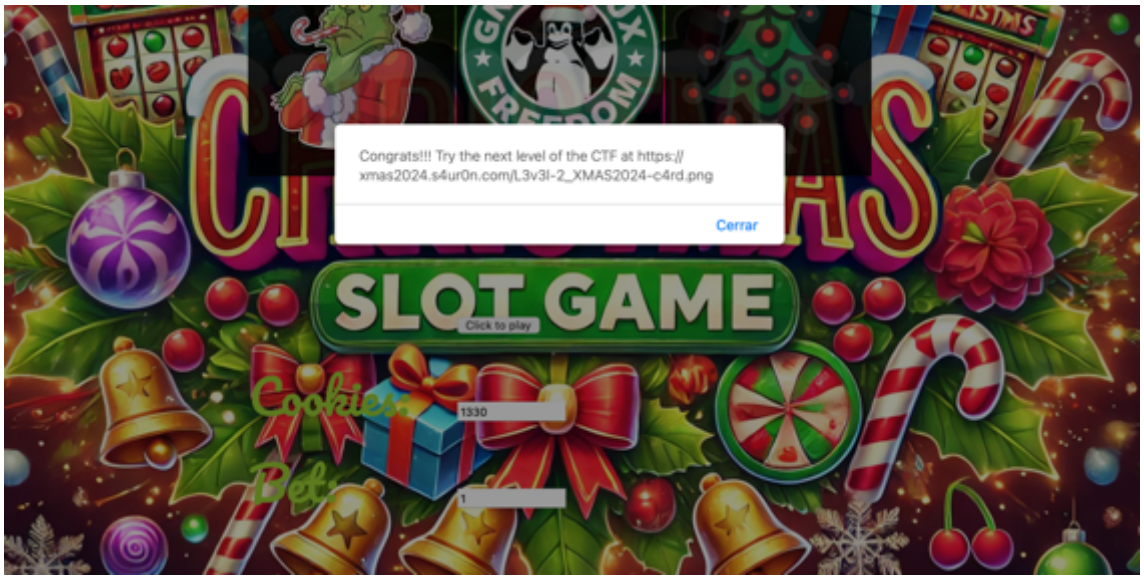
# CTF s4ur0n - Xmas 2024

Writeup

@oreos_es

**Challenge 1:**

Introduce un número elevado de cookies (p.e. 1337) y apuesta 1 en cada jugada, hasta obtener el enlace al siguiente reto.




https://xmas2024.s4ur0n.com/L3v3l-2_XMAS2024-c4rd.png

**Challenge 2:**

Disponemos de una clave privada parcialmente filtrada, pero recuperable.


Utilizamos el siguiente script para regenerar la clave:

```
from Crypto.PublicKey import RSA
from Crypto.Util.number import isPrime

q =
0x00e781ed2c4a257b1a0c39ba0ccdf00ed81ab6c5121e758b81154cfb0f2f8d46cf0365f5a4e
2e86f7ab602706c2ac48bc84a8f3e726dc58fa90965eb19c51d63c4f7010bb8aab746d5c7e785
c3bdbebe090f5d952c32a0734565b86e965e69d4c696834894a56b11240b8a1bd6e4822e6555c
174f291bfb514df002ef67b82ab195f1fb67880c0f10e3288909621d87d730f14078639034ff9
0cb887cf0543c88552236515cb0948b06edfd7232a2cb9e37a5cb019df4cf517e1035be592ad3
100da643e37f5b2525e8c0f2e77150b3a1bc007e0fc81ec5933713bad691e52100019e03420aa
d2895abbc3d059c086a20de025f840c1e98c7460bd8e4b5b5b774f
dp =
0x0091b5b9d3287055ad70d4c424ce308730294f351fb330bacad6adfcce9edee9b5b5785ee31
7cb11c248e225d008103279b13259e8049cef0ea34551a7b94b8cffc8b166dc10b068fc35109e
c977ef46448e1338f56a0d9b00bcdfbe5a38cbca4713c5cd44fe07adf89afa545fa5e03eb04c5
4e7c9c7d592b15d78f8d77a77b3722fb3d956d16c90096fa436eea5181b2e8f831b6d9d2c7d82
95f10eba53a9e4488fffd79d62ddbba8643ca1ac6f690b6fa24bafad202d09fff30aa3cb7e3fd
bdc23f8392a9230e622dde262cfb83869b678dc7537a4166bc88c175b5a2cbec2a51dc4edd421
1ab9fa6394b183ab012a971ba812888b873d1a41555d758d25bc05

e = 65537

for e in range(65535,65539):
  for kp in range(3, e):
    p_mul = dp * e - 1
    if p_mul % kp == 0:
        p = (p_mul // kp) + 1
        if isPrime(p):
            #print(f"Possible p: {p}")
            try:
```

```
            N = p*q
            #print(N)

            phi = (p-1)*(q-1)
            d = pow(e,-1,phi)

            key = RSA.construct((N,e,d,p,q))
            pem = key.export_key('PEM')
            print(pem.decode('utf-8'))
          except:
            pass
```

