https://github.com/aguayro @9v@yr0

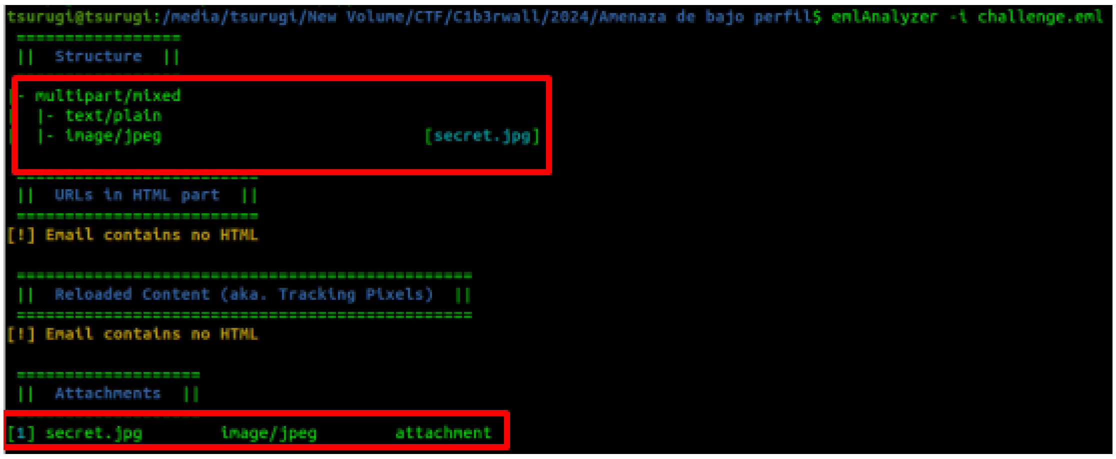
Retos CTF C1B3RWALL 2.024

Reto: Amenaza de bajo perfil

Un caso policial está en juego. Un correo electrónico con un archivo de imagen podría contener la clave para resolver el caso. Investiga el contenido del correo y examina el archivo adjunto. La información que necesitas para descifrar el misterio se esconde dentro del archivo de imagen.

Veamos que tenemos en el correo electrónico

$ emlAnalyzer -i challenge.eml

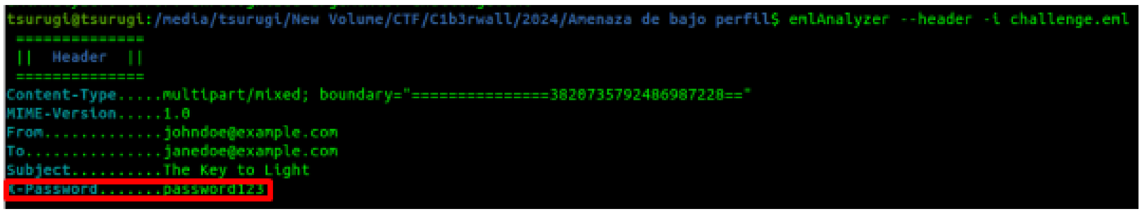


```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil$ emlAnalyzer -i challenge.eml
 =================
 ||  Structure  ||
 =================
|- multipart/mixed
|  |- text/plain
|  |- image/jpeg                                [secret.jpg]

 ==========================
 ||  URLs in HTML part  ||
 ==========================
[!] Email contains no HTML

 ==============================================
 ||  Reloaded Content (aka. Tracking Pixels)  ||
 ==============================================
[!] Email contains no HTML

 ===================
 ||  Attachments  ||
 ===================
[1] secret.jpg        image/jpeg        attachment
```

Tenemos un correo electrónico en formato de texto, sin html con una imagen jpg como adjunto.

Analizamos la cabecera del correo electrónico

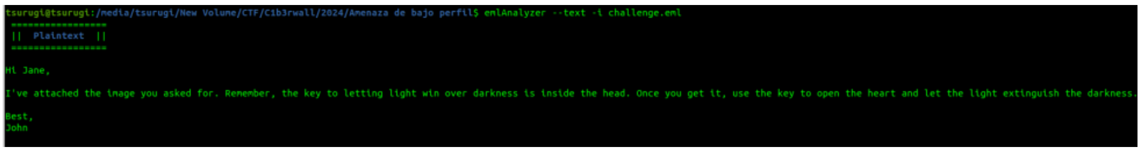
$ emlAnalyzer –header -i challenge.eml



```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil$ emlAnalyzer --header -i challenge.eml
 =============
 ||  Header  ||
 =============
Content-Type.....multipart/mixed; boundary="===============3828735792486987228=="
MIME-Version.....1.0
From.............johndoe@example.com
To...............janedoe@example.com
Subject..........The Key to Light
X-Password.......password123
```

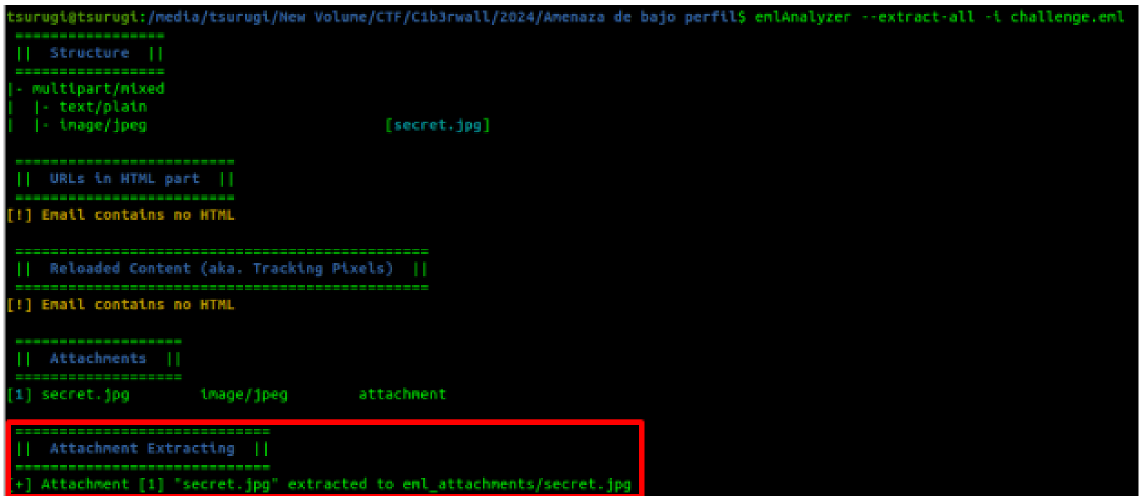
Aquí tenemos alguna Información importante, la cabecera X-Password no es una cabecera estándar por lo que es una pista.

$ emlAnalyzer -i challenge.eml –text



```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil$ emlAnalyzer --text -i challenge.eml
 ================
 ||  Plaintext  ||
 ================

Hi Jane,

I've attached the image you asked for. Remember, the key to letting light win over darkness is inside the head. Once you get it, use the key to open the heart and let the light extinguish the darkness.

Best,
John
```

$ emlAnalyzer -i challenge.eml –extract-all

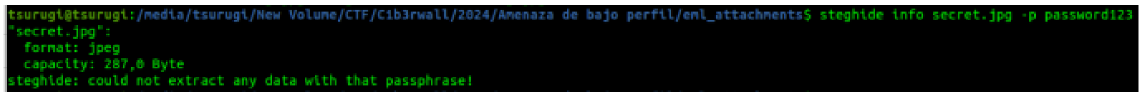


```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil$ emlAnalyzer --extract-all -i challenge.eml
 ================
 ||  Structure  ||
 ================
|- multipart/mixed
|  |- text/plain
|  |- image/jpeg                           [secret.jpg]

 =========================
 ||  URLs in HTML part  ||
 =========================
[!] Email contains no HTML

 ===============================================
 ||  Reloaded Content (aka. Tracking Pixels)  ||
 ===============================================
[!] Email contains no HTML

 ==================
 ||  Attachments  ||
 ==================
[1] secret.jpg        image/jpeg        attachment

 ============================
 ||  Attachment Extracting  ||
 ============================
[+] Attachment [1] "secret.jpg" extracted to eml_attachments/secret.jpg
```

Lo más rápido es copiar del fichero .eml el texto en base64 y decodificarlo

$ cat fichero.base64 | base64 -d > imagen.jpg

$ steghide info secret.jpg -p password123

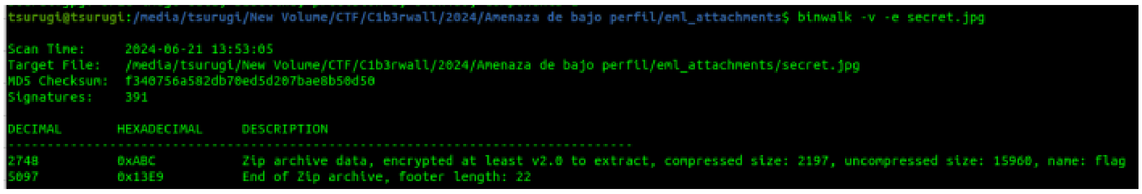


```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments$ steghide info secret.jpg -p password123
"secret.jpg":
  format: jpeg
  capacity: 287,0 Byte
steghide: could not extract any data with that passphrase!
```

La clave que tenemos en la cabecera no nos sirve para obtener información de lo que oculta la imagen.

