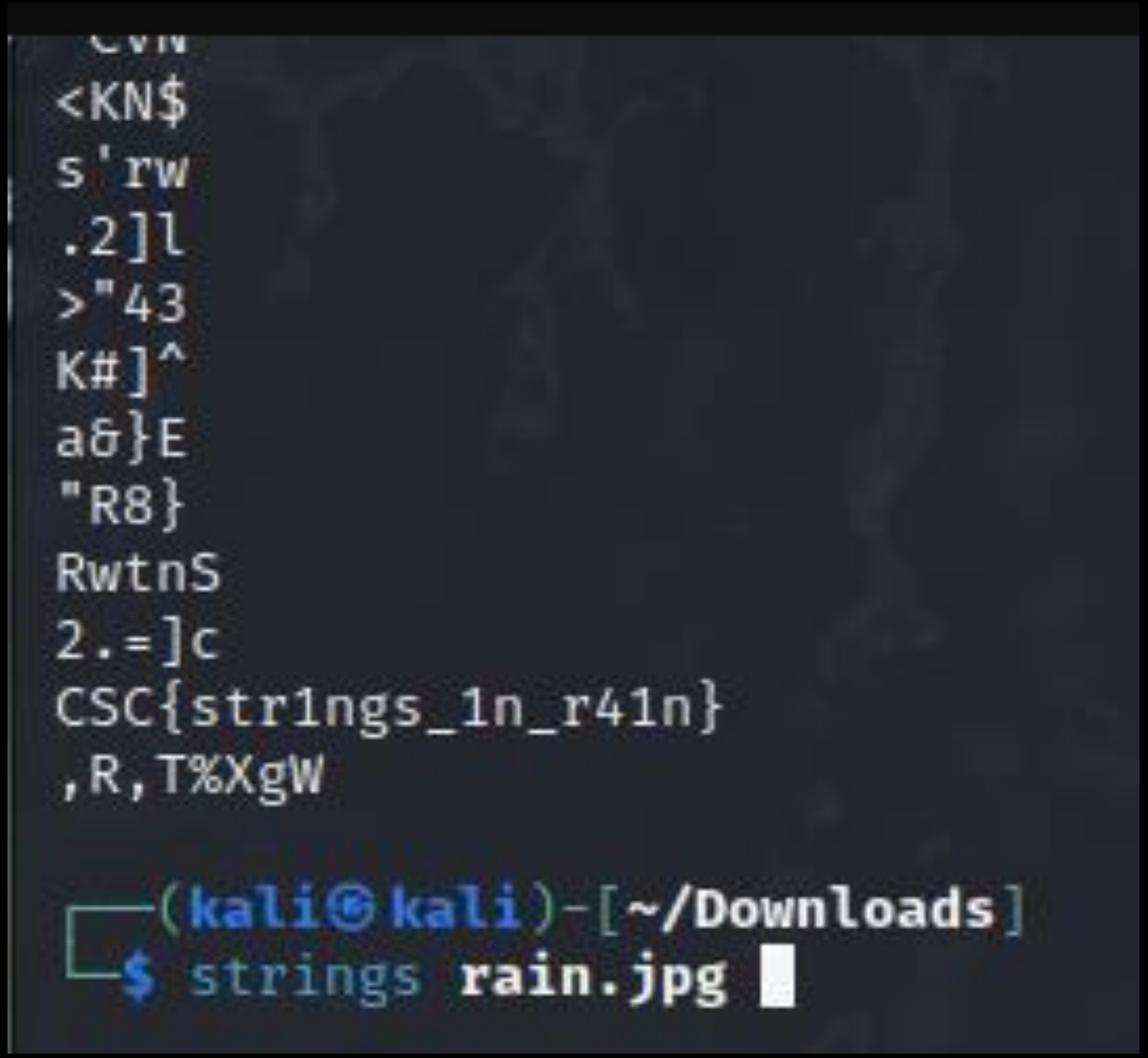


```
└─$ binwalk -e --dd='.*' dark-landscape.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
338118        0x528C6         PNG image, 283 x 32, 8-bit/color RGBA, non-interlaced
338269        0x5295D         Zlib compressed data, default compression
```

strings

Printable text inside the image/binary

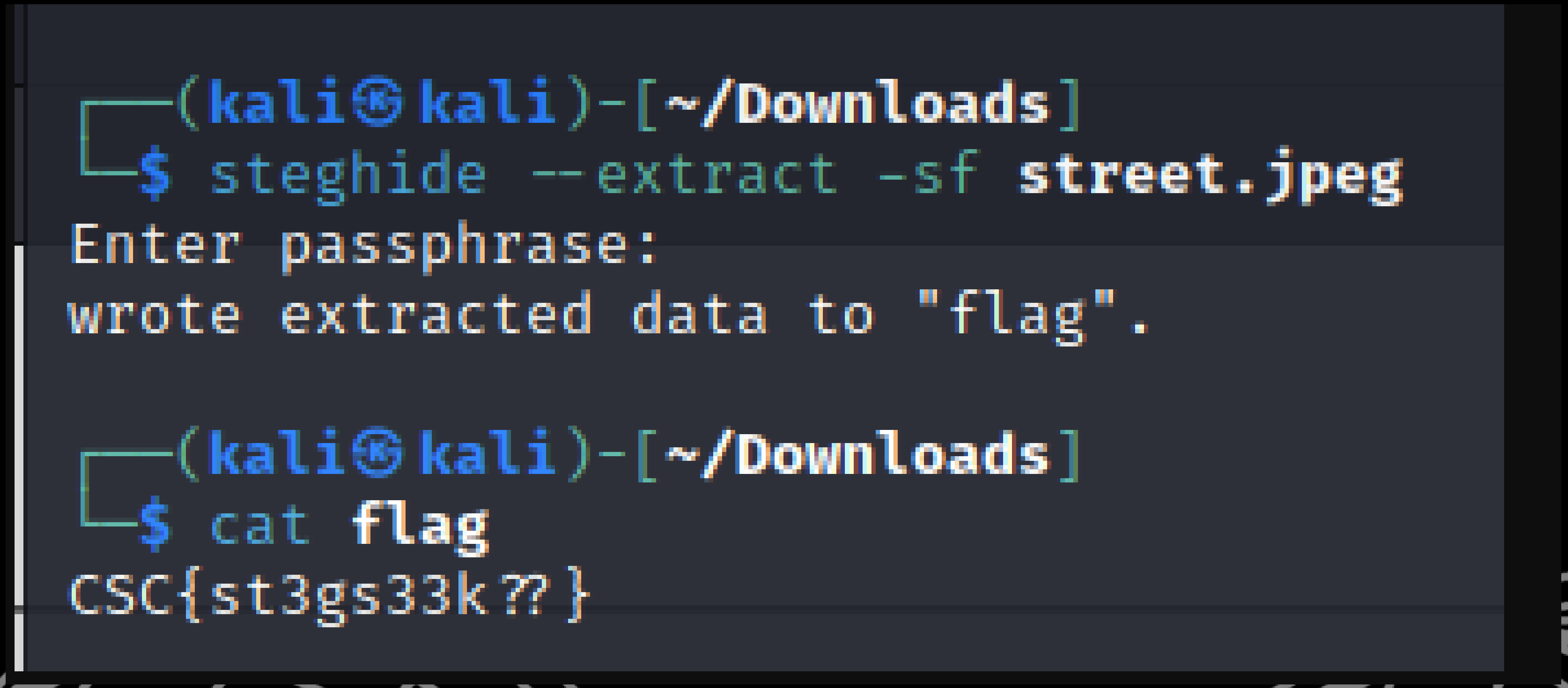


```
CVN
<KN$
s'rw
.2]l
>"43
K#]^
a&}E
"R8}
RwtnS
2.=]c
CSC{str1ngs_1n_r41n}
,R,T%XgW

┌──(kali㉿kali)-[~/Downloads]
└─$ strings rain.jpg
```

steghide

This is a tool that embeds (hides) files inside images but encrypts the file with a passphrase.