Obtenemos la clave privada:

```
-----BEGIN RSA PRIVATE KEY-----
MIIJKwIBAAKCAgEA5D97FNEntF8wroWxNTux9+Gld187wWQXP987Bi6PsLMhy2cA
k0GDhPzSTBVnWOAfed0Fz+1O/+Flnqjcarvmsua/Xmeq56xSXg0yQasqZae7+UBa
jMVIGdJdCXSLZDLa7KRYWFGWT++gJWSC51J+gbnCNyPlaSx8JrTGDEBlsmHRyuUJ
jNA7hb67STnPLBbzgPDf3+aDsFOApA8ggHKr7FcHkU1U9/6Kzj44qFxyJZjUb4aj
OG4nTnwQjVe45EDq1WMFrQQmtuRSbVs2OseP4fsUw4CWBmSOnG3pNYHsxsmd/vO6
COkx9nEGku5STeSaqJjJqDNSTEiqxW9moiDVWNDUMS+9A+99EoZAzTkjcIh0YMF9
AxqkI6RnrDJqPeC9i7CM6mshzqVPKAzc8MzuEkIYmFy2M5FJynTommCamP+JwVLn
eFj+Yhrc5+O1tSrtuoeWQK3f1NzlvWb3AiXeK6RjSVyZQDHLWxqecW39J5eOrmtL
DaL4L3vcl9Cu+G1m1uBnu5F1uEWI2+z8LwjsTzSTqnLRdvYoBBg45mta4mxM3O3v
l11BlZnRIDjxldJ5xpR5seikfDdrCHv64yvRRIZN5jRf1kvz7DgKxVG1WDjf+INv
I4Q+v+6I+nwqByFqqQeNKbIWAcId2cha4hDIk9LQEMDzZiL3uNNcaFZji4MCAwEA
AQKCAgEA1Ar1eT1luXfFbhzdqCqxBywl4GQky2EFCF2GJBQVgX6pIqGqMyNl36JQ
bEZmIHb2RuxCfgxkm+r10RPm0XGGvSUJG9cLOvcn/iAcVE2DsbTGOKTEeoq8lOCN
dj9DT+6+26FCQapqDhD7okFiKyzEQhgkib1bXv3oyLygULlywOmHUQq+eIbrBTFQ
JJMEGF2qElu0X/ly1dh9Zex3sVzWw1WGvkItccaThU7gq+hWUv9MO9/EuqP6+Drh
1a1tIv/8Kgk4OKfmn3o16UoXczv6O2Jaw9UtivrYUhL52K+/HF4p3bTnW2fo9p3C
EbY92AdMdtyaWxxylFPd8lWv72a5S2wIL1XiUdFgJVbojVnj1v7VmG+R68AnrMqc
03kXCkHmxaWf76XI4e/CpxA8Ms46Dw8yvgfqHD1xBeK8j3jwgSWYu6G11zDKaxYb
FG0UQB4G5uC89wC4GHb3Yl5AU0jKpdkPpQ+7VjMnuf4YM0dYuuW5ljOma+vkvLJo
KrhDT9zJvixYqKjJV8FEwNRLjLPvMqUQt0OzkMvpJsI/lEgfXbZtXNdOqCNsfUS9
OqlNkIzWipDRdMu+H7eoPMMuT3OBx2TYnYAkTLNY5FOb+H4zL3hJ2gOntrDRAoYX
v4ONlrx/ityx1ff/3f9V31t2rxBoGIh/WEyp1Xla9CZ//aU6cyECggEBAPxlRg6C
MTRqmTc5yZMj+MdBS+nFt8e+2hXVWxWP0BldL4UL8FIMXZl7iKl+vL0QvuLtA8H7
/L/+0yxpuPYKvysXjDagemruYM+6hf+1CQjGTob6xiYdQDGHJR77d7b/7nLiAywF
55nb6ldELt1SiAuarV1tBYYMl23djGwliI/jw/vduhHKfTPArejIvY3UIeBb4MyP
fDZQHzQSEz7rqsoOCS+YW7eJjSZBoboH9BxXbmnS0hcWY/93O0jW2tsXS80DIH1w
/8PxAZXC8Tx/aFIh3U2oFqrRtI5HitHEXmzeH9QgBmFN1tQM51uqGXy7iNJJ42/r
pU9RJnahea5am40CggEBAOeB7SxKJXsaDDm6DM3wDtgatsUSHnWLgRVM+w8vjUbP
A2X1pOLob3q2AnBsKsSLyEqPPnJtxY+pCWXrGcUdY8T3AQu4qrdG1cfnhcO9vr4J
D12VLDKgc0VluG6WXmnUxpaDSJSlaxEkC4ob1uSCLmVVwXTykb+1FN8ALvZ7gqsZ
Xx+2eIDA8Q4yiJCWIdh9cw8UB4Y5A0/5DLiHzwVDyIVSI2UVywlIsG7f1yMqLLnj
elywGd9M9RfhA1vlkq0xANpkPjf1slJejA8udxULOhvAB+D8gexZM3E7rWkeUhAA
GeA0IKrSiVq7w9BZwIaiDeAl+EDB6Yx0YL2OS1tbd08CggEBAJG1udMocFWtcNTE
JM4whzApTzUfszC6ytat/M6e3um1tXhe4xfLEcJI4iXQCBAyebEyWegEnO8Oo0VR
p7lLjP/IsWbcELBo/DUQnsl370ZEjhM49WoNmwC8375aOMvKRxPFzUT+B634mvpU
X6XgPrBMVOfJx9WSsV14+Nd6d7NyL7PZVtFskAlvpDbupRgbLo+DG22dLH2ClfEO
ulOp5EiP/9edYt27qGQ8oaxvaQtvokuvrSAtCf/zCqPLfj/b3CP4OSqSMOYi3eJi
z7g4abZ43HU3pBZryIwXW1osvsKlHcTt1CEaufpjlLGDqwEqlxuoEoiLhz0aQVVd
dY0lvAUCggEBALG6rGMpFTc5mxMiQzxCxJKhh5kpvNqO2+2HaOKSpgorWTeIayqM
OTFi0+KNGBRGH+ElsVJV9arBoeZtpB4Q3wxSeKoP/nev2OWcV7QbUnlAKVy17fV7
+qLXYcz8gcULxd29MhZ0HAtPudAwaTyKuKWxPVDT/JLJqRk+Yc92qK1EUCPfiQmH
lkhJAVDHAXrbbF6yCMjBskpOL7bnBEbNb/7yPRwYrAQXmuOz0s07TpTzD3hi9anZ
wfuwEk0VpRJzIW2IMb/yTxEvZqUtDdzI/rZZKXNPR0s0e+q9XvbpgSSpfzQBsOaT
tUFEDyNAFC8H8FEZtUm51NuwaKh9ulqLkL8CggEBAINxIsHvYOPaMchRz2vwBhvj
```

```
SWFWxrWUYmUoV7No5EbJhpQl4CEN5FX6oS6yj4X+5aKT3P/W5WdQ0dph/K4vhrD2
ChbmXZ/y63xghCUjAU8rEfTIsu/6u36qxf7D4UAHwHPvB5Jx0to7HhkRFfXw1f16
NEZe+vhMIFRFvaipj95UNHuMnbmuaTBFxmgfRsbsg9AQorr5Ky/SAKROrXUfbyx4
wLerPPfMvrv3EmP8UJtSmXNkncStfBrzgOY0yZUMmh5ML8wK5YfrlQ0nc769okCl
VtqnfOK0g+1YGqdX1m5dxlqTVYGAQLQboLyhJm9xnoLDfqlHuQ+1eJhGyaZ5qI8=
-----END RSA PRIVATE KEY-----
```

**Challenge 3:**

Accedemos a la máquina usando la clave privada, usando el protocolo “sftp”:

```
$ sftp -i test.key santa@xmas2024.s4ur0n.com
Connected to xmas2024.s4ur0n.com.
sftp> get *
Fetching /home/santa/README to README
README
100%  415     3.1KB/s   00:00
Fetching /home/santa/grinch to grinch
grinch
100% 8354    60.9KB/s   00:00
sftp>exit

$ cat README
The Grinch intercepted a message between s4ur0n and Santa, obtaining a file,
but we don't know what it contains or how to open it...

We know they used analog and old-school communications, but not much more...

In 1964, Xerox Corporation introduced (and patented) what many consider to be
the first commercialized version of the original machine used to send
messages between Santa and s4ur0n under the name LDX...

$ mv grinch grinch.g3
```

Usamos la herramienta https://convertio.co/es/fax-png/ para convertir el fichero de FAX a PNG.


https://xmas2024.s4ur0n.com/l3v3l_four/XM24g4me

**Challenge 4:**

Descargamos el fichero “XM24g4me” y analizamos el binario con binwalk.

```
$ sudo binwalk XM24g4me

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
---
0             0x0             Gameboy ROM,, [ROM ONLY], ROM: 256Kbit
32768         0x8000          GIF image data, version "89a", 465 x 465
```

Extraemos la imagen GIF:

Connect by ssh (user: level6, private key from #2)

**Challenge 5:**

Usamos la herramienta “scp” para descargar todas las imágenes del servidor:

```
$ scp -O -i test.key level6@xmas2024.s4ur0n.com:*.png .
```

Observamos que se trata de tarjetas perforadas.