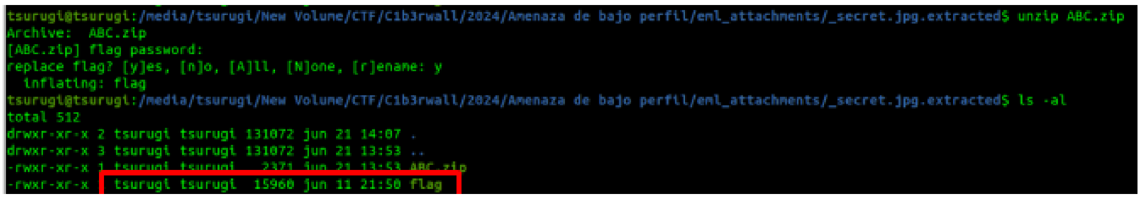
Usamos entonces binwalk para extraer toda lo que oculta la imagen.

$ binwalk -v -e secret.jpg



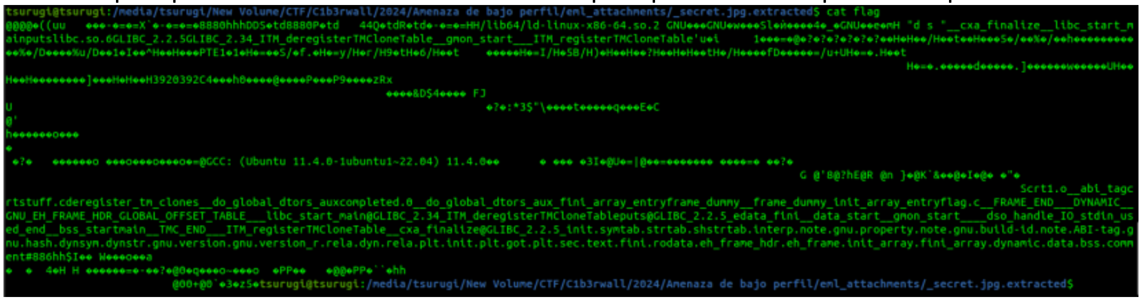
```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments$ binwalk -v -e secret.jpg

Scan Time:     2024-06-21 13:53:05
Target File:   /media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments/secret.jpg
MD5 Checksum:  f340756a582db70ed5d207bae8b50d50
Signatures:    391

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
2748          0xABC           Zip archive data, encrypted at least v2.0 to extract, compressed size: 2197, uncompressed size: 15960, name: flag
5097          0x13E9          End of Zip archive, footer length: 22
```

Parece que hemos tenido suerte y nos indica que hay dos fichero ABC.zip y flaq



```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments/_secret.jpg.extracted$ unzip ABC.zip
Archive:  ABC.zip
[ABC.zip] flag password:
replace flag? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: flag
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments/_secret.jpg.extracted$ ls -al
total 512
drwxr-xr-x 2 tsurugi tsurugi 131072 jun 21 14:07 .
drwxr-xr-x 3 tsurugi tsurugi 131072 jun 21 13:53 ..
-rwxr-xr-x 1 tsurugi tsurugi   2371 jun 21 13:53 ABC.zip
-rwxr-xr-x 1 tsurugi tsurugi  15960 jun 11 21:50 flag
```

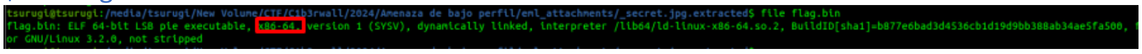
La clave que aparecía en la cabera del correo era para poder descomprimir el zip.



```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments/_secret.jpg.extracted$ cat flag
@@@@◆((uu   ◆◆◆-◆=◆=X`◆-◆=◆=◆8880hhhDDS◆td8880P◆td   44Q◆tdR◆td◆-◆=◆=HH/lib64/ld-linux-x86-64.so.2 GNU◆◆◆GNU◆w◆◆◆Sl◆й◆◆◆◆4◆_◆GNU◆◆◆◆mH "d s "__cxa_finalize__libc_start_m
ainputslibc.so.6GLIBC_2.2.5GLIBC_2.34_ITM_deregisterTMCloneTable__gmon_start___ITM_registerTMCloneTable'u◆i     1◆◆◆=◆@◆?◆?◆?◆?◆?◆?◆◆H◆H◆◆/H◆◆t◆◆H◆◆◆S◆/◆◆%◆/◆◆h◆◆◆◆◆◆◆◆◆
◆◆%◆/D◆◆◆◆%u/D◆◆1◆I◆◆^H◆◆H◆◆◆PTE1◆1◆H◆=◆◆S/◆f.◆H◆=y/H◆r/H9◆tH◆6/H◆◆t    ◆◆◆◆◆H◆=I/H◆5B/H)◆H◆◆H◆◆?H◆◆H◆H◆◆tH◆/H◆◆◆◆fD◆◆◆◆◆=/u+UH◆=◆.H◆◆t
                                                                                                          H◆=◆.◆◆◆◆◆d◆◆◆◆◆.]◆◆◆◆◆◆w◆◆◆◆◆UH◆◆
H◆◆H◆◆◆◆◆◆◆◆◆]◆◆◆H◆H◆◆H3920392C4◆◆◆h0◆◆◆◆◆@◆◆◆◆P◆◆◆P9◆◆◆◆◆zRx
                                                       ◆◆◆◆&D$4◆◆◆◆ FJ
U                                                                 ◆?◆:*3$"\◆◆◆◆t◆◆◆◆◆q◆◆◆E◆C
@'
h◆◆◆◆◆◆o◆◆◆
◆
 ◆?◆   ◆◆◆◆◆◆o ◆◆◆o◆◆◆◆o◆◆◆◆◆o◆=@GCC: (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0◆◆      ◆ ◆◆◆ ◆3I◆@U◆=|@◆◆=◆◆◆◆◆◆◆ ◆◆◆◆=◆ ◆◆?◆
                                                                                              G @'B@?hE@R @n }◆@K`&◆◆@◆I◆@◆ ◆"◆
                                                                                                                         Scrt1.o__abi_tagc
rtstuff.cderegister_tm_clones__do_global_dtors_auxcompleted.0__do_global_dtors_aux_fini_array_entryframe_dummy__frame_dummy_init_array_entryflag.c__FRAME_END___DYNAMIC__
GNU_EH_FRAME_HDR_GLOBAL_OFFSET_TABLE___libc_start_main@GLIBC_2.34_ITM_deregisterTMCloneTableputs@GLIBC_2.2.5_edata_fini__data_start__gmon_start____dso_handle_IO_stdin_us
ed_end__bss_startmain__TMC_END___ITM_registerTMCloneTable__cxa_finalize@GLIBC_2.2.5_init.symtab.strtab.shstrtab.interp.note.gnu.property.note.gnu.build-id.note.ABI-tag.g
nu.hash.dynsym.dynstr.gnu.version.gnu.version_r.rela.dyn.rela.plt.init.plt.got.plt.sec.text.fini.rodata.eh_frame_hdr.eh_frame.init_array.fini_array.dynamic.data.bss.comm
ent#886hh$I◆◆ W◆◆◆◆o◆◆a
◆  ◆  4◆H H ◆◆◆◆◆◆=◆-◆◆?◆@0◆q◆◆◆◆o~◆◆◆◆  ◆PP◆◆   ◆@@◆PP◆``◆hh
                        @00+@0`◆3◆z5◆tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments/_secret.jpg.extracted$
```

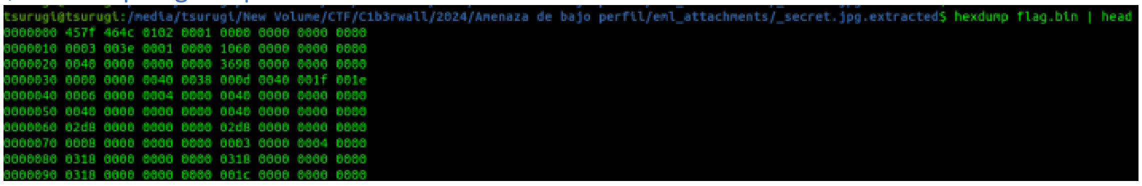
Tenemos un fichero binario, vamos a ver qué tipo de fichero contiene

$ file flag.bin



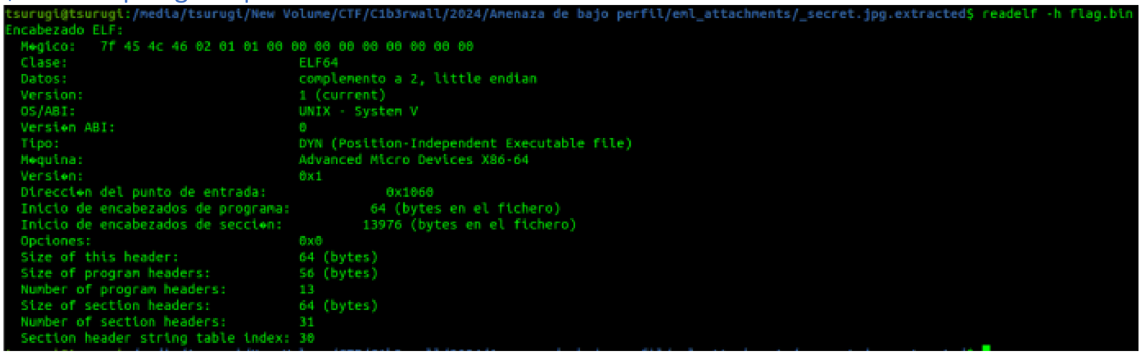
```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments/_secret.jpg.extracted$ file flag.bin
flag.bin: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=b877e6bad3d4536cb1d19d9bb388ab34ae5fa500, f
or GNU/Linux 3.2.0, not stripped
```