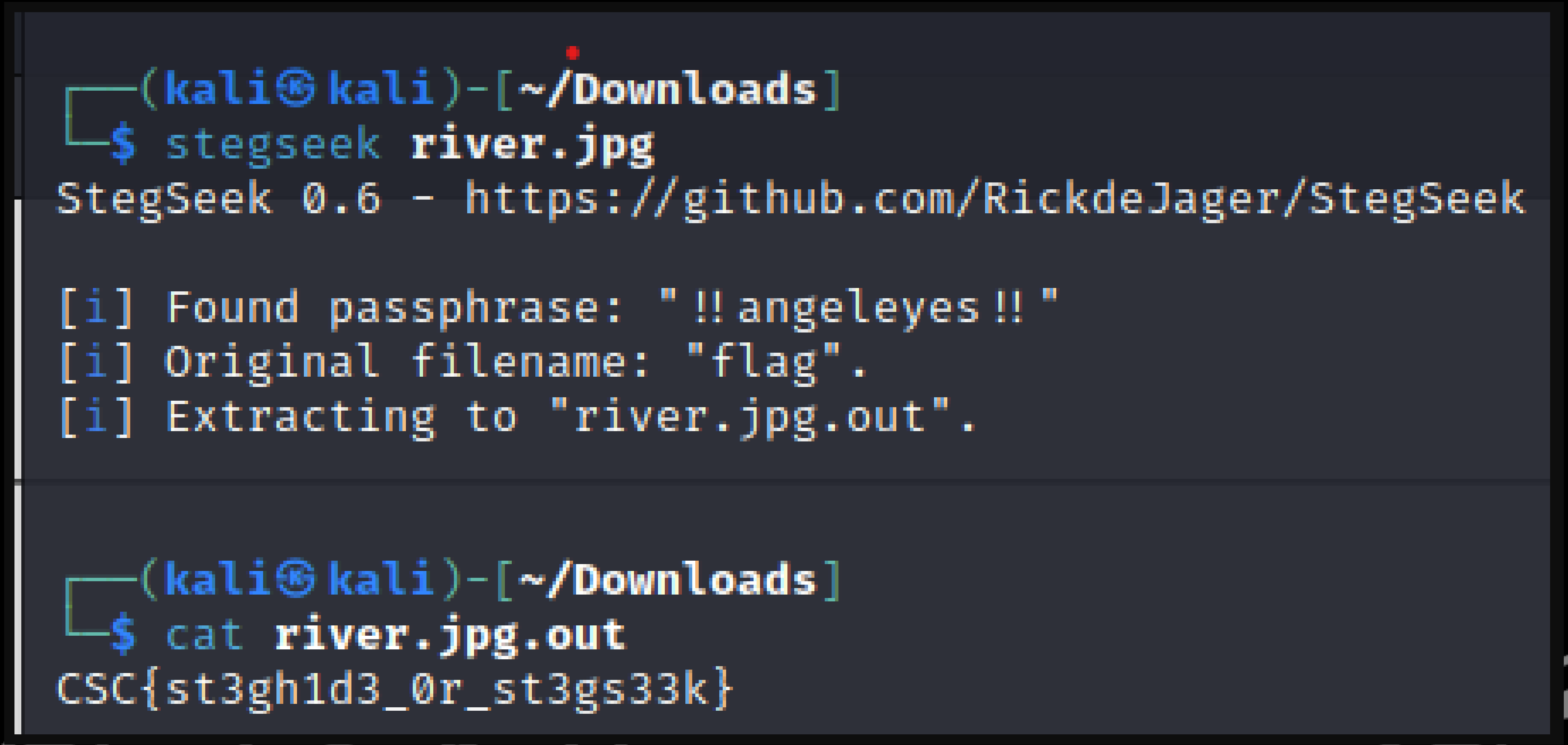


```
┌──(kali㉿kali)-[~/Downloads]
└─$ steghide --extract -sf street.jpeg
Enter passphrase:
wrote extracted data to "flag".

┌──(kali㉿kali)-[~/Downloads]
└─$ cat flag
CSC{st3gs33k??}
```

stegseek

This tool brute forces the passphrase to extract the files out of an image that have been hidden using steghide.

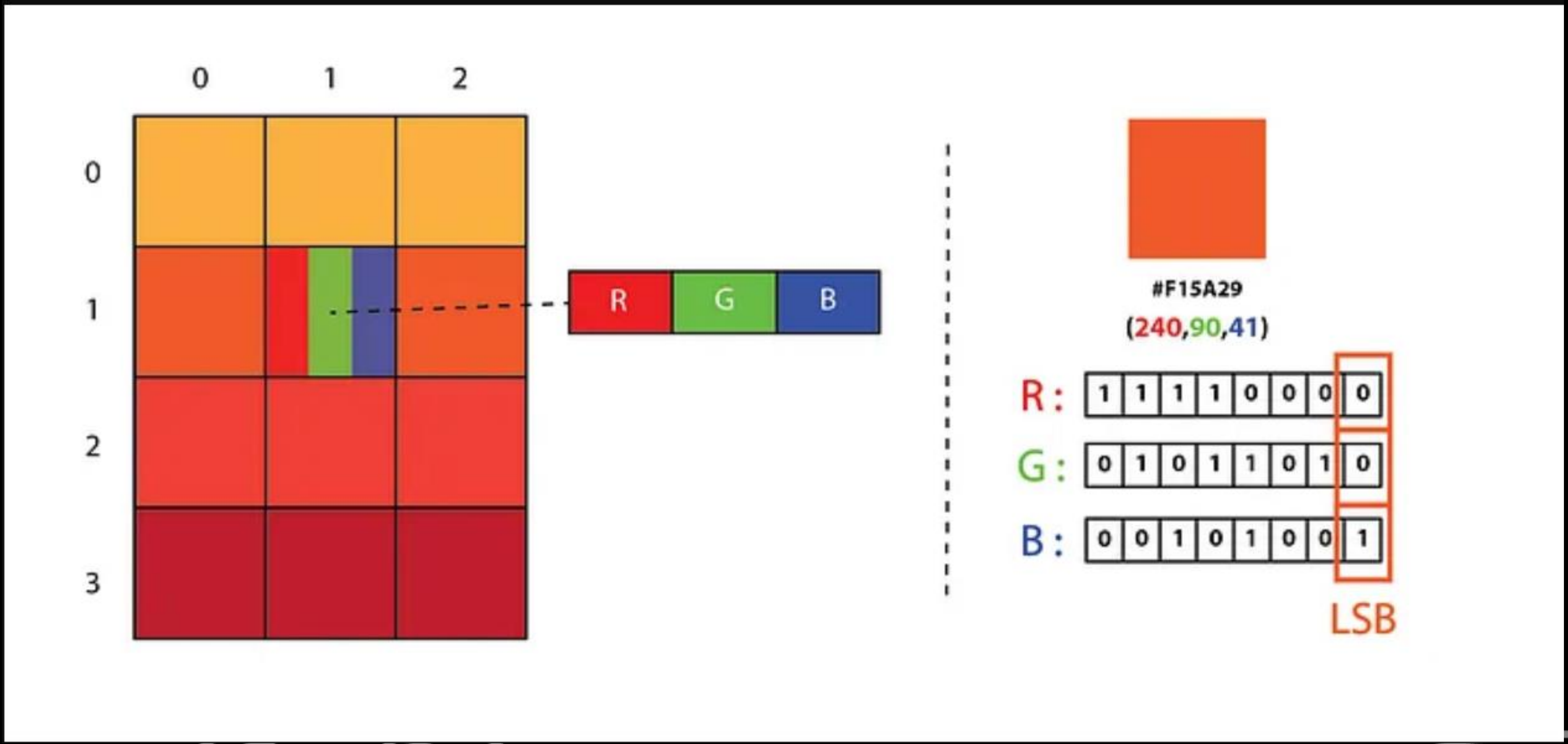


```
┌──(kali㉿kali)-[~/Downloads]
└─$ stegseek river.jpg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "!!angeleyes!!"
[i] Original filename: "flag".
[i] Extracting to "river.jpg.out".

┌──(kali㉿kali)-[~/Downloads]
└─$ cat river.jpg.out
CSC{st3gh1d3_0r_st3gs33k}
```