Usamos una herramienta online para decodificar el mensaje oculto en algunas tarjetas, y obtenemos la siguiente URL:

HTTPS://XMAS2024.S4UR0N.COM/L3V3L7XMAS24CTFR3V

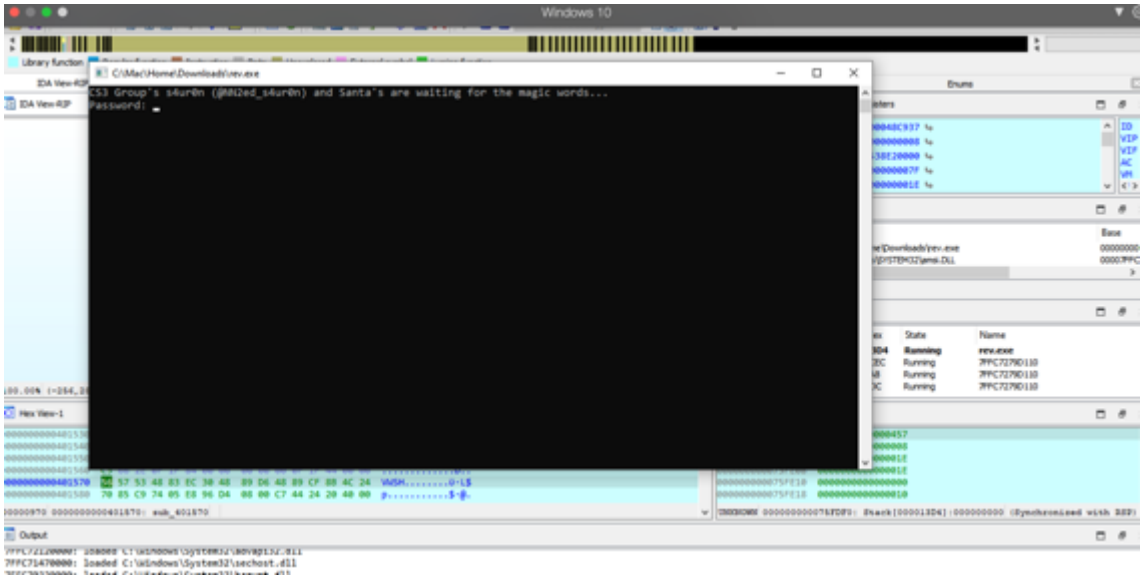
## Challenge 6:

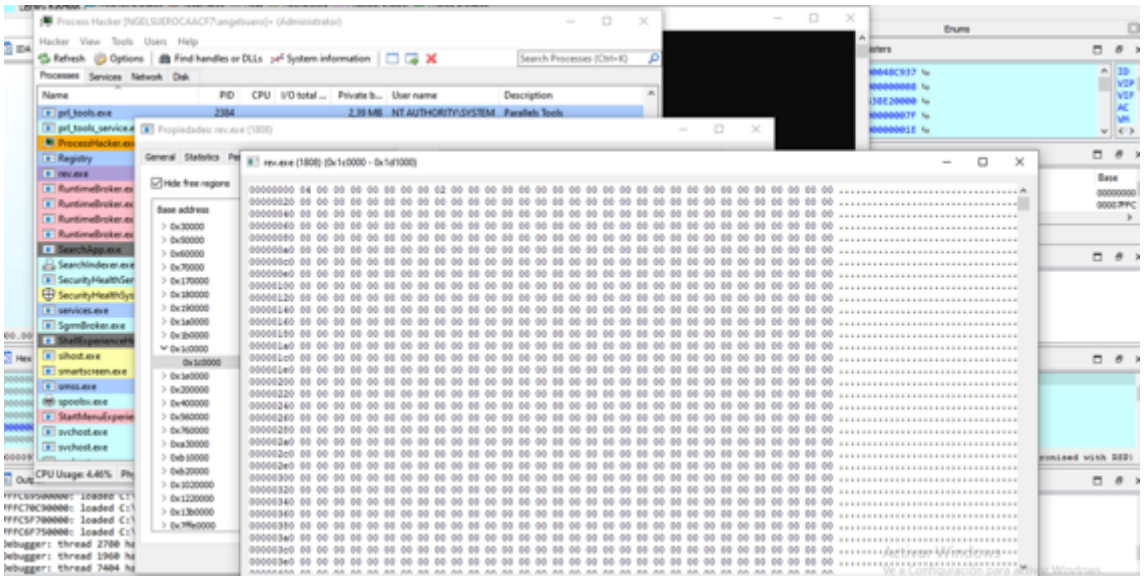
Descargamos el binario y observamos que se trata de un fichero PE32+.

Tras realizar un análisis del mismo, observamos que emplea inyección de shellcode mediante APC a través de NtTestAlert (ntdll):
https://cocomelonc.github.io/tutorial/2021/11/20/malware-injection-4.html

```
__int64 __fastcall sub_401570(LPCVOID lpBuffer, SIZE_T nSize, __int64 a3, __int64 a4, int a5)
{
  void (__stdcall *v7)(ULONG_PTR); // rax
  void (__stdcall *v8)(ULONG_PTR); // rbx
  BOOL v9; // eax
  unsigned int v10; // esi
  HANDLE CurrentThread; // rax
  HMODULE ModuleHandleA; // rax
  void (*NtTestAlert)(void); // rsi
  SIZE_T NumberOfBytesWritten[4]; // [rsp+28h] [rbp-20h] BYREF

  if ( a5 )
    sub_48EA20();
  v7 = (void (__stdcall *)(ULONG_PTR))VirtualAllocEx((HANDLE)0xFFFFFFFFFFFFFFFFi64, 0i64, nSize, 0x3000u, 0x40u);
  if ( v7 )
  {
    v8 = v7;
    NumberOfBytesWritten[0] = 0i64;
    v9 = WriteProcessMemory((HANDLE)0xFFFFFFFFFFFFFFFFi64, v7, lpBuffer, nSize, NumberOfBytesWritten);
    v10 = 2;
    if ( v9 )
    {
      CurrentThread = GetCurrentThread();
      if ( QueueUserAPC(v8, CurrentThread, 0i64) )
      {
        ModuleHandleA = GetModuleHandleA("ntdll.dll");
        NtTestAlert = (void (*)(void))GetProcAddress(ModuleHandleA, "NtTestAlert");
        NtTestAlert();
        return (unsigned int)(NtTestAlert == 0i64) + 1;
      }
    }
  }
  else
  {
    return 2;
```