$ hexdump flag.bin | head



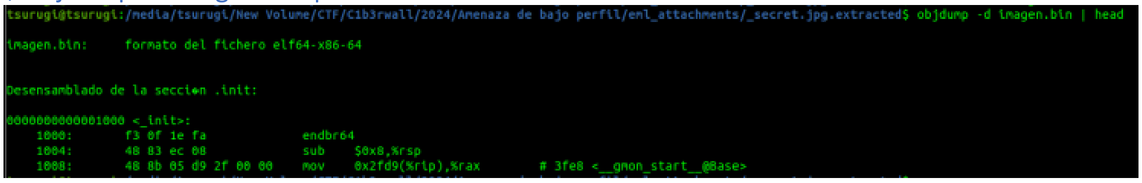
```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments/_secret.jpg.extracted$ hexdump flag.bin | head
0000000 457f 464c 0102 0001 0000 0000 0000 0000
0000010 0003 003e 0001 0000 1060 0000 0000 0000
0000020 0040 0000 0000 0000 3698 0000 0000 0000
0000030 0000 0000 0040 0038 000d 0040 001f 001e
0000040 0006 0000 0004 0000 0040 0000 0000 0000
0000050 0040 0000 0000 0000 0040 0000 0000 0000
0000060 02d8 0000 0000 0000 02d8 0000 0000 0000
0000070 0008 0000 0000 0000 0003 0000 0004 0000
0000080 0318 0000 0000 0000 0318 0000 0000 0000
0000090 0318 0000 0000 0000 001c 0000 0000 0000
```

$ hexdump flag.bin | head



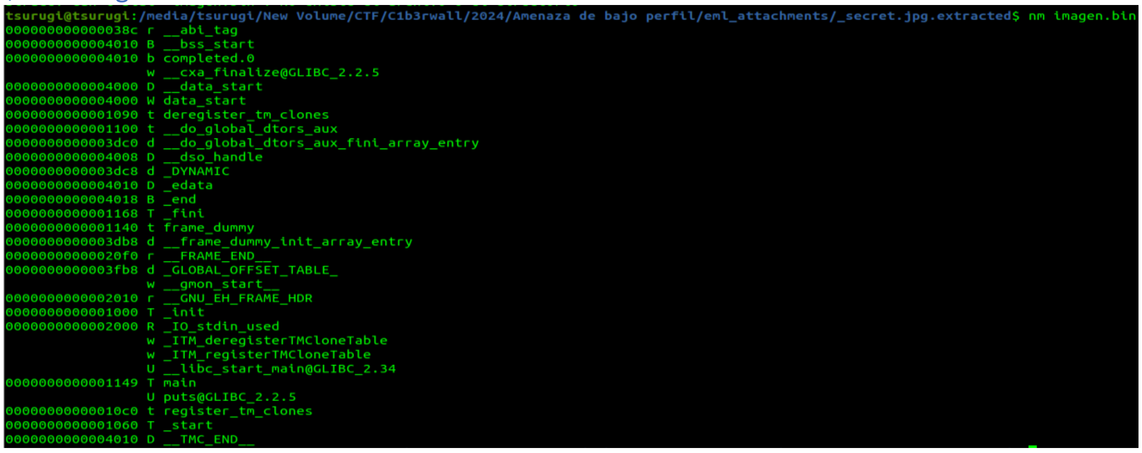
```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments/_secret.jpg.extracted$ readelf -h flag.bin
Encabezado ELF:
  M◆gico:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Clase:                             ELF64
  Datos:                             complemento a 2, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  Versi◆n ABI:                       0
  Tipo:                              DYN (Position-Independent Executable file)
  M◆quina:                           Advanced Micro Devices X86-64
  Versi◆n:                           0x1
  Direcci◆n del punto de entrada:               0x1060
  Inicio de encabezados de programa:          64 (bytes en el fichero)
  Inicio de encabezados de secci◆n:          13976 (bytes en el fichero)
  Opciones:                          0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         13
  Size of section headers:           64 (bytes)
  Number of section headers:         31
  Section header string table index: 30
```

$ objdump -d imagen.bin | head



```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments/_secret.jpg.extracted$ objdump -d imagen.bin | head

imagen.bin:     formato del fichero elf64-x86-64


Desensamblado de la secci�n .init:

0000000000001000 <_init>:
    1000:       f3 0f 1e fa             endbr64
    1004:       48 83 ec 08             sub    $0x8,%rsp
    1008:       48 8b 05 d9 2f 00 00    mov    0x2fd9(%rip),%rax        # 3fe8 <__gmon_start__@Base>
```

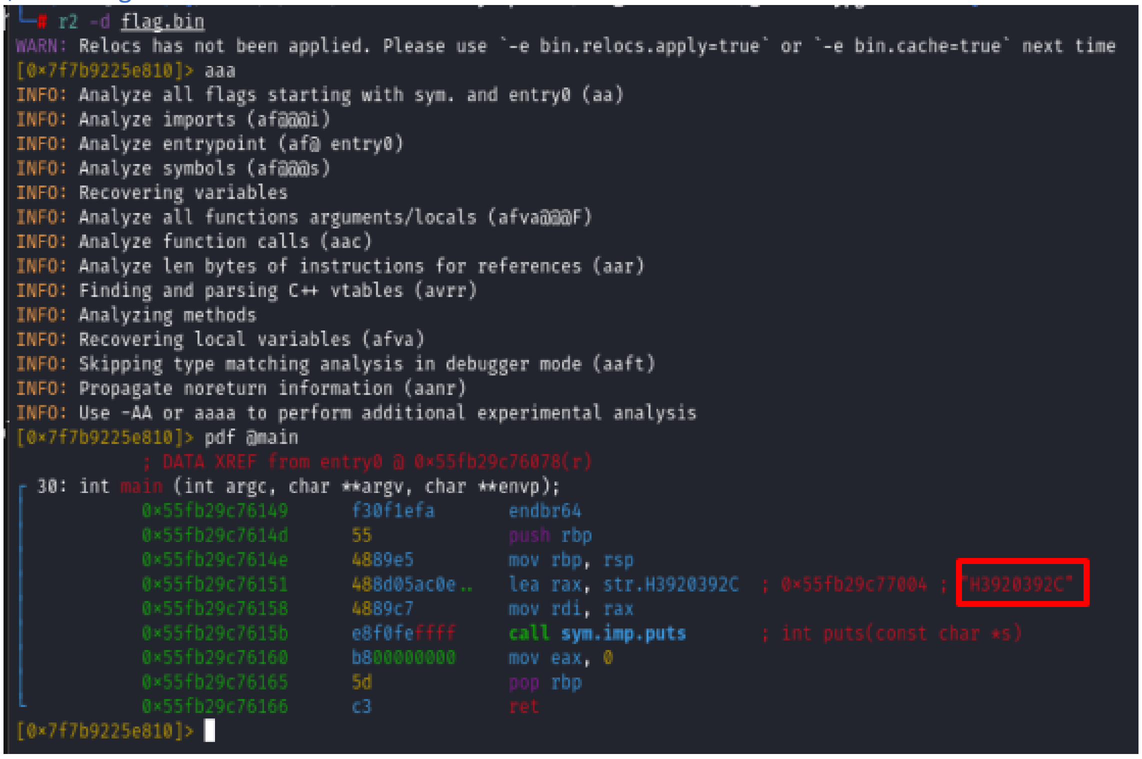
$ nm imagen.bin



```
tsurugi@tsurugi:/media/tsurugi/New Volume/CTF/C1b3rwall/2024/Amenaza de bajo perfil/eml_attachments/_secret.jpg.extracted$ nm imagen.bin
000000000000038c r __abi_tag
0000000000004010 B __bss_start
0000000000004010 b completed.0
                 w __cxa_finalize@GLIBC_2.2.5
0000000000004000 D __data_start
0000000000004000 W data_start
0000000000001090 t deregister_tm_clones
0000000000001100 t __do_global_dtors_aux
0000000000003dc0 d __do_global_dtors_aux_fini_array_entry
0000000000004008 D __dso_handle
0000000000003dc8 d _DYNAMIC
0000000000004010 D _edata
0000000000004018 B _end
0000000000001168 T _fini
0000000000001140 t frame_dummy
0000000000003db8 d __frame_dummy_init_array_entry
00000000000020f0 r __FRAME_END__
0000000000003fb8 d _GLOBAL_OFFSET_TABLE_
                 w __gmon_start__
0000000000002010 r __GNU_EH_FRAME_HDR
0000000000001000 T _init
0000000000002000 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
                 U __libc_start_main@GLIBC_2.34
0000000000001149 T main
                 U puts@GLIBC_2.2.5
00000000000010c0 t register_tm_clones
0000000000001060 T _start
0000000000004010 D __TMC_END__
```