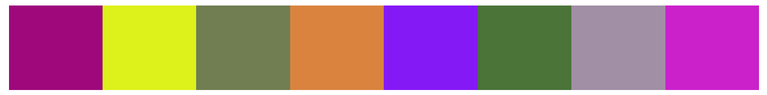
LSB (Least Significant Bit) steganography



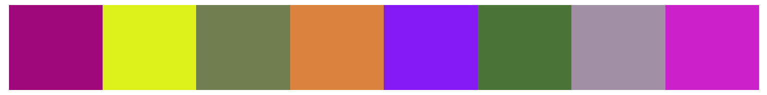
LSB (Least Significant Bit) steganography

Our message = 01000011 01010011 01000011
C S C

Pixels values (before embedding):
10011110 00001000 01111011 11011101 11110001 00011010 01110000 01111110 01010001 11011010
10000011 00111111 10000101 00011001 11110110 01001010 01110100 00111000 10100001 10001111
10100101 11001010 00100001 11001010



Pixels values (after embedding):
10011110 00001001 01111010 11011100 11110000 00011010 01110001 01111111 01010000 11011011
10000010 00111111 10000100 00011000 11110111 01001011 01110100 00111001 10100000 10001110
10100100 11001010 00100001 11001011



Threat hunting and threat intelligence

Threat hunting is the process of finding undetected attacks in the network that managed to get into the network without being detected.

Threat intelligence is data collected about malware or any threat actor that can help in detecting malicious activity in the future.

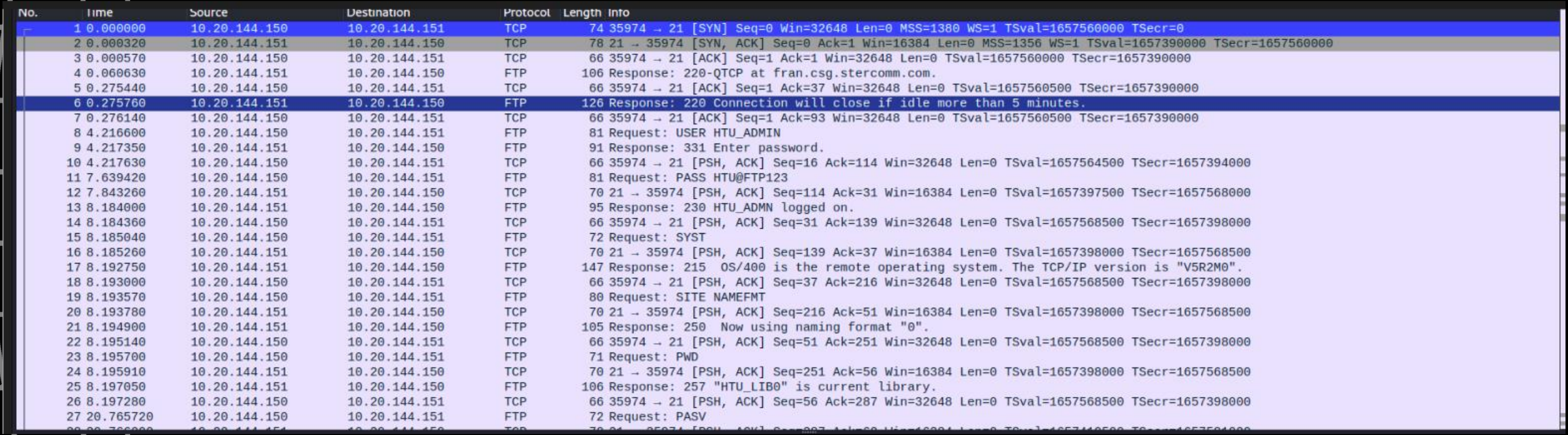
Network analysis

PCAP file is s a network capture of a traffic for a period of time.

The traffic includes many protocols, including encrypted and plaintext traffic.

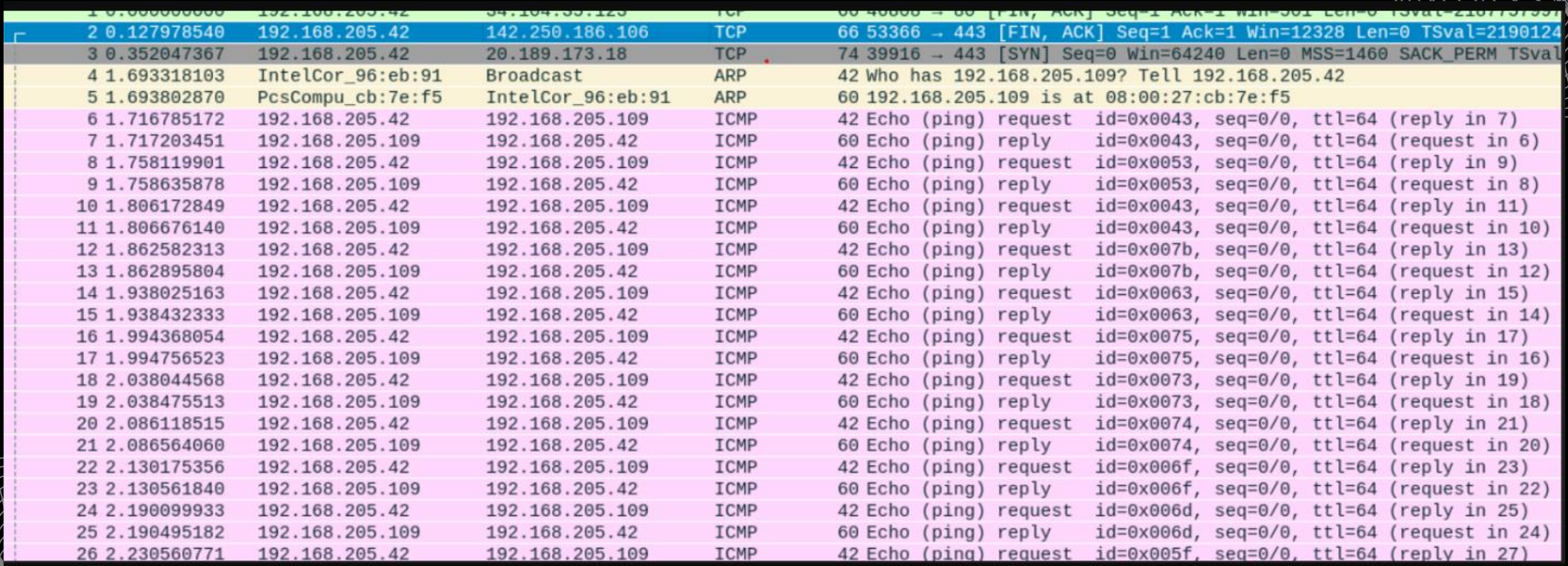
We can inspect the traffic using Wireshark