Podemos parar la ejecución del binario justo cuando se va a escribir el shellcode en memoria (WriteProcessMemory), pero optamos por otra opción. Extraer el código inyectado directamente desde memoria con Process Hacker:



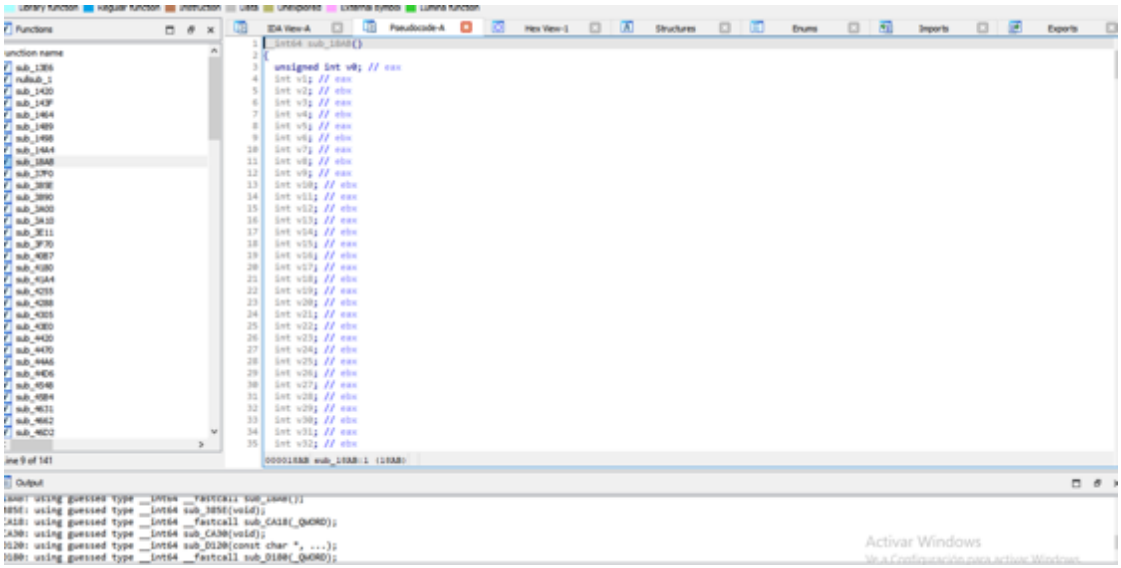


Guardamos dump en un nuevo binario y lo analizamos con radare2:

```
$ r2 -AAA shellcode.bin
INFO: Analyze all flags starting with sym. and entry0 (aa)
INFO: Analyze imports (af@@@i)
INFO: Analyze symbols (af@@@s)
INFO: Analyze all functions arguments/locals (afva@@@F)
INFO: Analyze function calls (aac)
INFO: find and analyze function preludes (aap)
INFO: Analyze len bytes of instructions for references (aar)
INFO: Finding and parsing C++ vtables (avrr)
INFO: Analyzing methods (af @@ method.*)
INFO: Recovering local variables (afva@@@F)
INFO: Type matching analysis for all functions (aaft)
INFO: Propagate noreturn information (aanr)
INFO: Scanning for strings constructed in code (/azs)
INFO: Enable anal.types.constraint for experimental type propagation
INFO: Reanalizing graph references to adjust functions count (aarr)
INFO: Autoname all functions (.afna@@c:afla)
[0x00000000]> izz | grep -C3 s4ur0n
753 0x0000d16d 0x0000d16d 4   6            utf8    8[]Ð
754 0x0000d188 0x0000d188 4   5            ascii   l$0H
755 0x0000d1cd 0x0000d1cd 4   6            utf8    8[]Ð
756 0x0000f000 0x0000f000 82  83           ascii   CS3 Group's s4ur0n
(@NN2ed_s4ur0n) and Santa's are waiting for the magic words...\n
757 0x0000f053 0x0000f053 10  11           ascii   Password:
758 0x0000f061 0x0000f061 17  18           ascii   Congratulations!\n
759 0x0000f078 0x0000f078 81  82           ascii   Next level is waiting
you... ssh level8@xmas2024.s4ur0n.com (use this password)\n\n
760 0x0000f0d0 0x0000f0d0 66  67           ascii   Opsss! The Grinch has
obfuscated the code of this binary a bit...\n
761 0x0000f113 0x0000f113 14  15           ascii   Try again...\n\n
762 0x0000f1a0 0x0000f1a0 30  31           ascii   Argument domain error
(DOMAIN)
[0x00000000]> axt @ 0x0000f000
sub.sub.qword_385e_18ab 0x18fc [DATA:r--] lea rax, [0x0000f000]
[0x00000000]> pdf @ sub.sub.qword_385e_18ab
Do you want to print 2382 lines? (y/N)
```

Se observa que en la posición de memoria con offset 0x18ab se encuentra la función que comprueba la entrada en el binario:

Abrimos el binario en IDA para obtener una aproximación del código original con el decompilador:



Observando el código fuente observamos que compara carácter a carácter de la entrada del usuario con un valor calculado a partir de una función (sub_1498), para cada carácter.