Analizamos el fichero con Radare2

$ r2 -d flag.bin



```
└─# r2 -d flag.bin
WARN: Relocs has not been applied. Please use `-e bin.relocs.apply=true` or `-e bin.cache=true` next time
[0x7f7b9225e810]> aaa
INFO: Analyze all flags starting with sym. and entry0 (aa)
INFO: Analyze imports (af@@@i)
INFO: Analyze entrypoint (af@ entry0)
INFO: Analyze symbols (af@@@s)
INFO: Recovering variables
INFO: Analyze all functions arguments/locals (afva@@@F)
INFO: Analyze function calls (aac)
INFO: Analyze len bytes of instructions for references (aar)
INFO: Finding and parsing C++ vtables (avrr)
INFO: Analyzing methods
INFO: Recovering local variables (afva)
INFO: Skipping type matching analysis in debugger mode (aaft)
INFO: Propagate noreturn information (aanr)
INFO: Use -AA or aaaa to perform additional experimental analysis
[0x7f7b9225e810]> pdf @main
            ; DATA XREF from entry0 @ 0x55fb29c76078(r)
┌ 30: int main (int argc, char **argv, char **envp);
│           0x55fb29c76149      f30f1efa       endbr64
│           0x55fb29c7614d      55             push rbp
│           0x55fb29c7614e      4889e5         mov rbp, rsp
│           0x55fb29c76151      488d05ac0e..   lea rax, str.H3920392C    ; 0x55fb29c77004 ; "H3920392C"
│           0x55fb29c76158      4889c7         mov rdi, rax
│           0x55fb29c7615b      e8f0feffff     call sym.imp.puts         ; int puts(const char *s)
│           0x55fb29c76160      b800000000     mov eax, 0
│           0x55fb29c76165      5d             pop rbp
└           0x55fb29c76166      c3             ret
[0x7f7b9225e810]>
```

Flag: H3920392C

Herramientas:

Binwalk

ElmAnazyler

Hexdump

Strings

Realelf

Nm

Gdb

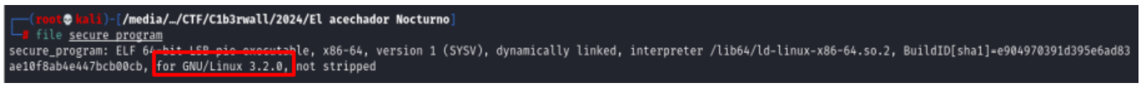
Radare2

Reto: El acechador Nocturno

Hay un fichero con información sospechosa

Analizamos el contenido del fichero secure_program

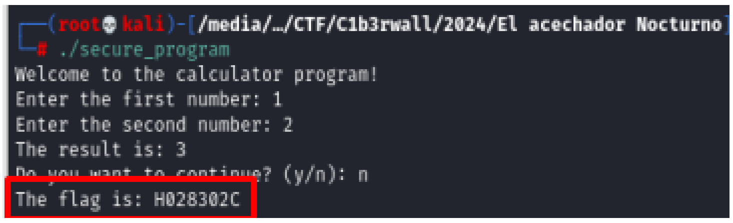
$ file secure_program



```
┌──(root㉿kali)-[/media/…/CTF/C1b3rwall/2024/El acechador Nocturno]
└─# file secure_program
secure_program: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=e904970391d395e6ad83
ae10f8ab4e447bcb00cb, for GNU/Linux 3.2.0, not stripped
```

Es un fichero binario de Linux, vamos a ejecutarlo a ver lo que nos desvela

$ ./secure_program

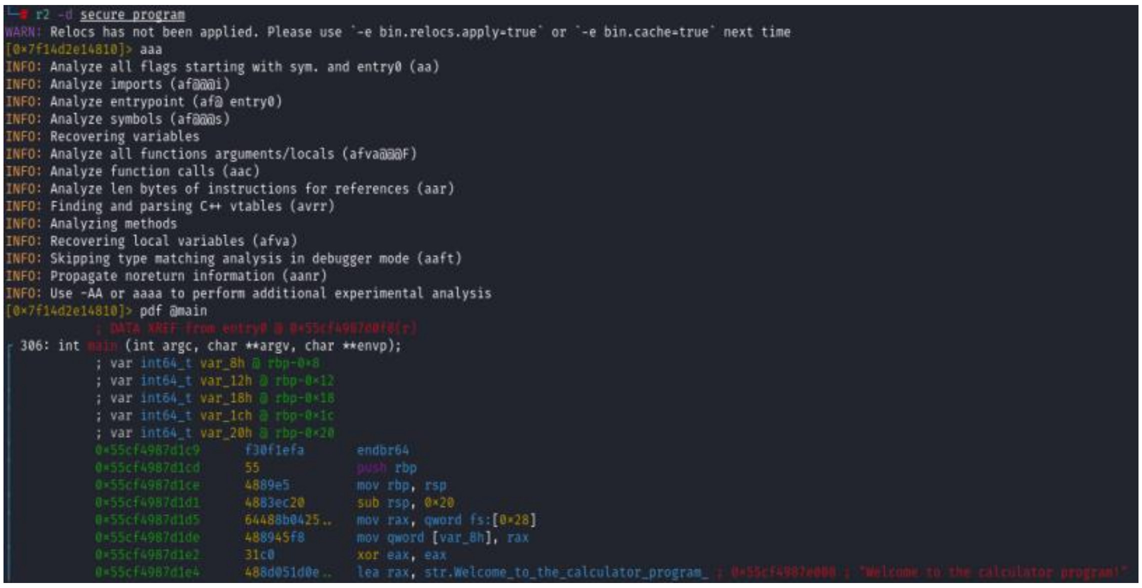


```
┌──(root㉿kali)-[/media/…/CTF/C1b3rwall/2024/El acechador Nocturno]
└─# ./secure_program
Welcome to the calculator program!
Enter the first number: 1
Enter the second number: 2
The result is: 3
Do you want to continue? (y/n): n
The flag is: H028302C
```

Tenemos la flag: H028302C

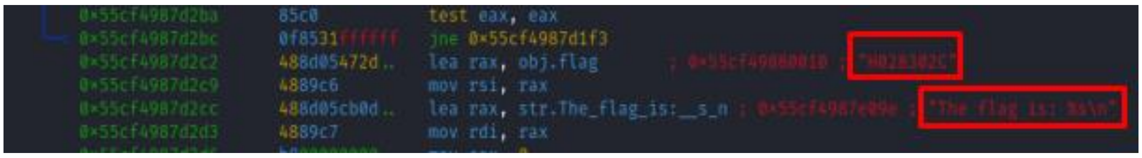
Vamos un poco más allá y vamos a decompilar el binario:

$ r2 -d secure_program



```
└─# r2 -d secure_program
WARN: Relocs has not been applied. Please use `-e bin.relocs.apply=true` or `-e bin.cache=true` next time
[0×7f14d2e14810]> aaa
INFO: Analyze all flags starting with sym. and entry0 (aa)
INFO: Analyze imports (af@@@i)
INFO: Analyze entrypoint (af@ entry0)
INFO: Analyze symbols (af@@@s)
INFO: Recovering variables
INFO: Analyze all functions arguments/locals (afva@@@F)
INFO: Analyze function calls (aac)
INFO: Analyze len bytes of instructions for references (aar)
INFO: Finding and parsing C++ vtables (avrr)
INFO: Analyzing methods
INFO: Recovering local variables (afva)
INFO: Skipping type matching analysis in debugger mode (aaft)
INFO: Propagate noreturn information (aanr)
INFO: Use -AA or aaaa to perform additional experimental analysis
[0×7f14d2e14810]> pdf @main
            ; DATA XREF from entry0 @ 0×55cf4987e0f8(r)
┌ 306: int main (int argc, char **argv, char **envp);
│           ; var int64_t var_8h @ rbp-0×8
│           ; var int64_t var_12h @ rbp-0×12
│           ; var int64_t var_18h @ rbp-0×18
│           ; var int64_t var_1ch @ rbp-0×1c
│           ; var int64_t var_20h @ rbp-0×20
│           0×55cf4987d1c9      f30f1efa       endbr64
│           0×55cf4987d1cd      55             push rbp
│           0×55cf4987d1ce      4889e5         mov rbp, rsp
│           0×55cf4987d1d1      4883ec20       sub rsp, 0×20
│           0×55cf4987d1d5      64488b0425..   mov rax, qword fs:[0×28]
│           0×55cf4987d1de      488945f8       mov qword [var_8h], rax
│           0×55cf4987d1e2      31c0           xor eax, eax
│           0×55cf4987d1e4      488d051d0e..   lea rax, str.Welcome_to_the_calculator_program_ ; 0×55cf4987e008 ; "Welcome to the calculator program!"
```

Más abajo tenemos el contenido de la flag



```
      0×55cf4987d2ba      85c0           test eax, eax
    └ 0×55cf4987d2bc      0f8531ffffff   jne 0×55cf4987d1f3
      0×55cf4987d2c2      488d05472d..   lea rax, obj.flag        ; 0×55cf49880010 ; "H028302C"
      0×55cf4987d2c9      4889c6         mov rsi, rax
      0×55cf4987d2cc      488d05cb0d..   lea rax, str.The_flag_is:__s_n ; 0×55cf4987e09e ; "The flag is: %s\n"
      0×55cf4987d2d3      4889c7         mov rdi, rax
```

Flag: H028302C

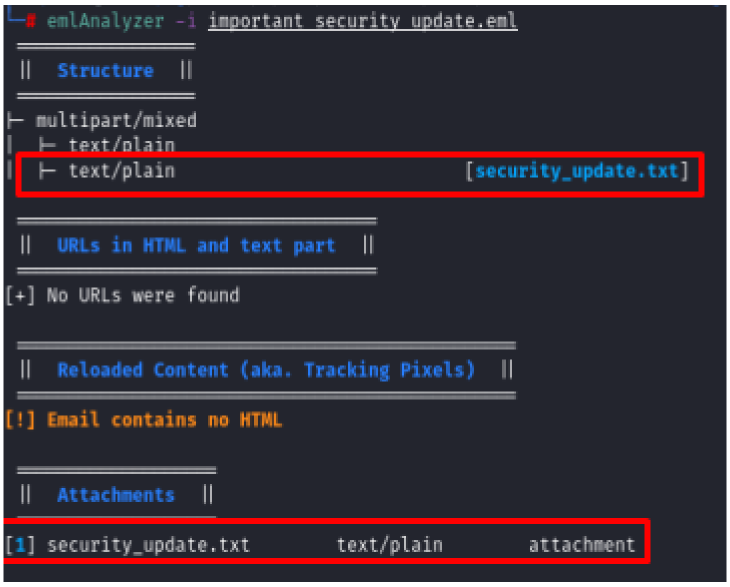
Reto: El enigma de la Araña

Pero algo no cuadra. Las palabras Support Team podrían ser una pista falsa. El Correo electrónico es aparentemente simple, pero solo un detective digital con un ojo agudo y una mente perspicaz puede encontrar la pista crucial.