Wireshark example



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.20.144.150	10.20.144.151	TCP	74	35974 → 21 [SYN] Seq=0 Win=32648 Len=0 MSS=1380 WS=1 TSval=1657560000 TSecr=0
2	0.000320	10.20.144.151	10.20.144.150	TCP	78	21 → 35974 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1356 WS=1 TSval=1657390000 TSecr=1657560000
3	0.000570	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [ACK] Seq=1 Ack=1 Win=32648 Len=0 TSval=1657560000 TSecr=1657390000
4	0.060630	10.20.144.151	10.20.144.150	FTP	106	Response: 220-QTCP at fran.csg.stercomm.com.
5	0.275440	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [ACK] Seq=1 Ack=37 Win=32648 Len=0 TSval=1657560500 TSecr=1657390000
6	0.275760	10.20.144.151	10.20.144.150	FTP	126	Response: 220 Connection will close if idle more than 5 minutes.
7	0.276140	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [ACK] Seq=1 Ack=93 Win=32648 Len=0 TSval=1657560500 TSecr=1657390000
8	4.216600	10.20.144.150	10.20.144.151	FTP	81	Request: USER HTU_ADMIN
9	4.217350	10.20.144.151	10.20.144.150	FTP	91	Response: 331 Enter password.
10	4.217630	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [PSH, ACK] Seq=16 Ack=114 Win=32648 Len=0 TSval=1657564500 TSecr=1657394000
11	7.639420	10.20.144.150	10.20.144.151	FTP	81	Request: PASS HTU@FTP123
12	7.843260	10.20.144.151	10.20.144.150	TCP	70	21 → 35974 [PSH, ACK] Seq=114 Ack=31 Win=16384 Len=0 TSval=1657397500 TSecr=1657568000
13	8.184000	10.20.144.151	10.20.144.150	FTP	95	Response: 230 HTU_ADMN logged on.
14	8.184360	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [PSH, ACK] Seq=31 Ack=139 Win=32648 Len=0 TSval=1657568500 TSecr=1657398000
15	8.185040	10.20.144.150	10.20.144.151	FTP	72	Request: SYST
16	8.185260	10.20.144.151	10.20.144.150	TCP	70	21 → 35974 [PSH, ACK] Seq=139 Ack=37 Win=16384 Len=0 TSval=1657398000 TSecr=1657568500
17	8.192750	10.20.144.151	10.20.144.150	FTP	147	Response: 215 OS/400 is the remote operating system. The TCP/IP version is "V5R2M0".
18	8.193000	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [PSH, ACK] Seq=37 Ack=216 Win=32648 Len=0 TSval=1657568500 TSecr=1657398000
19	8.193570	10.20.144.150	10.20.144.151	FTP	80	Request: SITE NAMEFMT
20	8.193780	10.20.144.151	10.20.144.150	TCP	70	21 → 35974 [PSH, ACK] Seq=216 Ack=51 Win=16384 Len=0 TSval=1657398000 TSecr=1657568500
21	8.194900	10.20.144.151	10.20.144.150	FTP	105	Response: 250 Now using naming format "0".
22	8.195140	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [PSH, ACK] Seq=51 Ack=251 Win=32648 Len=0 TSval=1657568500 TSecr=1657398000
23	8.195700	10.20.144.150	10.20.144.151	FTP	71	Request: PWD
24	8.195910	10.20.144.151	10.20.144.150	TCP	70	21 → 35974 [PSH, ACK] Seq=251 Ack=56 Win=16384 Len=0 TSval=1657398000 TSecr=1657568500
25	8.197050	10.20.144.151	10.20.144.150	FTP	106	Response: 257 "HTU_LIB0" is current library.
26	8.197280	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [PSH, ACK] Seq=56 Ack=287 Win=32648 Len=0 TSval=1657568500 TSecr=1657398000
27	20.765720	10.20.144.150	10.20.144.151	FTP	72	Request: PASV

Wireshark example



```
  1 0.000000000   192.168.205.42      34.104.35.123       TCP       66 40808 → 80 [FIN, ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2187
  2 0.127978540   192.168.205.42      142.250.186.106     TCP       66 53366 → 443 [FIN, ACK] Seq=1 Ack=1 Win=12328 Len=0 TSval=2190124
  3 0.352047367   192.168.205.42      20.189.173.18       TCP       74 39916 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval
  4 1.693318103   IntelCor_96:eb:91   Broadcast           ARP       42 Who has 192.168.205.109? Tell 192.168.205.42
  5 1.693802870   PcsCompu_cb:7e:f5   IntelCor_96:eb:91   ARP       60 192.168.205.109 is at 08:00:27:cb:7e:f5
  6 1.716785172   192.168.205.42      192.168.205.109     ICMP      42 Echo (ping) request  id=0x0043, seq=0/0, ttl=64 (reply in 7)
  7 1.717203451   192.168.205.109     192.168.205.42      ICMP      60 Echo (ping) reply    id=0x0043, seq=0/0, ttl=64 (request in 6)
  8 1.758119901   192.168.205.42      192.168.205.109     ICMP      42 Echo (ping) request  id=0x0053, seq=0/0, ttl=64 (reply in 9)
  9 1.758635878   192.168.205.109     192.168.205.42      ICMP      60 Echo (ping) reply    id=0x0053, seq=0/0, ttl=64 (request in 8)
 10 1.806172849   192.168.205.42      192.168.205.109     ICMP      42 Echo (ping) request  id=0x0043, seq=0/0, ttl=64 (reply in 11)
 11 1.806676140   192.168.205.109     192.168.205.42      ICMP      60 Echo (ping) reply    id=0x0043, seq=0/0, ttl=64 (request in 10)
 12 1.862582313   192.168.205.42      192.168.205.109     ICMP      42 Echo (ping) request  id=0x007b, seq=0/0, ttl=64 (reply in 13)
 13 1.862895804   192.168.205.109     192.168.205.42      ICMP      60 Echo (ping) reply    id=0x007b, seq=0/0, ttl=64 (request in 12)
 14 1.938025163   192.168.205.42      192.168.205.109     ICMP      42 Echo (ping) request  id=0x0063, seq=0/0, ttl=64 (reply in 15)
 15 1.938432333   192.168.205.109     192.168.205.42      ICMP      60 Echo (ping) reply    id=0x0063, seq=0/0, ttl=64 (request in 14)
 16 1.994368054   192.168.205.42      192.168.205.109     ICMP      42 Echo (ping) request  id=0x0075, seq=0/0, ttl=64 (reply in 17)
 17 1.994756523   192.168.205.109     192.168.205.42      ICMP      60 Echo (ping) reply    id=0x0075, seq=0/0, ttl=64 (request in 16)
 18 2.038044568   192.168.205.42      192.168.205.109     ICMP      42 Echo (ping) request  id=0x0073, seq=0/0, ttl=64 (reply in 19)
 19 2.038475513   192.168.205.109     192.168.205.42      ICMP      60 Echo (ping) reply    id=0x0073, seq=0/0, ttl=64 (request in 18)
 20 2.086118515   192.168.205.42      192.168.205.109     ICMP      42 Echo (ping) request  id=0x0074, seq=0/0, ttl=64 (reply in 21)
 21 2.086564060   192.168.205.109     192.168.205.42      ICMP      60 Echo (ping) reply    id=0x0074, seq=0/0, ttl=64 (request in 20)
 22 2.130175356   192.168.205.42      192.168.205.109     ICMP      42 Echo (ping) request  id=0x006f, seq=0/0, ttl=64 (reply in 23)
 23 2.130561840   192.168.205.109     192.168.205.42      ICMP      60 Echo (ping) reply    id=0x006f, seq=0/0, ttl=64 (request in 22)
 24 2.190099933   192.168.205.42      192.168.205.109     ICMP      42 Echo (ping) request  id=0x006d, seq=0/0, ttl=64 (reply in 25)
 25 2.190495182   192.168.205.109     192.168.205.42      ICMP      60 Echo (ping) reply    id=0x006d, seq=0/0, ttl=64 (request in 24)
 26 2.230560771   192.168.205.42      192.168.205.109     ICMP      42 Echo (ping) request  id=0x005f, seq=0/0, ttl=64 (reply in 27)
```