Reescribimos el código para imprimir los caracteres usados para validar cada carácter de la entrada del usuario:

```
#include <stdio.h>

void sub_1498(int a) {
  printf("%c", (char)a);
}

int main()
{
  int v75 = 0x48;
  int v76 = 0x48;
  sub_1498(
                      194 * (v75 & ~v76)
                    + 159 * v75
                    + 97 * v76
                    + 72
                    + 159 * (v75 ^ v76)
                    + (v76 + 72 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                    * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 192 + 8 *
(v75 ^ v76)));
  sub_1498(
                      194 * (v75 & ~v76)
                    + 159 * v75
                    + 97 * v76
                    + 180
                    + 159 * (v75 ^ v76)
                    + (v76 + 52 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                    * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 96 + 8 *
(v75 ^ v76)));
  sub_1498(
```

```
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 146
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 112 + 8 *
(v75 ^ v76))
                  * (v76 + 210 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 228
                  + 159 * (v75 ^ v76)
                  + (v76 + 100 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76))
                  * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 224 + 8 *
(v75 ^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 114
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 112 + 8 *
(v75 ^ v76))
                  * (v76 + 178 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 173
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 152 + 8 *
(v75 ^ v76))
                  * (v76 + 205 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 150
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 80 + 8 *
(v75 ^ v76))
                  * (v76 + 86 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 251
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 40 + 8 *
(v75 ^ v76))
                  * (v76 + 219 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76)));
```

```
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 146
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 112 + 8 *
(v75 ^ v76))
                  * (v76 + 210 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 59
                  + 159 * (v75 ^ v76)
                  + (v76 + 27 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                  * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 40 + 8 *
(v75 ^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 185
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 56 + 8 *
(v75 ^ v76))
                  * (v76 + 89 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 142
                  + 159 * (v75 ^ v76)
                  + (v76 + 78 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                  * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 144 + 8 *
(v75 ^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 111
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 136 + 8 *
(v75 ^ v76))
                  * (v76 + 207 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 95
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 8 + 8 *
(v75 ^ v76))
```

```
                    * (v76 + 191 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76)));
  sub_1498(
                      194 * (v75 & ~v76)
                    + 159 * v75
                    + 97 * v76
                    + 185
                    + 159 * (v75 ^ v76)
                    + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 56 + 8 *
(v75 ^ v76))
                    * (v76 + 89 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76)));
  sub_1498(
                      194 * (v75 & ~v76)
                    + 159 * v75
                    + 97 * v76
                    + 244
                    + 159 * (v75 ^ v76)
                    + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 96 + 8 *
(v75 ^ v76))
                    * (v76 + 116 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76)));
  sub_1498(
                      194 * (v75 & ~v76)
                    + 159 * v75
                    + 97 * v76
                    + 104
                    + 159 * (v75 ^ v76)
                    + (v76 + 104 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76))
                    * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 192 + 8 *
(v75 ^ v76)));
  sub_1498(
                      194 * (v75 & ~v76)
                    + 159 * v75
                    + 97 * v76
                    + 21
                    + 159 * (v75 ^ v76)
                    + (v76 + 53 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                    * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 88 + 8 *
(v75 ^ v76)));
  sub_1498(
                      194 * (v75 & ~v76)
                    + 159 * v75
                    + 97 * v76
                    + 98
                    + 159 * (v75 ^ v76)
                    + (v76 + 162 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76))
                    * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 240 + 8 *
(v75 ^ v76)));
  sub_1498(
                      194 * (v75 & ~v76)
                    + 159 * v75
                    + 97 * v76
                    + 180
                    + 159 * (v75 ^ v76)
```

```
                  + (v76 + 52 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                  * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 96 + 8 *
(v75 ^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 59
                  + 159 * (v75 ^ v76)
                  + (v76 + 27 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                  * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 40 + 8 *
(v75 ^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 201
                  + 159 * (v75 ^ v76)
                  + (v76 + 105 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76))
                  * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 184 + 8 *
(v75 ^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 110
                  + 159 * (v75 ^ v76)
                  + (v76 + 46 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                  * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 144 + 8 *
(v75 ^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 196
                  + 159 * (v75 ^ v76)
                  + (v76 + 68 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                  * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 224 + 8 *
(v75 ^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 21
                  + 159 * (v75 ^ v76)
                  + (v76 + 53 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                  * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 88 + 8 *
(v75 ^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 173
```

```
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 152 + 8 *
(v75 ^ v76))
                  * (v76 + 205 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 146
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 112 + 8 *
(v75 ^ v76))
                  * (v76 + 210 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 146
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 112 + 8 *
(v75 ^ v76))
                  * (v76 + 210 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 129
                  + 159 * (v75 ^ v76)
                  + 8
                  * (30 * (v75 & ~v76) + 31 * (v76 + 1) + v75 + (v75 ^
v76))
                  * (v76 + 33 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 88
                  + 159 * (v75 ^ v76)
                  + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 64 + 8 *
(v75 ^ v76))
                  * (v76 + 88 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
                  + 97 * v76
                  + 53
                  + 159 * (v75 ^ v76)
                  + (v76 + 85 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                  * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 88 + 8 *
(v75 ^ v76)));
  sub_1498(
                    194 * (v75 & ~v76)
                  + 159 * v75
```

```
                    + 97 * v76
                    + 233
                    + 159 * (v75 ^ v76)
                    + (v76 + 137 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76))
                    * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 184 + 8 *
(v75 ^ v76)));
  sub_1498(
                      194 * (v75 & ~v76)
                    + 159 * v75
                    + 97 * v76
                    + 59
                    + 159 * (v75 ^ v76)
                    + (v76 + 27 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75 ^
v76))
                    * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 40 + 8 *
(v75 ^ v76)));
  sub_1498(
                       194 * (v75 & ~v76)
                     + 159 * v75
                     + 97 * v76
                     + 180
                     + 159 * (v75 ^ v76)
                     + (v76 + 52 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76))
                     * (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 96 + 8 *
(v75 ^ v76)));
  sub_1498(
                       194 * (v75 & ~v76)
                     + 159 * v75
                     + 97 * v76
                     + 189
                     + 159 * (v75 ^ v76)
                     + (240 * (v75 & ~v76) + 8 * v75 + 248 * v76 + 24 + 8 *
(v75 ^ v76))
                     * (v76 + 221 + 255 * v75 + 2 * (v75 & ~v76) + 255 * (v75
^ v76)));
}
```

Tras ejecutarlo, obtenemos la entrada válida:

```
H4rdRev3rs1ngW1thMB4sANDMerryXmas4u
```

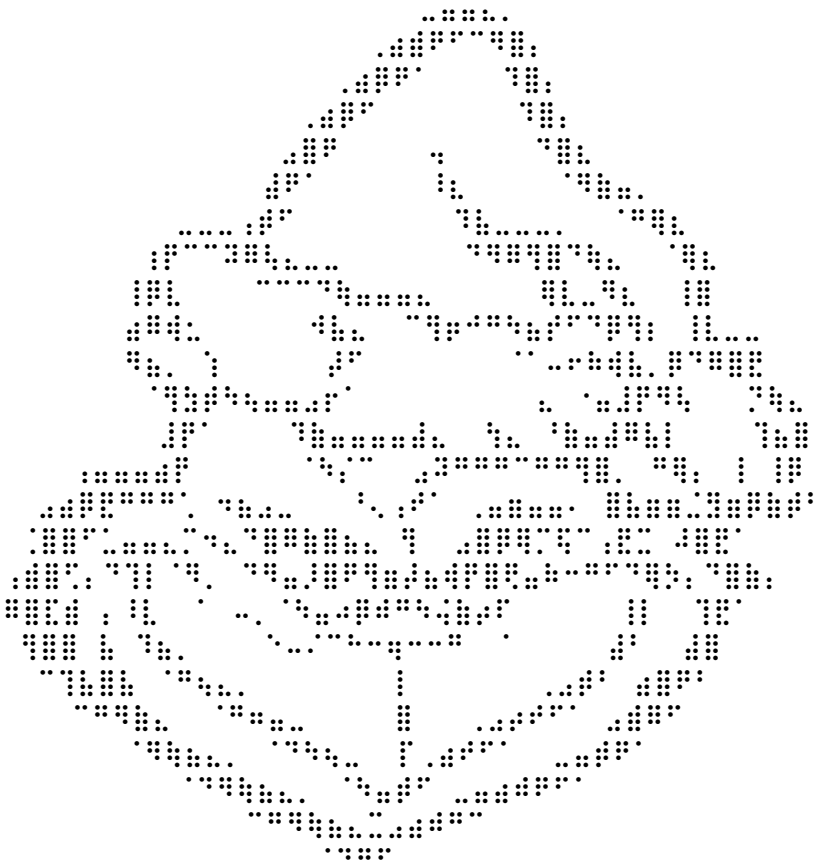
La introducimos en el binario:

```
Congratulations!
Next level is waiting you... ssh level8@xmas2024.s4ur0n.com (use this
password)
```