El Enigma de la Araña te desafía a descifrar el correo electrónico.

Veamos la estructura del correo

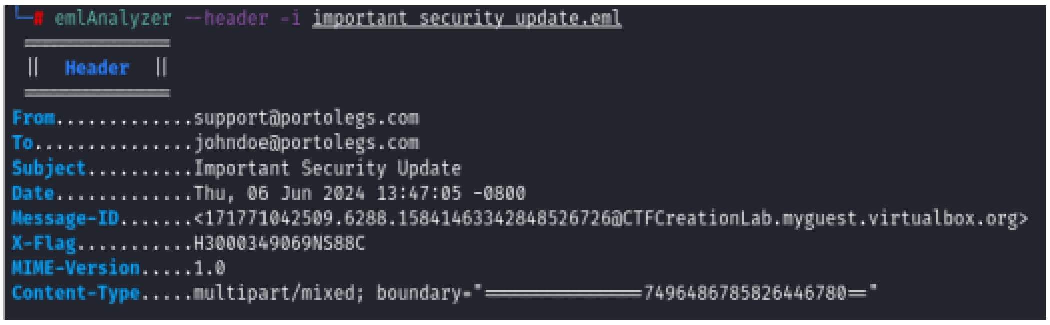
$ emlAnalyzer -i important_security_update.eml



```
└─# emlAnalyzer -i important_security_update.eml
 ═══════════════
 ||  Structure  ||
 ═══════════════
├ multipart/mixed
│  ├ text/plain
│  ├ text/plain                          [security_update.txt]

 ═══════════════════════════
 ||  URLs in HTML and text part  ||
 ═══════════════════════════
[+] No URLs were found

 ═══════════════════════════════════
 ||  Reloaded Content (aka. Tracking Pixels)  ||
 ═══════════════════════════════════
[!] Email contains no HTML

 ═════════════════
 ||  Attachments  ||
 ═════════════════
[1] security_update.txt        text/plain        attachment
```

Email en texto plano con un fichero adjunto: security_update.txt

$ emlAnalyzer –header -i important_security_update.eml

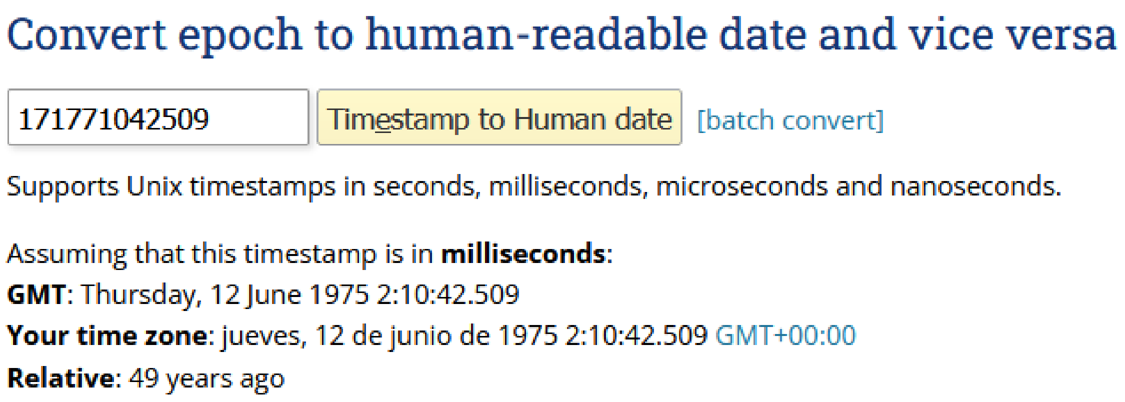


```
└─# emlAnalyzer --header -i important_security_update.eml
 ══════════
 ||  Header  ||
 ══════════
From............support@portolegs.com
To..............johndoe@portolegs.com
Subject.........Important Security Update
Date............Thu, 06 Jun 2024 13:47:05 -0800
Message-ID.......<171771042509.6288.15841463342848526726@CTFCreationLab.myguest.virtualbox.org>
X-Flag..........H3000349069NS88C
MIME-Version.....1.0
Content-Type.....multipart/mixed; boundary="===============7496486785826446780=="
```

Analizamos la cabecera, nos centramos en el Message-Id y el campo X-Flag.

Message-ID.......<171771042509.6288.15841463342848526726@CTFCreationLab.myguest.virtualbox.org>
171771042509 se corresponde con la fecha en formato timestamp.
6288 se corresponde valor aleatorio incremental.
15841463342848526726 corresponde es otro valor aleatorio númerico incremental usado entre el servidor de correo y el cliente.
CTFCreationLab.myguest.virtualbox.org corresponde con el nombre de máquina y del dominio.

La fecha no se corresponde con la cabecera del e-mail que indica: Thu, 06 Jun 2024 13:47:05 -0800

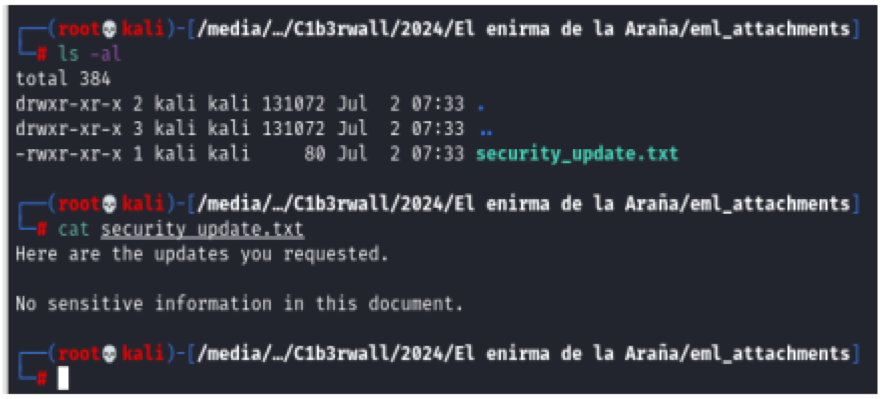


Convert epoch to human-readable date and vice versa

171771042509 | Timestamp to Human date | [batch convert]

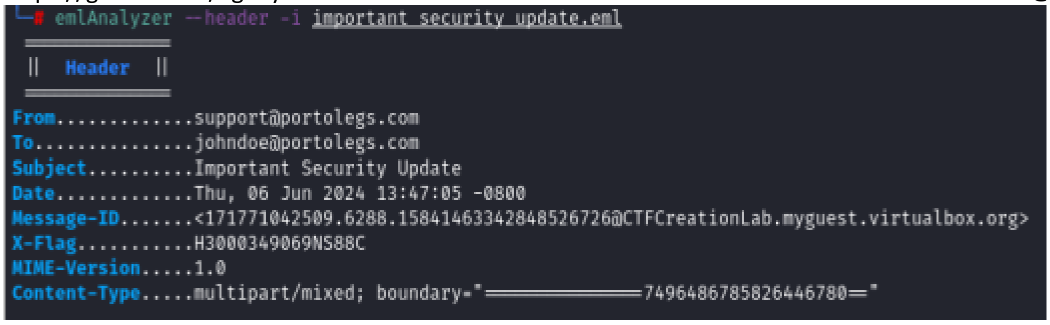
Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.