Memory analysis

When an incident happens, let’s say a malware infected a machine, forensic experts gain a memory dump of the infected machine to analyze what happened.

A memory dump is an exact copy of the RAM.

Memory analysis

What does RAM include?

It includes anything that is currently being processed in the machine. Which includes:

- Processes
- Open files
- Open applications
- Registry keys
- OS information
- other stuff

Volatility

This is the tool that we’ll be using for memory forensics.

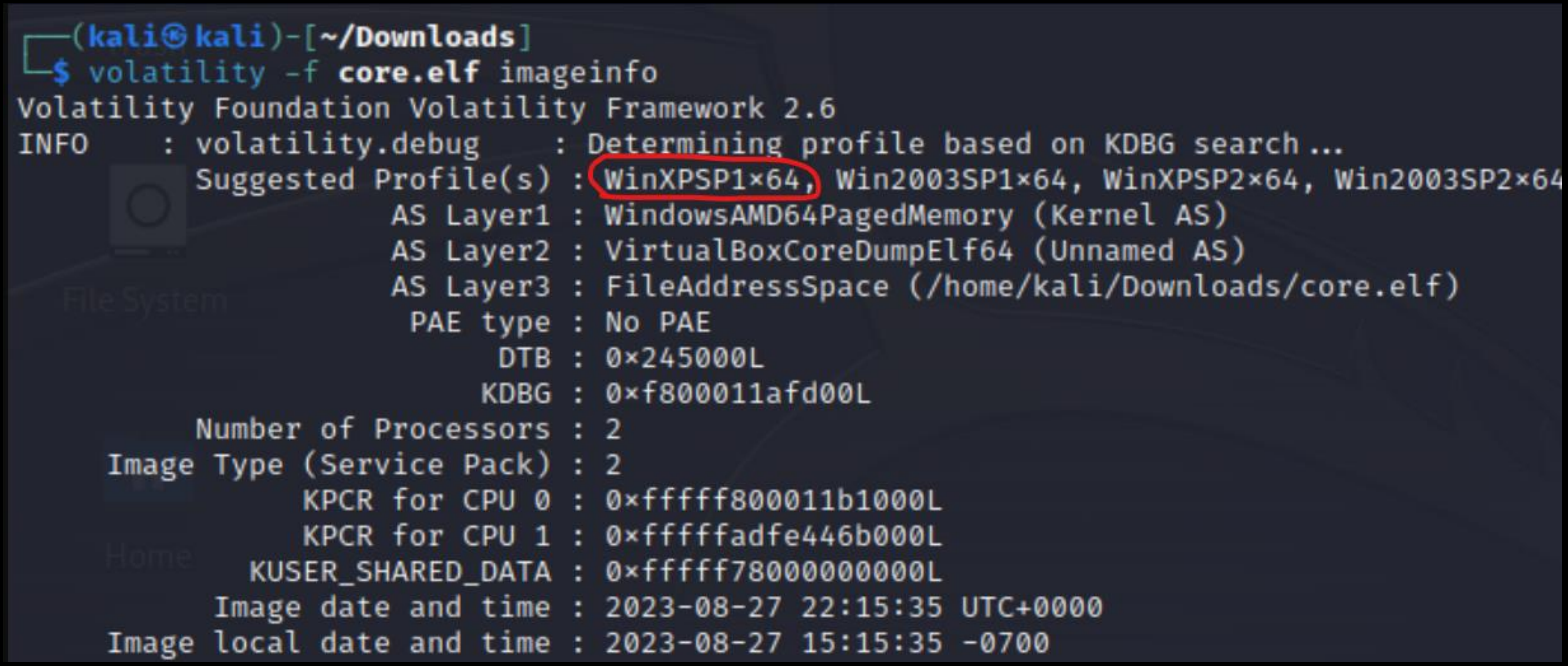
Those are the general steps for memory forensics:

- Identify the image profile (which operating system, version)
- Inspect processes (look for suspicious processes)
- Inspect files
- Check connections

Image profile

We need the profile to tell volatility how to deal with the memory dump.