## Challenge 7:

Accedemos por ssh al reto usando la clave anterior (H4rdRev3rs1ngW1thMB4sANDMerryXmas4u):

```
$ ssh level8@xmas2024.s4ur0n.com
level8@xmas2024.s4ur0n.com's password:
Linux xmas2024 6.1.0-28-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.119-1 (2024-
11-22) x86_64
```



```
  You are in my house... Merry XMAS!!!
Last login: Fri Jan  3 19:15:17 2025 from 152.204.183.66
-bash-5.2$ ls
README
-bash-5.2$ cat README
Welcome home again... but Santa hasn't left any gifts yet.

Are you waiting for the exploiting levels?

It's time to fuck around a bit more before we get to them...

Level 9 is located at
https://xmas2024.s4ur0n.com/ea6a958571c0450486dd1c842326fefe1a4fc03c43c60efcb
bae72966c91e8cf
```

**Challenge 8:**

Accedemos a la URL indicada en el reto anterior:
https://xmas2024.s4ur0n.com/ea6a958571c0450486dd1c842326fefe1a4fc03c43c60efcbbae72966c91e8cf


Tras revisar el código fuente de la página, observamos que existe un directorio “assets” que permite listar su contenido:

**Index of /ea6a958571c0450486dd1c842326fefe1a4fc03c43c60efcbbae72966c91e8cf/assets**

| Name | Last modified | Size | Description |
|---|---|---|---|
| Parent Directory | | - | |
| frame-01.png | 2024-12-21 21:41 | 1.5M | |
| frame-02.png | 2024-12-21 21:41 | 1.5M | |
| frame-03.png | 2024-12-21 21:41 | 1.5M | |
| xmas2024.mp4 | 2024-12-21 21:41 | 33M | |

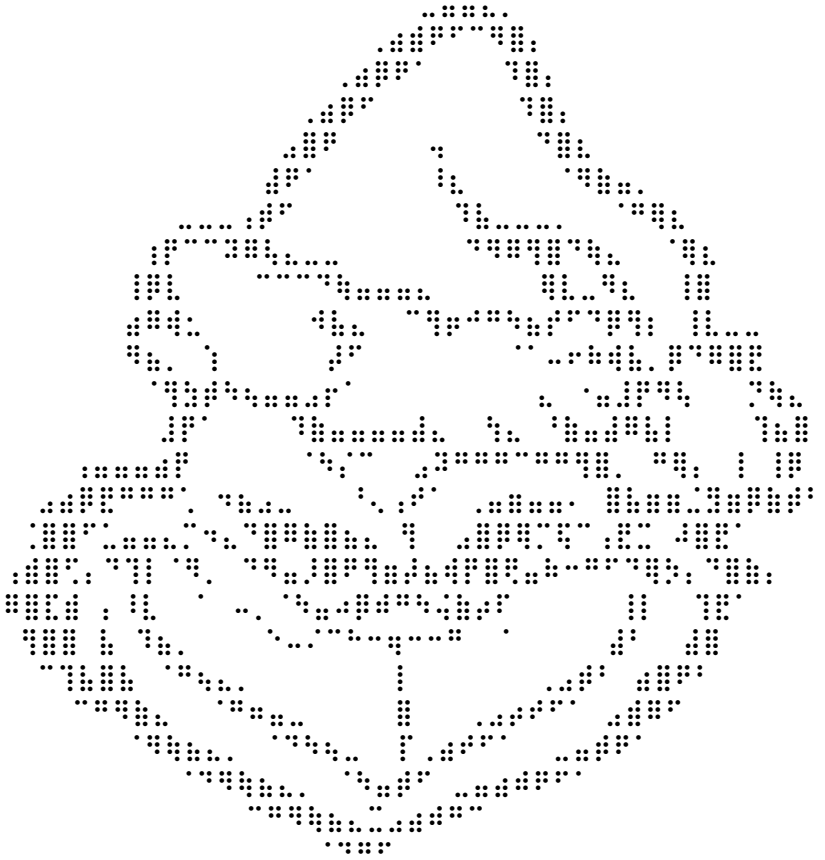
*Apache/2.4.62 (Debian) Server at xmas2024.s4ur0n.com Port 443*

Observamos que una de las imágenes nos indica el acceso al siguiente reto:

## C**hallenge 9:**

Accedemos al servidor usando el usuario “level9” y la clave del reto 2:

```
$ ssh -i test.key level9@xmas2024.s4ur0n.com
Linux xmas2024 6.1.0-28-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.119-1 (2024-
11-22) x86_64
```



```
  You are in my house... Merry XMAS!!!
Last login: Fri Jan  3 19:58:55 2025 from 152.204.183.66
-bash-5.2$ ls
README
-bash-5.2$ cat README
Did you remember old BBS's time?

Connect to https://xmas2024.s4ur0n.com/38d448ea401fbe9be522afd88e8b3cfd.html
and enjoy it!
```

**Challenge 10:**

Descargamos el fichero desde la web:
https://xmas2024.s4ur0n.com/38d448ea401fbe9be522afd88e8b3cfd.html