Assuming that this timestamp is in **milliseconds**:
GMT: Thursday, 12 June 1975 2:10:42.509
Your time zone: jueves, 12 de junio de 1975 2:10:42.509 GMT+00:00
Relative: 49 years ago



```
┌──(root㉿kali)-[/media/…/C1b3rwall/2024/El enirma de la Araña/eml_attachments]
└─# ls -al
total 384
drwxr-xr-x 2 kali kali 131072 Jul  2 07:33 .
drwxr-xr-x 3 kali kali 131072 Jul  2 07:33 ..
-rwxr-xr-x 1 kali kali     80 Jul  2 07:33 security_update.txt

┌──(root㉿kali)-[/media/…/C1b3rwall/2024/El enirma de la Araña/eml_attachments]
└─# cat security_update.txt
Here are the updates you requested.

No sensitive information in this document.

┌──(root㉿kali)-[/media/…/C1b3rwall/2024/El enirma de la Araña/eml_attachments]
└─# 
```

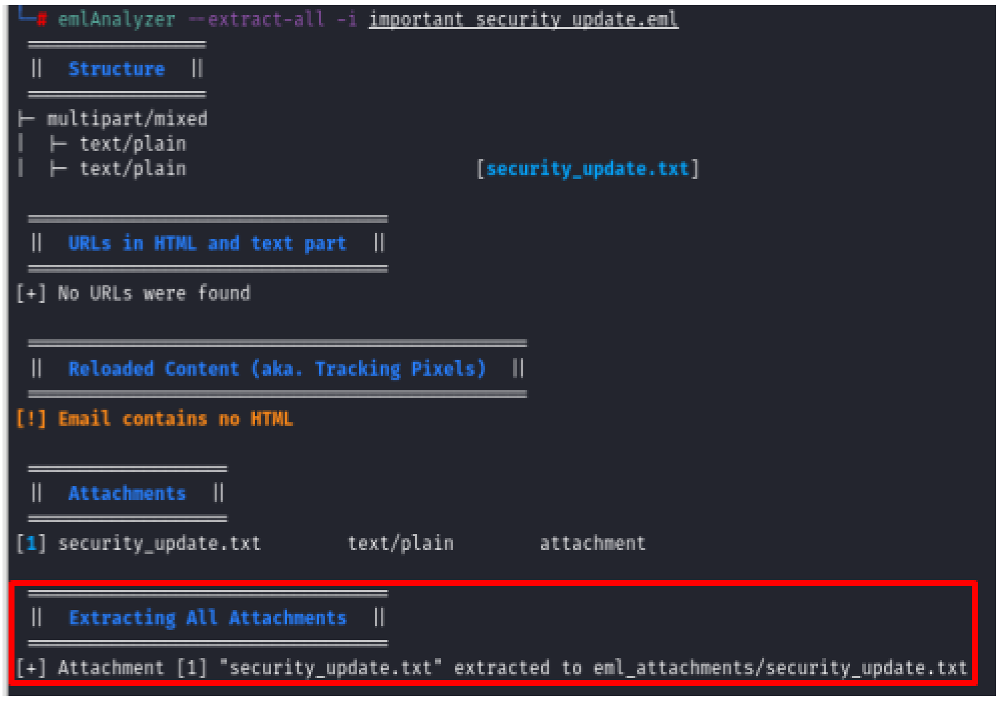
No hay nada relevante en el fichero adjunto.



```
└─# emlAnalyzer --header -i important_security_update.eml
 ════════════
 ||  Header  ||
 ════════════
From............support@portolegs.com
To..............johndoe@portolegs.com
Subject.........Important Security Update
Date............Thu, 06 Jun 2024 13:47:05 -0800
Message-ID.......<171771042509.6288.15841463342848526726@CTFCreationLab.myguest.virtualbox.org>
X-Flag..........H3000349069NS88C
MIME-Version.....1.0
Content-Type.....multipart/mixed; boundary="===============7496486785826446780=="
```

Volviendo a la cabecera, el campo X-Flag no se corresponde con un campo válido, por lo tanto la flag es ese campo:

H3000349069NS88C



```
└─# emlAnalyzer --extract-all -i important_security_update.eml
 ═══════════════
 ||  Structure  ||
 ═══════════════
├─ multipart/mixed
│  ├─ text/plain
│  ├─ text/plain                            [security_update.txt]

 ══════════════════════════════
 ||  URLs in HTML and text part  ||
 ══════════════════════════════
[+] No URLs were found

 ═════════════════════════════════════════
 ||  Reloaded Content (aka. Tracking Pixels)  ||
 ═════════════════════════════════════════
[!] Email contains no HTML

 ════════════════
 ||  Attachments  ||
 ════════════════
[1] security_update.txt        text/plain        attachment

 ══════════════════════════════
 ||  Extracting All Attachments  ||
 ══════════════════════════════
[+] Attachment [1] "security_update.txt" extracted to eml_attachments/security_update.txt
```

Reto: Amenaza interna

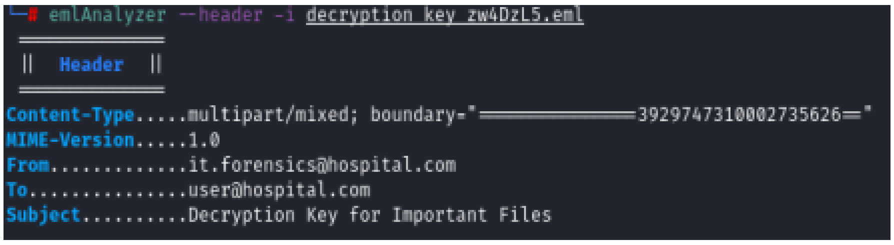
Amenaza Interna te desafía a enfrentar un escenario de ciberseguridad real.

En la escena del cibercrimen, has encontrado un conjunto de archivos: un .pcap con el Tráfico de red del ataque, un .log con Registros del sistema.

Debes analizar los archivos proporcionados, incluyendo el Correo electrónico que te envió el departamento de IT forensics, para comprender el ataque.

Revisamos el correo electrónico con emlAnalyzer

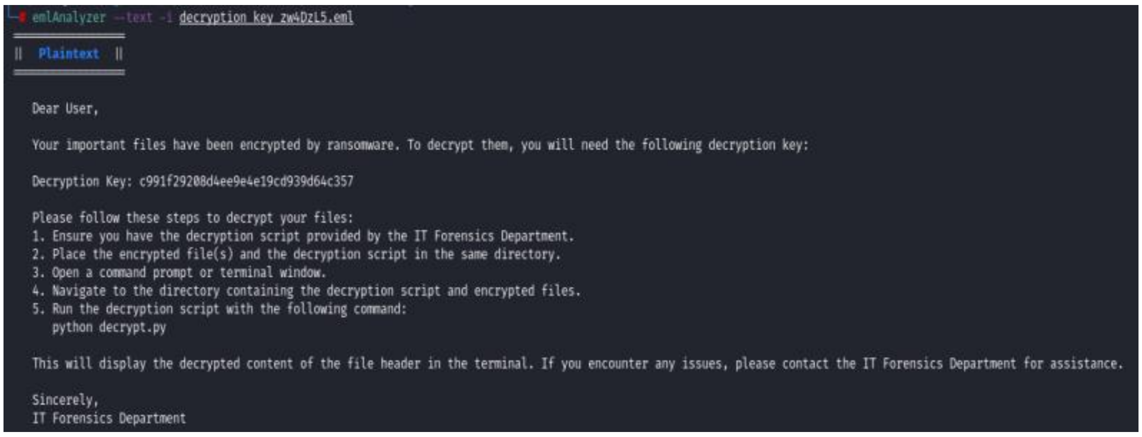
$ emlAnalyzer --header -i decryption_key_zw4DzL5.eml



```
└─# emlAnalyzer --header -i decryption_key_zw4DzL5.eml
 ════════════
 ||  Header  ||
 ════════════
Content-Type.....multipart/mixed; boundary="===============3929747310002735626=="
MIME-Version.....1.0
From.............it.forensics@hospital.com
To...............user@hospital.com
Subject..........Decryption Key for Important Files
```

El correo nos viene del dominio hospital.com del usuario it.forensics. Vamos a ver que contiene el cuerpo del correo.

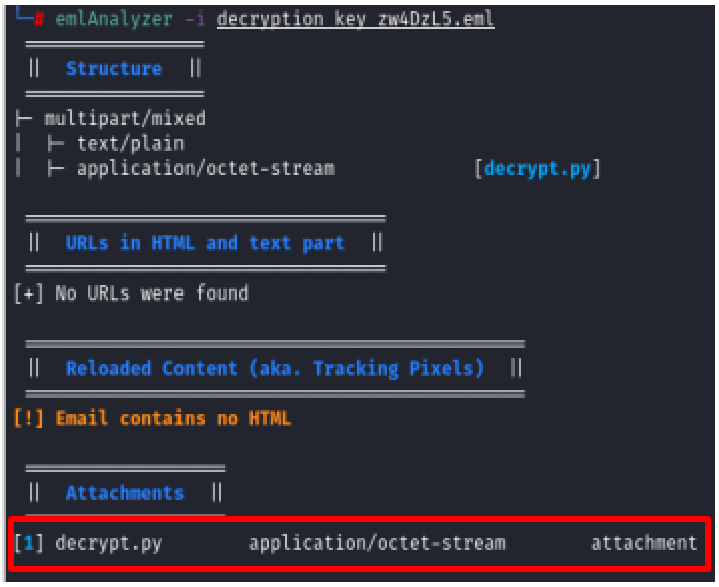
$ emlAnalyzer --text -i decryption_key_zw4DzL5.eml



```
└─# emlAnalyzer --text -i decryption_key_zw4DzL5.eml
 ════════════
 ||  Plaintext  ||
 ════════════

  Dear User,

  Your important files have been encrypted by ransomware. To decrypt them, you will need the following decryption key:

  Decryption Key: c991f29208d4ee9e4e19cd939d64c357

  Please follow these steps to decrypt your files:
  1. Ensure you have the decryption script provided by the IT Forensics Department.
  2. Place the encrypted file(s) and the decryption script in the same directory.
  3. Open a command prompt or terminal window.
  4. Navigate to the directory containing the decryption script and encrypted files.
  5. Run the decryption script with the following command:
     python decrypt.py

  This will display the decrypted content of the file header in the terminal. If you encounter any issues, please contact the IT Forensics Department for assistance.

  Sincerely,
  IT Forensics Department
```