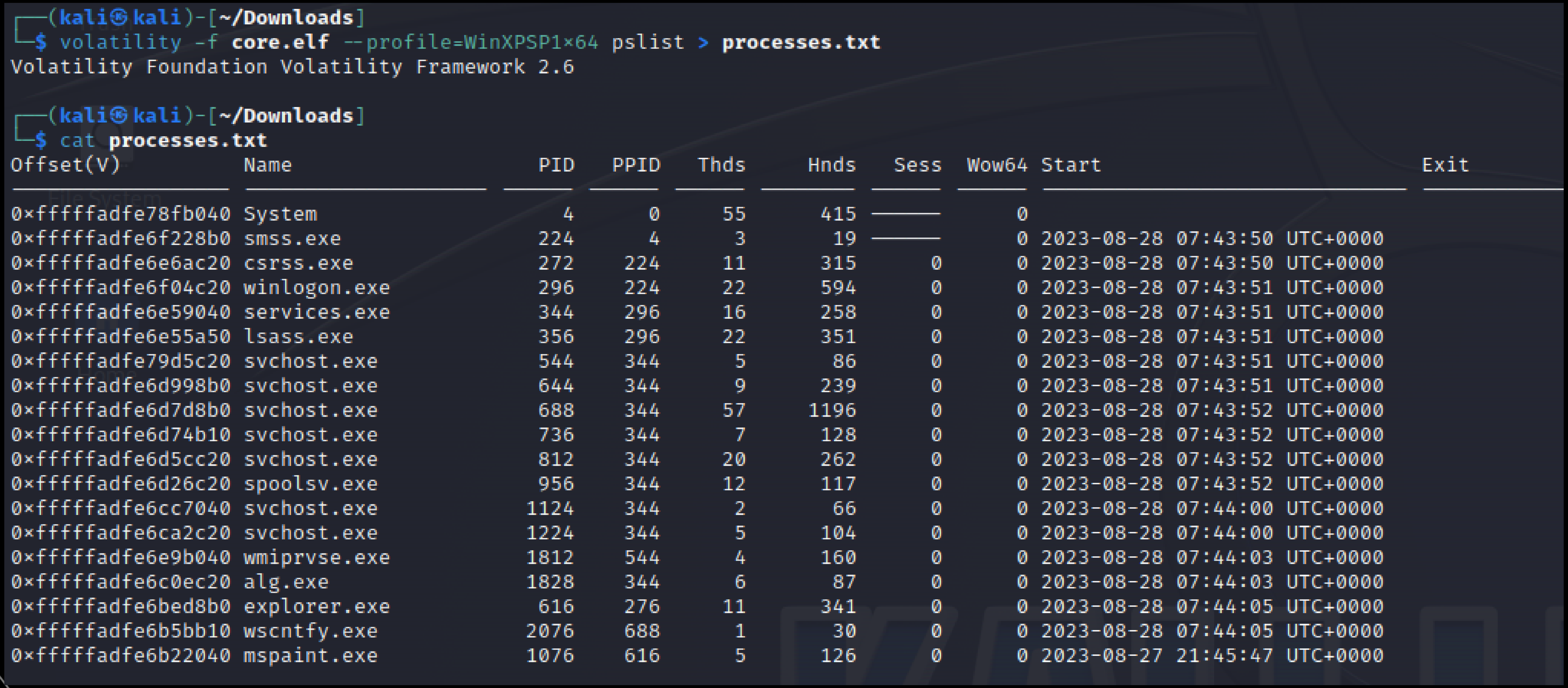
Syntax:
volatility -f <image_file> imageinfo



```
┌──(kali㉿kali)-[~/Downloads]
└─$ volatility -f core.elf imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP1x64, Win2003SP1x64, WinXPSP2x64, Win2003SP2x64
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : VirtualBoxCoreDumpElf64 (Unnamed AS)
                     AS Layer3 : FileAddressSpace (/home/kali/Downloads/core.elf)
                      PAE type : No PAE
                           DTB : 0x245000L
                          KDBG : 0xf800011afd00L
          Number of Processors : 2
     Image Type (Service Pack) : 2
                KPCR for CPU 0 : 0xfffff800011b1000L
                KPCR for CPU 1 : 0xfffffadfe446b000L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2023-08-27 22:15:35 UTC+0000
     Image local date and time : 2023-08-27 15:15:35 -0700
```

Processes list

Syntax:
volatility -f <image_file> --profile=<profile> pslist

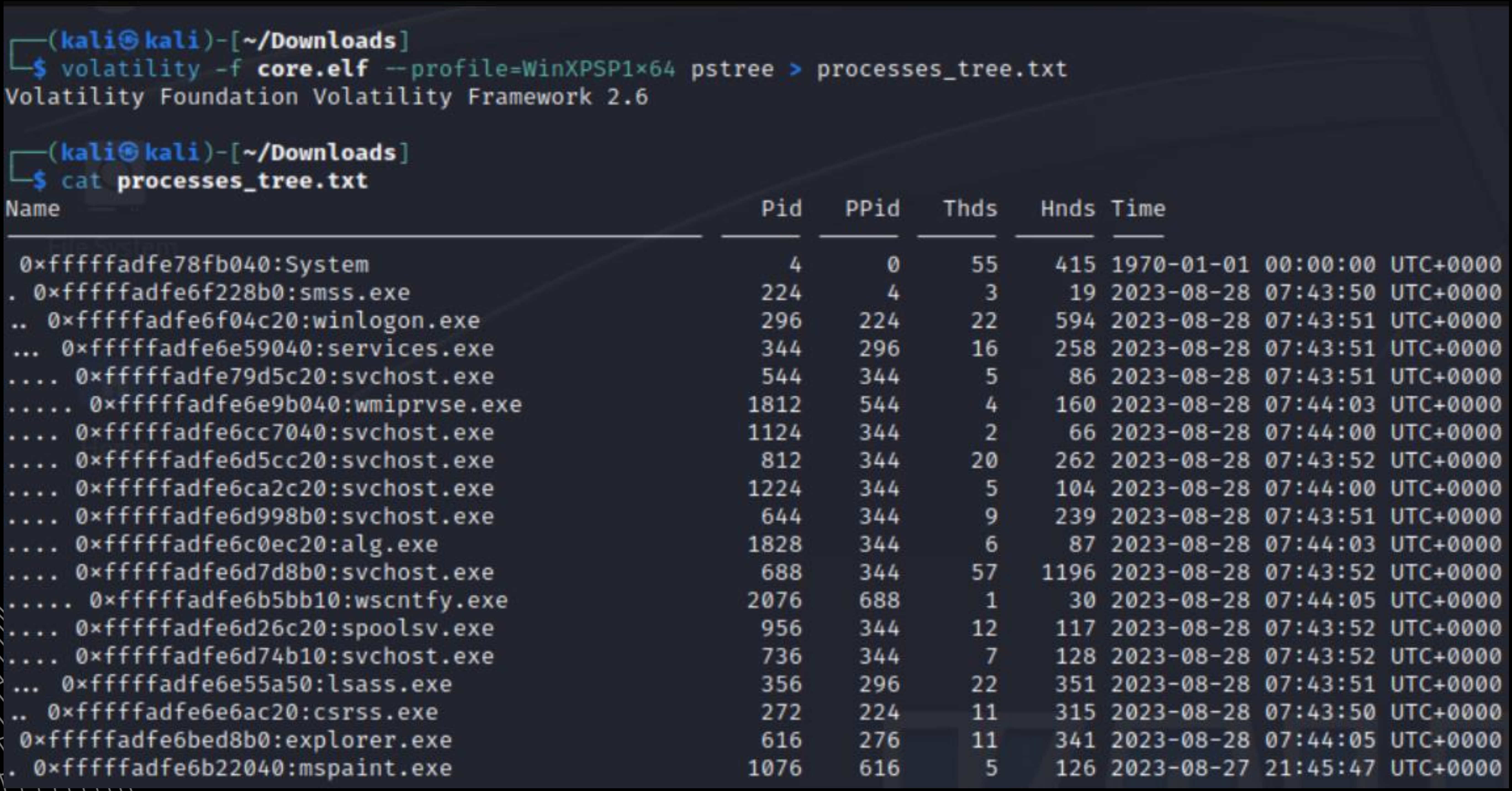


```
┌──(kali㉿kali)-[~/Downloads]
└─$ volatility -f core.elf --profile=WinXPSP1x64 pslist > processes.txt
Volatility Foundation Volatility Framework 2.6

┌──(kali㉿kali)-[~/Downloads]
└─$ cat processes.txt
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
────────────────── ──────────────────── ────── ────── ────── ──────── ────── ────── ────────────────────────────── ──────────────────────────────
0xfffffadfe78fb040 System                    4      0     55      415 ──────      0
0xfffffadfe6f228b0 smss.exe                224      4      3       19 ──────      0 2023-08-28 07:43:50 UTC+0000
0xfffffadfe6e6ac20 csrss.exe               272    224     11      315      0      0 2023-08-28 07:43:50 UTC+0000
0xfffffadfe6f04c20 winlogon.exe            296    224     22      594      0      0 2023-08-28 07:43:51 UTC+0000
0xfffffadfe6e59040 services.exe            344    296     16      258      0      0 2023-08-28 07:43:51 UTC+0000
0xfffffadfe6e55a50 lsass.exe               356    296     22      351      0      0 2023-08-28 07:43:51 UTC+0000
0xfffffadfe79d5c20 svchost.exe             544    344      5       86      0      0 2023-08-28 07:43:51 UTC+0000
0xfffffadfe6d998b0 svchost.exe             644    344      9      239      0      0 2023-08-28 07:43:51 UTC+0000
0xfffffadfe6d7d8b0 svchost.exe             688    344     57     1196      0      0 2023-08-28 07:43:52 UTC+0000
0xfffffadfe6d74b10 svchost.exe             736    344      7      128      0      0 2023-08-28 07:43:52 UTC+0000
0xfffffadfe6d5cc20 svchost.exe             812    344     20      262      0      0 2023-08-28 07:43:52 UTC+0000
0xfffffadfe6d26c20 spoolsv.exe             956    344     12      117      0      0 2023-08-28 07:43:52 UTC+0000
0xfffffadfe6cc7040 svchost.exe            1124    344      2       66      0      0 2023-08-28 07:44:00 UTC+0000
0xfffffadfe6ca2c20 svchost.exe            1224    344      5      104      0      0 2023-08-28 07:44:00 UTC+0000
0xfffffadfe6e9b040 wmiprvse.exe           1812    544      4      160      0      0 2023-08-28 07:44:03 UTC+0000
0xfffffadfe6c0ec20 alg.exe                1828    344      6       87      0      0 2023-08-28 07:44:03 UTC+0000
0xfffffadfe6bed8b0 explorer.exe            616    276     11      341      0      0 2023-08-28 07:44:05 UTC+0000
0xfffffadfe6b5bb10 wscntfy.exe            2076    688      1       30      0      0 2023-08-28 07:44:05 UTC+0000
0xfffffadfe6b22040 mspaint.exe            1076    616      5      126      0      0 2023-08-27 21:45:47 UTC+0000
```