Se trata de una comunicación modem, usamos la herramienta "minimodem" para decodificar el mensaje, sabiendo la velocidad de la comunicación a partir del fondo de la web (300bps):

```
<!DOCTYPE html>
<html>
<head>
<title>XMAS CTF by s4ur0n (CS3 Group)</title>
<link rel="icon" href="favicon.ico" type="image/x-icon">
<style>
body {
  background-image: url('modem300bps.png');
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: 100% 100%;
}
</style>
</head>
<body>
  <audio id="range" src="santa.wav" type="audio/wav" controls="controls"></audio> 
  <a href="santa.wav" download="santa.wav"><img src="download.png" height="54px"/></a>
</body>
</html>
```

```
$ minimodem -r -f santa.wav 300
### CARRIER 300 @ 1250.0 Hz ###
SG8hIEhvISBIbyEgQ29uZ3JhdHMgbXkgZnJpZW5kISBJIHdpbGwgaWdub3JlIHlvdXIgR29vZ2xlI
GFuZCBQb3JuaHViIGhpc3Rvcnkgd2hlbiByZWFkaW5nIHlvdXIgbGV0dGVyLiBJJ20gdmVyeSBidX
N5IHJpZ2h0IG5vdyBiZWNhdXNlIEkgaGF2ZSB0byBmZWVkIFJ1ZG9scGgsIERvbm5lciwgQmxpdHp
lbiwgVml4ZW4sIEN1cGlkLCBDb21ldCwgRGFzaGVyLCBEYW5jZXIgYW5kIFByYW5jZXIuIEl0IG9u
bHkgcmVtYWlucyBmb3IgbWUgdG8gY29udmV5IG15IGNvbmdyYXR1bGF0aW9ucyBhbmQgd2lzaCB5b
3UgZXZlcnkgc3VjY2VzcyBmb3IgeW91ciBuZXh0IGxldmVsIHRoYXQgaXMgYXQgaHR0cHM6Ly94bW
FzMjAyNC5zNHVyMG4uY29tL2xldmVsMTAvZjc1MTcwYzM0MTJmMGRkMjYxMDNjMGNjNGY0MjhlMDY
3MTc1MjgzNi5odG1s

### NOCARRIER ndata=557 confidence=2.355 ampl=0.993 bps=300.00 (rate perfect)
###

$ echo -n
"SG8hIEhvISBIbyEgQ29uZ3JhdHMgbXkgZnJpZW5kISBJIHdpbGwgaWdub3JlIHlvdXIgR29vZ2xl
```

```
IGFuZCBQb3JuaHViIGhpc3Rvcnkgd2hlbiByZWFkaW5nIHlvdXIgbGV0dGVyLiBJJ20gdmVyeSBid
XN5IHJpZ2h0IG5vdyBiZWNhdXNlIEkgaGF2ZSB0byBmZWVkIFJ1ZG9scGgsIERvbm5lciwgQmxpdH
plbiwgVml4ZW4sIEN1cGlkLCBDb21ldCwgRGFzaGVyLCBEYW5jZXIgYW5kIFByYW5jZXIuIEl0IG9
ubHkgcmVtYWlucyBmb3IgbWUgdG8gY29udmV5IG15IGNvbmdyYXR1bGF0aW9ucyBhbmQgd2lzaCB5
b3UgZXZlcnkgc3VjY2VzcyBmb3IgeW91ciBuZXh0IGxldmVsIHRoYXQgaXMgYXQgaHR0cHM6Ly94b
WFzMjAyNC5zNHVyMG4uY29tL2xldmVsMTAvZjc1MTcwYzM0MTJmMGRkMjYxMDNjMGNjNGY0MjhlMD
Y3MTc1MjgzNi5odG1s" | base64 -d

Ho! Ho! Ho! Congrats my friend! I will ignore your Google and Pornhub history
when reading your letter. I'm very busy right now because I have to feed
Rudolph, Donner, Blitzen, Vixen, Cupid, Comet, Dasher, Dancer and Prancer. It
only remains for me to convey my congratulations and wish you every success
for your next level that is at
https://xmas2024.s4ur0n.com/level10/f75170c3412f0dd26103c0cc4f428e0671752836.
html
```

**Challenge 11:**

En este reto, tenemos que evadir el comentario en Java inyectando código que será ejecutado.


Revisando información, observamos que es posible inyectar código usando caracteres Unicode: https://stackoverflow.com/questions/30727515/why-is-executing-java-code-in-comments-with-certain-unicode-characters-allowed

Probamos el siguiente payload:

```
\u002a\u002f\u0053\u0079\u0073\u0074\u0065\u006d\u002e\u006f\u0075\u0074\u002e\u0070\u0072\u0069\u006e\u0074\u006c\u006e\u0028\u0022\u0042\u004c\u0055\u0045\u0020\u006f\u0072\u0020\u0052\u0045\u0044\u0020\u0070\u0069\u006c\u006c\u003f\u0022\u0029\u003b\u002f\u002a
```

Logramos pasar al siguiente reto:

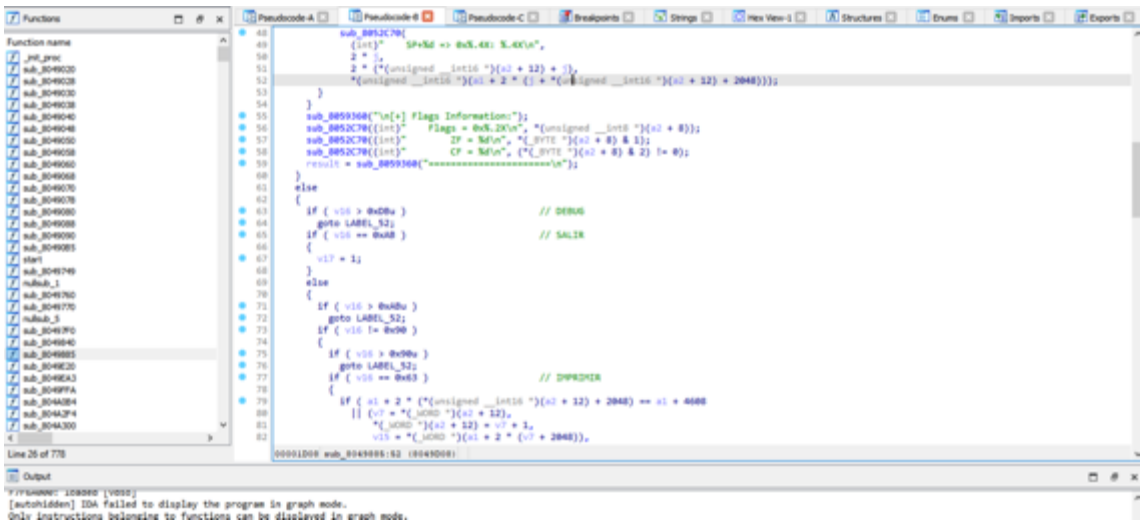
http://xmas2024.s4ur0n.com/level5/xmas2024-vm.zip

## Challenge 12:

Descargamos el adjunto, y lo descomprimimos:

```
$ wget http://xmas2024.s4ur0n.com/level5/xmas2024-vm.zip &>/dev/null

┌──(venv)─(oreos㉿oreos)-[/media/psf/Descargas/xmas2024-vm/tmp]
└─$ 7z x xmas2024-vm.zip

7-Zip 24.08 (x64) : Copyright (c) 1999-2024 Igor Pavlov : 2024-08-11
 64-bit locale=C.UTF-8 Threads:32 OPEN_MAX:1024

Scanning the drive for archives:
1 file, 314281 bytes (307 KiB)

Extracting archive: xmas2024-vm.zip
--
Path = xmas2024-vm.zip
Type = zip
Physical Size = 314281

Everything is Ok

Files: 3
Size:       682376
Compressed: 314281

$ ls
checksums.txt  ctf.hex  vm  xmas2024-vm.zip
```

Analizamos el binario con IDA, y observamos que posee una VM con su propio juego de instrucciones, los cuales son introducidos al binario a través de ctf.hex.