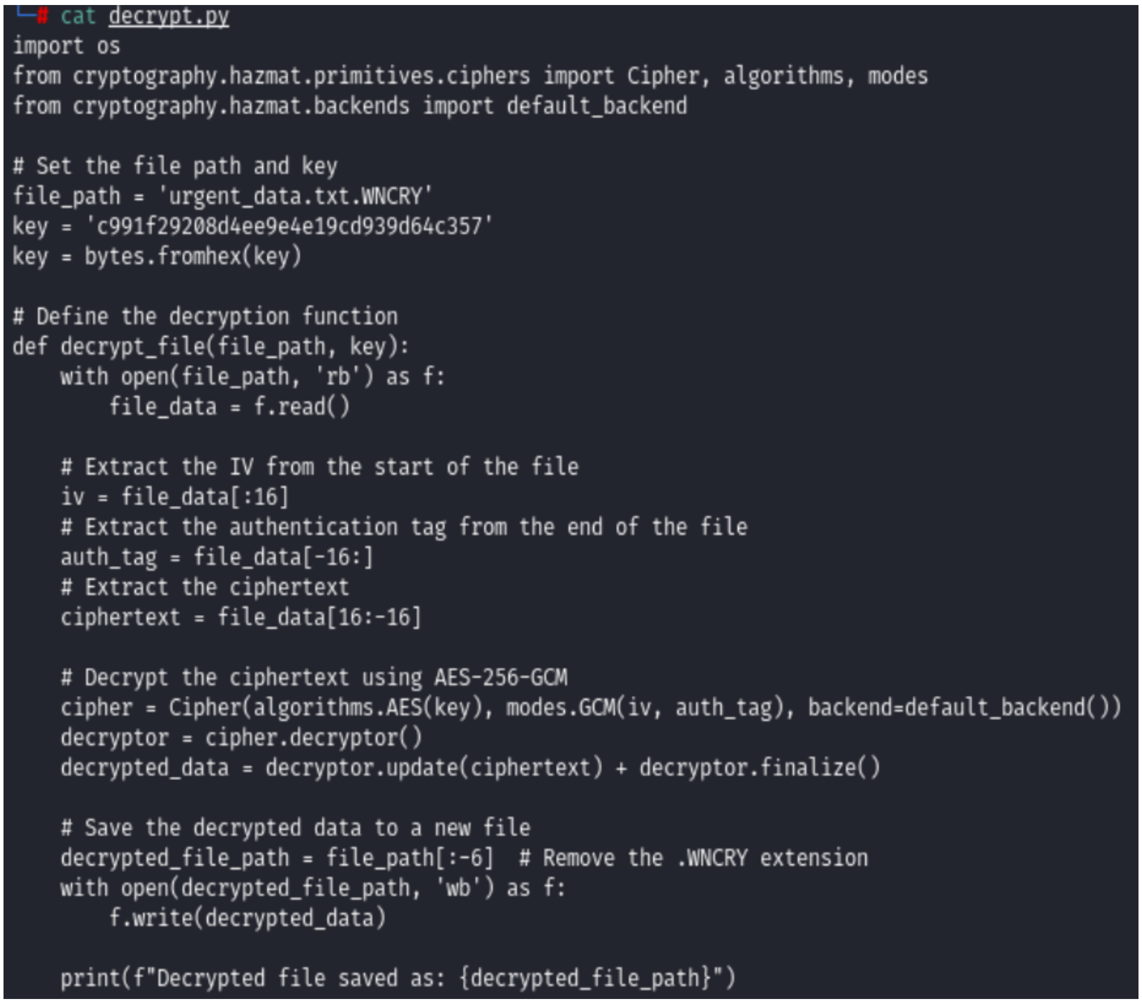
Nos comenta desde el departamento de IT que hemos sido víctima de encriptación de ficheros.

Veamos la estructura del fichero

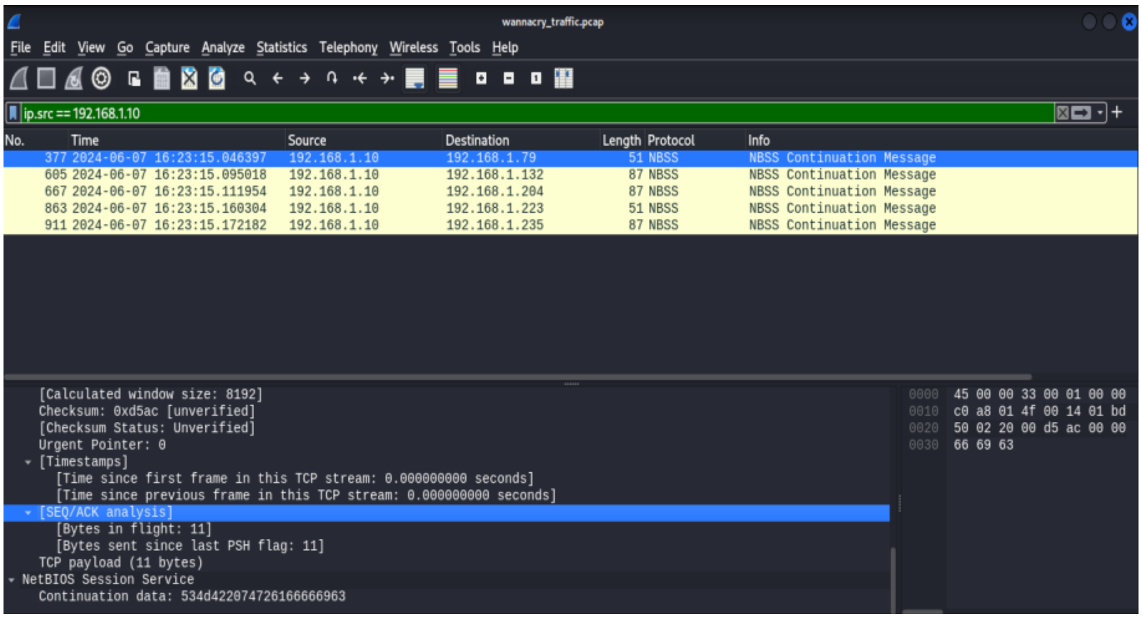


```
└─# emlAnalyzer -i decryption_key_zw4DzL5.eml
 ════════════
 ||  Structure  ||
 ════════════
├─ multipart/mixed
│  ├─ text/plain
│  ├─ application/octet-stream            [decrypt.py]

 ══════════════════════════
 ||  URLs in HTML and text part  ||
 ══════════════════════════
[+] No URLs were found

 ════════════════════════════════════
 ||  Reloaded Content (aka. Tracking Pixels)  ||
 ════════════════════════════════════
[!] Email contains no HTML

 ══════════════
 ||  Attachments  ||
 ══════════════
[1] decrypt.py        application/octet-stream        attachment
```

Contiene un fichero adjunto decrypt.py

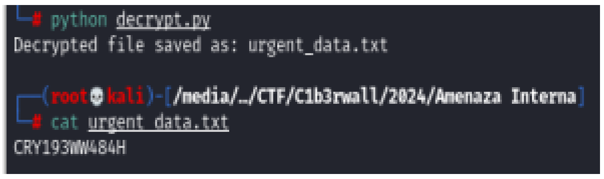


```
└─# cat decrypt.py
import os
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend

# Set the file path and key
file_path = 'urgent_data.txt.WNCRY'
key = 'c991f29208d4ee9e4e19cd939d64c357'
key = bytes.fromhex(key)

# Define the decryption function
def decrypt_file(file_path, key):
    with open(file_path, 'rb') as f:
        file_data = f.read()

    # Extract the IV from the start of the file
    iv = file_data[:16]
    # Extract the authentication tag from the end of the file
    auth_tag = file_data[-16:]
    # Extract the ciphertext
    ciphertext = file_data[16:-16]

    # Decrypt the ciphertext using AES-256-GCM
    cipher = Cipher(algorithms.AES(key), modes.GCM(iv, auth_tag), backend=default_backend())
    decryptor = cipher.decryptor()
    decrypted_data = decryptor.update(ciphertext) + decryptor.finalize()

    # Save the decrypted data to a new file
    decrypted_file_path = file_path[:-6]  # Remove the .WNCRY extension
    with open(decrypted_file_path, 'wb') as f:
        f.write(decrypted_data)

    print(f"Decrypted file saved as: {decrypted_file_path}")
```


wannacry_traffic.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

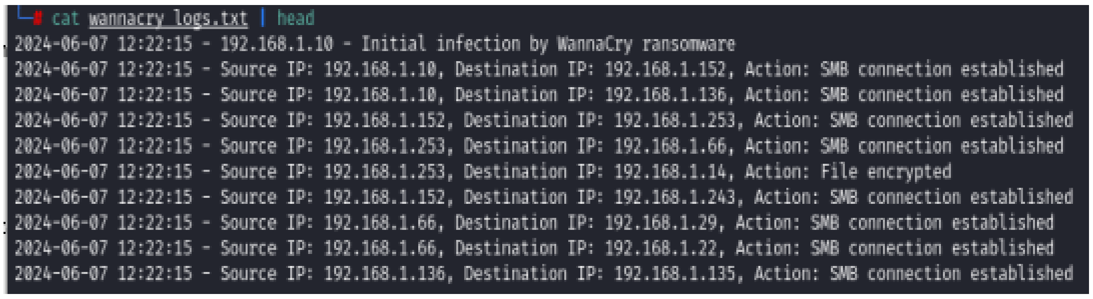
```
ip.src == 192.168.1.10

No.     Time                        Source          Destination      Length Protocol  Info
    377 2024-06-07 16:23:15.046397  192.168.1.10    192.168.1.79         51 NBSS      NBSS Continuation Message
    605 2024-06-07 16:23:15.095018  192.168.1.10    192.168.1.132        87 NBSS      NBSS Continuation Message
    667 2024-06-07 16:23:15.111954  192.168.1.10    192.168.1.204        87 NBSS      NBSS Continuation Message
    863 2024-06-07 16:23:15.160304  192.168.1.10    192.168.1.223        51 NBSS      NBSS Continuation Message
    911 2024-06-07 16:23:15.172182  192.168.1.10    192.168.1.235        87 NBSS      NBSS Continuation Message

    [Calculated window size: 8192]                                              0000  45 00 00 33 00 01 00 00
    Checksum: 0xd5ac [unverified]                                               0010  c0 a8 01 4f 00 14 01 bd
    [Checksum Status: Unverified]                                               0020  50 02 20 00 d5 ac 00 00
    Urgent Pointer: 0                                                           0030  66 69 63
  ▾ [Timestamps]
      [Time since first frame in this TCP stream: 0.000000000 seconds]
      [Time since previous frame in this TCP stream: 0.000000000 seconds]
  ▾ [SEQ/ACK analysis]
      [Bytes in flight: 11]
      [Bytes sent since last PSH flag: 11]
    TCP payload (11 bytes)
▾ NetBIOS Session Service
    Continuation data: 534d42207472616666963
```