Processes list

Syntax:
volatility -f <image_file> --profile=<profile> pstree

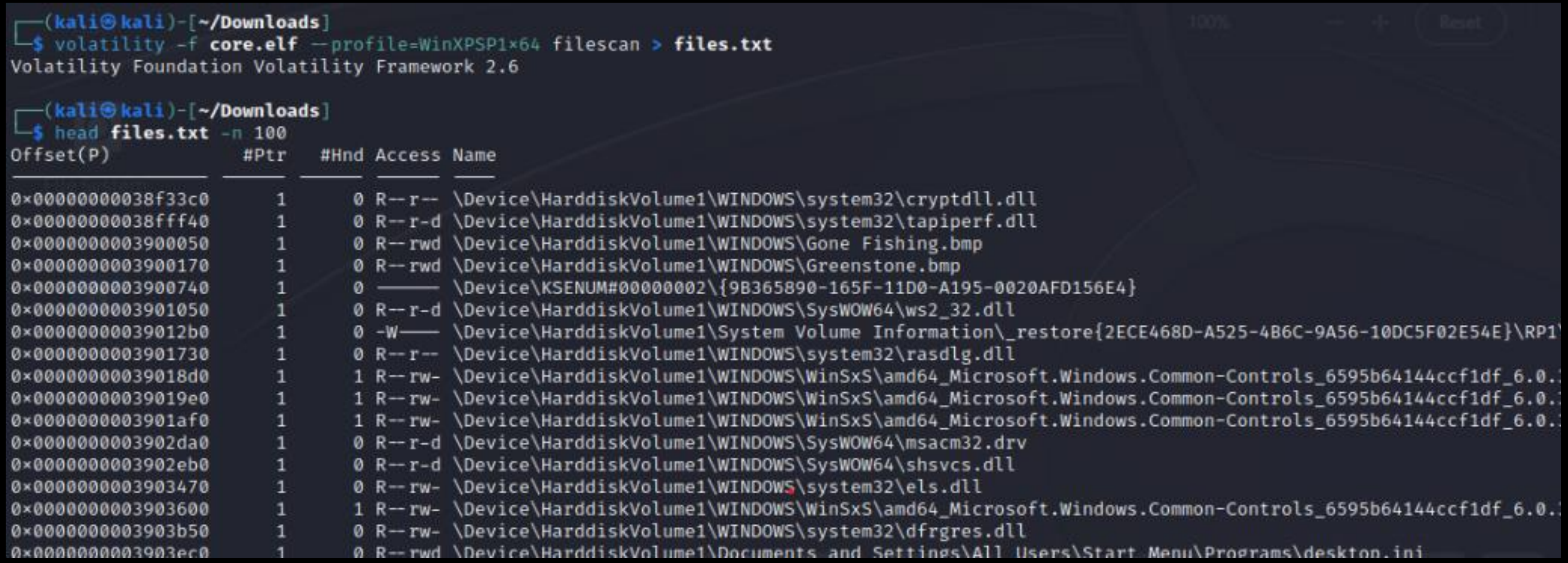


```
┌──(kali㉿kali)-[~/Downloads]
└─$ volatility -f core.elf --profile=WinXPSP1x64 pstree > processes_tree.txt
Volatility Foundation Volatility Framework 2.6

┌──(kali㉿kali)-[~/Downloads]
└─$ cat processes_tree.txt
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0xfffffadfe78fb040:System                              4      0     55    415 1970-01-01 00:00:00 UTC+0000
. 0xfffffadfe6f228b0:smss.exe                         224      4      3     19 2023-08-28 07:43:50 UTC+0000
.. 0xfffffadfe6f04c20:winlogon.exe                    296    224     22    594 2023-08-28 07:43:51 UTC+0000
... 0xfffffadfe6e59040:services.exe                   344    296     16    258 2023-08-28 07:43:51 UTC+0000
.... 0xfffffadfe79d5c20:svchost.exe                   544    344      5     86 2023-08-28 07:43:51 UTC+0000
..... 0xfffffadfe6e9b040:wmiprvse.exe                1812    544      4    160 2023-08-28 07:44:03 UTC+0000
.... 0xfffffadfe6cc7040:svchost.exe                  1124    344      2     66 2023-08-28 07:44:00 UTC+0000
.... 0xfffffadfe6d5cc20:svchost.exe                   812    344     20    262 2023-08-28 07:43:52 UTC+0000
.... 0xfffffadfe6ca2c20:svchost.exe                  1224    344      5    104 2023-08-28 07:44:00 UTC+0000
.... 0xfffffadfe6d998b0:svchost.exe                   644    344      9    239 2023-08-28 07:43:51 UTC+0000
.... 0xfffffadfe6c0ec20:alg.exe                      1828    344      6     87 2023-08-28 07:44:03 UTC+0000
.... 0xfffffadfe6d7d8b0:svchost.exe                   688    344     57   1196 2023-08-28 07:43:52 UTC+0000
..... 0xfffffadfe6b5bb10:wscntfy.exe                 2076    688      1     30 2023-08-28 07:44:05 UTC+0000
.... 0xfffffadfe6d26c20:spoolsv.exe                   956    344     12    117 2023-08-28 07:43:52 UTC+0000
.... 0xfffffadfe6d74b10:svchost.exe                   736    344      7    128 2023-08-28 07:43:52 UTC+0000
... 0xfffffadfe6e55a50:lsass.exe                      356    296     22    351 2023-08-28 07:43:51 UTC+0000
.. 0xfffffadfe6e6ac20:csrss.exe                       272    224     11    315 2023-08-28 07:43:50 UTC+0000
 0xfffffadfe6bed8b0:explorer.exe                      616    276     11    341 2023-08-28 07:44:05 UTC+0000
. 0xfffffadfe6b22040:mspaint.exe                     1076    616      5    126 2023-08-27 21:45:47 UTC+0000
```