Podemos observar algunas de las operaciones que realiza la VM, como son las siguientes:

- Si detecta el byte “0xAB”, establece el valor “v17 = 1” y sale del bucle, o lo que es lo mismo, finaliza el programa.
- Si detecta el byte “0xDB”, imprime el estado de la VM con sus registros.
- Si detecta el byte “0x63”, imprime el valor que tenga en la pila la VM.
- Etc.

Entre las operaciones que realiza la VM, se encuentra “0x33”, que realiza un salto condicional si el registro “ZF” no está activado (JNZ).

Realizando un análisis del fichero “ctf.hex”, observamos lo siguiente:

```
90
90
13 00dead
43 002e2e
90
33 2500                         # Salto JNZ (ZF=0) a 0x25 (Ops!)
13 01dead
90
43 012631
33 2500                         # Salto JNZ (ZF=0) a 0x25 (Ops!)
13 002d00                       # Guarda 0x2d (Well done!)
90
53 00
63                              # Imprime
23 2c00
13 003c00                       # Guarda 0x3c (Ops!)
53 00
63                              # Imprime
Ab                              # Salir
57656c6c20646f6e652121210d0a00 # Well done!!!
4f7073210d0a00                  # Ops!
```

Depurando la ejecución del binario, observamos que realiza la siguiente operación con la entrada del usuario (8 bytes):

C1 C2 C3 C4 C5 C6 C7 C8

[C1 C2] ^ [C5 C6] = [0x2e 0x2e]
[C3 C4] ^ [C7 C8] = [0x26 0x31]

Realizamos un script en Python para generar todos los conjuntos que permitan satisfacer ambas condiciones, y poder generar todas las combinaciones:

```
cs="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"


def change_to_be_hex(s):
    return int(s,base=16)


def xor_two_str(str1,str2):
    a = change_to_be_hex(str1)
    b = change_to_be_hex(str2)
'''
elems1 = []
for c1 in cs:
  for c2 in cs:
    for c3 in cs:
      for c4 in cs:
        w1 = c1
        w2 = c2
        w3 = c3
        w4 = c4
        w1 = w1.encode('utf-8')
        w2 = w2.encode('utf-8')
        w3 = w3.encode('utf-8')
        w4 = w4.encode('utf-8')
        xor_result1 = bytes(a ^ b for a, b in zip(w1, w2))
        hex_result1 = xor_result1.hex()
        xor_result2 = bytes(a ^ b for a, b in zip(w3, w4))
        hex_result2 = xor_result2.hex()
        if hex_result1 == "2e" and hex_result2 == "2e":
            elems1.append(w1.decode('utf-8')+w3.decode('utf-8')+","+w2.decode('utf-
8')+w4.decode('utf-8'))


elems2 = []
for c1 in cs:
  for c2 in cs:
```

```
    for c3 in cs:
      for c4 in cs:
        w1 = c1
        w2 = c2
        w3 = c3
        w4 = c4
        w1 = w1.encode('utf-8')
        w2 = w2.encode('utf-8')
        w3 = w3.encode('utf-8')
        w4 = w4.encode('utf-8')
        xor_result1 = bytes(a ^ b for a, b in zip(w1, w2))
        hex_result1 = xor_result1.hex()
        xor_result2 = bytes(a ^ b for a, b in zip(w3, w4))
        hex_result2 = xor_result2.hex()
        if hex_result1 == "31" and hex_result2 == "26":
            elems2.append(w1.decode('utf-8')+w3.decode('utf-8')+","+w2.decode('utf-
8')+w4.decode('utf-8'))

print(elems1)
print(elems2)
```

Una vez que disponemos de los conjuntos [C1 C2],[C5 C6] y [C3 C4],[C7 C8], podemos generar todos las entradas válidas:

```
for e1 in elems1:
    for e2 in elems2:
        t1 = e1.split(',')
        t2 = e2.split(',')
        pwd = t1[0]+t2[0]+t1[1]+t2[1]
        print(pwd)
```

Guardamos todas las combinaciones posibles y filtramos aquellas que posean la palabra xmas:

```
$ grep -ri xmas pwd4
vcpuXMAS
vcpUXMAs
vcPuXMaS
vcPUXMas
vCpuXmAS
vCpUXmAs
vCPuXmaS
vCPUXmas
```

```
xmasVCPU
xmaSVCPu
xmAsVCpU
xmASVCpu
xMasVcPU
xMaSVcPu
xMAsVcpU
xMASVcpu
VcpuxMAS
VcpUxMAs
VcPuxMaS
VcPUxMas
VCpuxmAS
VCpUxmAs
VCPuxmaS
VCPUxmas
XmasvCPU
XmaSvCPu
XmAsvCpU
XmASvCpu
XMasvcPU
XMaSvcPu
XMAsvcpU
XMASvcpu
```

Comprobamos con el siguiente script:

```
for e1 in elems1:
    for e2 in elems2:
        t1 = e1.split(',')
        t2 = e2.split(',')
        pwd = t1[0]+t2[0]+t1[1]+t2[1]
        h = hashlib.new('sha256')
        #print(pwd)
        h.update(pwd.encode('utf-8'))
        hsh=h.hexdigest()
        #print(hsh)
        if (pwd.lower() == "vcpuxmas"):
            r=requests.get("https://xmas2024.s4ur0n.com/level5/verify-
xmas.php?pwd="+hsh,allow_redirects=False)
```

```
            if 'Location' not in r.headers or r.headers['Location'] !=
"https://xmas2024.s4ur0n.com/level5/c2ed60eb1f5f3ff15b5c56678f4f865900a03d56ed3e922ad
faa3ce0fbe89b36/":
                print("FOUND!!")
                print(r.headers['Location'])
                print(pwd)
                print(hsh)
                exit()
```

Ejecutamos y obtenemos el input válido:

```
$ python3 guesser.py
FOUND!!
https://xmas2024.s4ur0n.com/level5/7ec62471a03e0a9e41e06a56701665dca4f73d90e49b6eb24c
89b0f6858e731f/
vCPUXmas
08bdbd247ddf8b8ce6b5a4f85178ba323c50cb075cfb1620a2ecfd09043ae926
```