Ejecutamos el script de python



```
└─# python decrypt.py
Decrypted file saved as: urgent_data.txt

┌──(root💀kali)-[/media/…/CTF/C1b3rwall/2024/Amenaza Interna]
└─# cat urgent_data.txt
CRY193WW484H
```

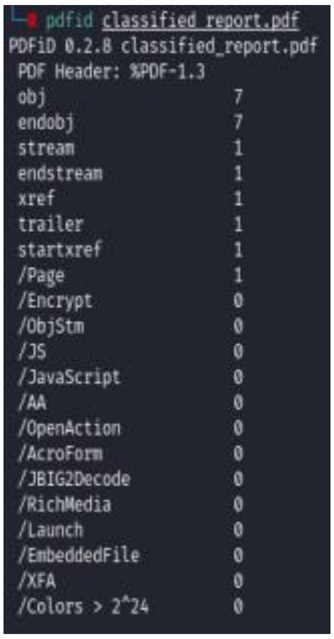
Según el log y la captura del tráfico de red se ha propagado un ransomware por la red encriptado los ficheros. La infección empieza por la dirección ip 192.168.1.10



```
└─# cat wannacry_logs.txt | head
2024-06-07 12:22:15 - 192.168.1.10 - Initial infection by WannaCry ransomware
2024-06-07 12:22:15 - Source IP: 192.168.1.10, Destination IP: 192.168.1.152, Action: SMB connection established
2024-06-07 12:22:15 - Source IP: 192.168.1.10, Destination IP: 192.168.1.136, Action: SMB connection established
2024-06-07 12:22:15 - Source IP: 192.168.1.152, Destination IP: 192.168.1.253, Action: SMB connection established
2024-06-07 12:22:15 - Source IP: 192.168.1.253, Destination IP: 192.168.1.66, Action: SMB connection established
2024-06-07 12:22:15 - Source IP: 192.168.1.253, Destination IP: 192.168.1.14, Action: File encrypted
2024-06-07 12:22:15 - Source IP: 192.168.1.152, Destination IP: 192.168.1.243, Action: SMB connection established
2024-06-07 12:22:15 - Source IP: 192.168.1.66, Destination IP: 192.168.1.29, Action: SMB connection established
2024-06-07 12:22:15 - Source IP: 192.168.1.66, Destination IP: 192.168.1.22, Action: SMB connection established
2024-06-07 12:22:15 - Source IP: 192.168.1.136, Destination IP: 192.168.1.135, Action: SMB connection established
```

Reto: Crimen de Guerra
El Crimen de Guerra te desafía a desentrañar los secretos de un Archivo. Debes utilizar tus habilidades de análisis de archivos y tu conocimiento de los metadatos.

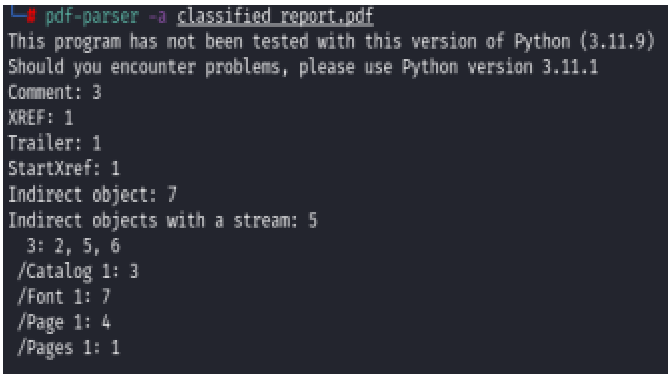
$ pdfid classified_report.pdf



```
└─# pdfid classified_report.pdf
PDFiD 0.2.8 classified_report.pdf
 PDF Header: %PDF-1.3
 obj                    7
 endobj                 7
 stream                 1
 endstream              1
 xref                   1
 trailer                1
 startxref              1
 /Page                  1
 /Encrypt               0
 /ObjStm                0
 /JS                    0
 /JavaScript            0
 /AA                    0
 /OpenAction            0
 /AcroForm              0
 /JBIG2Decode           0
 /RichMedia             0
 /Launch                0
 /EmbeddedFile          0
 /XFA                   0
 /Colors > 2^24         0
```

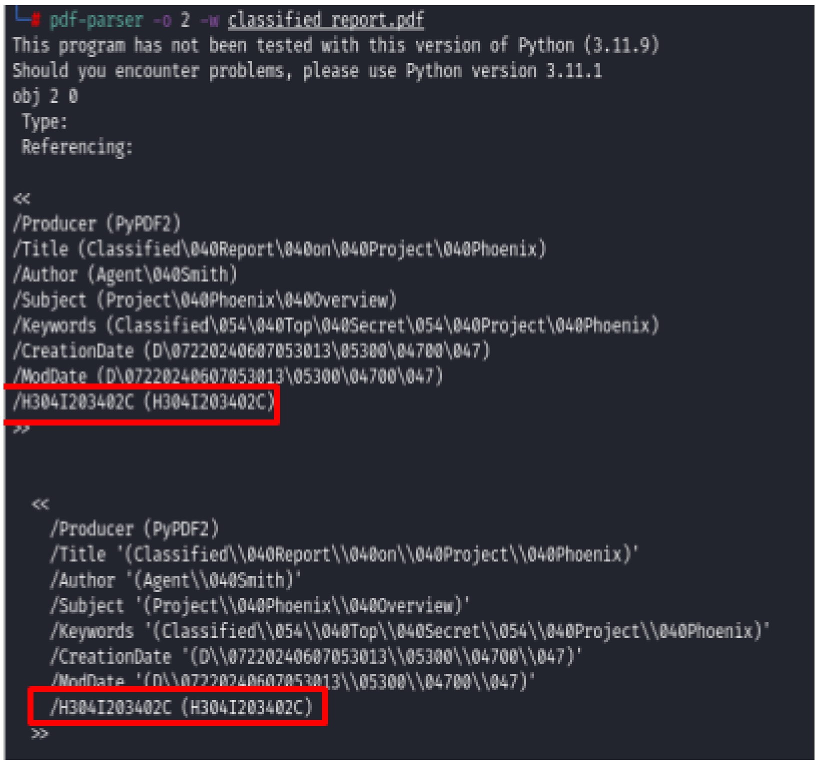
Tenemos un fichero pdf con 7 objetos, de los cuales sólo uno tiene stream. No hay código javascript ni OpenAction ni formularios.

Veamos que nos dice pdf-parser:
$ pdf-parser -a classified_report.pdf



```
└─# pdf-parser -a classified_report.pdf
This program has not been tested with this version of Python (3.11.9)
Should you encounter problems, please use Python version 3.11.1
Comment: 3
XREF: 1
Trailer: 1
StartXref: 1
Indirect object: 7
Indirect objects with a stream: 5
  3: 2, 5, 6
 /Catalog 1: 3
 /Font 1: 7
 /Page 1: 4
 /Pages 1: 1
```

$ pdf-parser -o 2 -w classified_report.pdf



```
└─# pdf-parser -o 2 -w classified_report.pdf
This program has not been tested with this version of Python (3.11.9)
Should you encounter problems, please use Python version 3.11.1
obj 2 0
 Type:
 Referencing:

<<
/Producer (PyPDF2)
/Title (Classified\040Report\040on\040Project\040Phoenix)
/Author (Agent\040Smith)
/Subject (Project\040Phoenix\040Overview)
/Keywords (Classified\054\040Top\040Secret\054\040Project\040Phoenix)
/CreationDate (D\07220240607053013\05300\04700\047)
/ModDate (D\07220240607053013\05300\04700\047)
/H304I203402C (H304I203402C)
>>


  <<
    /Producer (PyPDF2)
    /Title '(Classified\\040Report\\040on\\040Project\\040Phoenix)'
    /Author '(Agent\\040Smith)'
    /Subject '(Project\\040Phoenix\\040Overview)'
    /Keywords '(Classified\\054\\040Top\\040Secret\\054\\040Project\\040Phoenix)'
    /CreationDate '(D\\07220240607053013\\05300\\04700\\047)'
    /ModDate '(D\\07220240607053013\\05300\\04700\\047)'
    /H304I203402C (H304I203402C)
  >>
```

Fuente:

https://c1b3rwall.hackrocks.com