Files

Syntax:
```
volatility -f <image_file> --profile=<profile> filescan
```

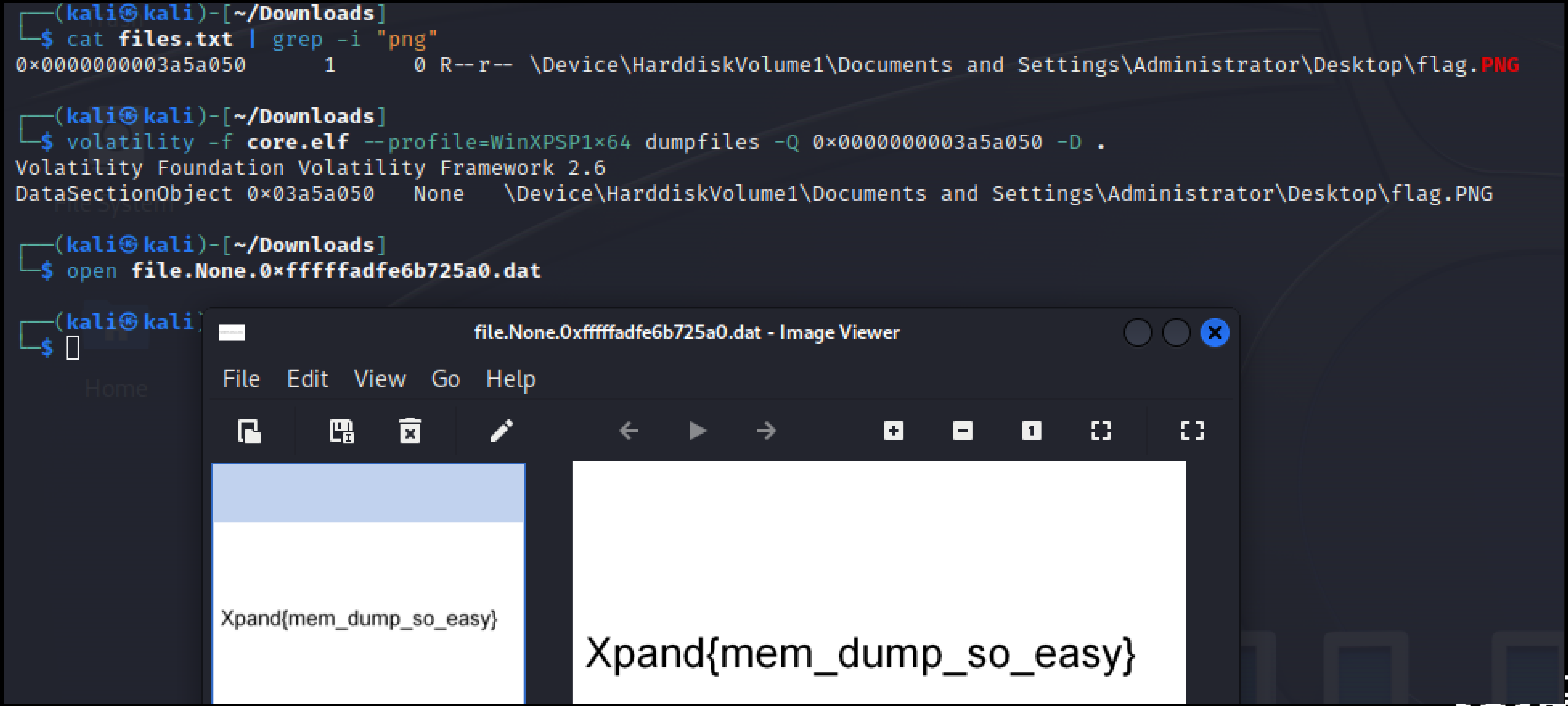


```
┌──(kali㉿kali)-[~/Downloads]
└─$ volatility -f core.elf --profile=WinXPSP1x64 filescan > files.txt
Volatility Foundation Volatility Framework 2.6

┌──(kali㉿kali)-[~/Downloads]
└─$ head files.txt -n 100
Offset(P)            #Ptr   #Hnd Access Name
------------------ ------ ------ ------ ----
0x00000000038f33c0      1      0 R--r-- \Device\HarddiskVolume1\WINDOWS\system32\cryptdll.dll
0x00000000038fff40      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\tapiperf.dll
0x0000000003900050      1      0 R--rwd \Device\HarddiskVolume1\WINDOWS\Gone Fishing.bmp
0x0000000003900170      1      0 R--rwd \Device\HarddiskVolume1\WINDOWS\Greenstone.bmp
0x0000000003900740      1      0 ------ \Device\KSENUM#00000002\{9B365890-165F-11D0-A195-0020AFD156E4}
0x0000000003901050      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\SysWOW64\ws2_32.dll
0x00000000039012b0      1      0 -W---- \Device\HarddiskVolume1\System Volume Information\_restore{2ECE468D-A525-4B6C-9A56-10DC5F02E54E}\RP1\
0x0000000003901730      1      0 R--r-- \Device\HarddiskVolume1\WINDOWS\system32\rasdlg.dll
0x00000000039018d0      1      1 R--rw- \Device\HarddiskVolume1\WINDOWS\WinSxS\amd64_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.
0x00000000039019e0      1      1 R--rw- \Device\HarddiskVolume1\WINDOWS\WinSxS\amd64_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.
0x0000000003901af0      1      1 R--rw- \Device\HarddiskVolume1\WINDOWS\WinSxS\amd64_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.
0x0000000003902da0      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\SysWOW64\msacm32.drv
0x0000000003902eb0      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\SysWOW64\shsvcs.dll
0x0000000003903470      1      0 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\els.dll
0x0000000003903600      1      1 R--rw- \Device\HarddiskVolume1\WINDOWS\WinSxS\amd64_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.
0x0000000003903b50      1      0 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\dfrgres.dll
0x0000000003903ec0      1      0 R--rwd \Device\HarddiskVolume1\Documents and Settings\All Users\Start Menu\Programs\desktop.ini
```

File dump

Syntax:
volatility -f <image_file> --profile=<profile> dumpfiles -Q <offset> -D <dump_dir>



```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat files.txt | grep -i "png"
0x000000003a5a050      1      0 R--r-- \Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\flag.PNG

┌──(kali㉿kali)-[~/Downloads]
└─$ volatility -f core.elf --profile=WinXPSP1x64 dumpfiles -Q 0x000000003a5a050 -D .
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x03a5a050   None   \Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\flag.PNG

┌──(kali㉿kali)-[~/Downloads]
└─$ open file.None.0xfffffadfe6b725a0.dat

┌──(kali㉿kali)
└─$ 
```

Xpand{mem_dump_so_easy}

Xpand{mem_dump_so_easy}

Lab Time