

## ChronoVM

El binario chronovm solicita una clave por entrada estándar y responde únicamente con Correct! o Nope..

No existe ninguna comparación directa contra la flag, ni cadenas visibles que permitan obtenerla trivialmente.

El reto implementa una máquina virtual personalizada (VM) junto con una verificación criptográfica basada en SHA-256, lo que obliga a comprender el flujo interno del binario para poder resolverlo.

```
└$ file chronovm
chronovm: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=f53391a2da21c16f1230d347e93fd4726fb07bad, for GNU/Linux 3.2.0, stripped
```

En la función de entrada del binario (entry) se observa el flujo estándar de inicialización de un ejecutable ELF en Linux. El código prepara los registros y realiza una llamada a \_\_libc\_start\_main, desde donde se transfiere el control a la función main.

No se detecta ninguna lógica de validación o comparación de la flag en esta etapa, confirmando que el binario sigue un flujo convencional y que el análisis debe centrarse en funciones posteriores.

```

+-----+
undefined processEntry entry()
△<UNASSIGNED> <RETURN>
entry
XREF[2]: Entry Point(*), 00100018(*)  

00101500 31 ed    XOR     EBP,Ebp
00101502 49 89 d1 MOV     R9,Rdx
00101505 5e       POP    Rsi
00101506 48 89 e2 MOV     Rdx,Rsp
00101509 48 83 e4 f0 AND    Rsp,-0x10
0010150d 50       PUSH   Rax
0010150e 54       PUSH   Rsp
0010150f 45 31 c0 XOR    R8d,R8d
00101512 31 c9 XOR    ECX,ECX
00101514 48 8d 3d LEA    Rdi,[LAB_00101100]
e5 fb ff ff
0010151b ff 15 9f CALL   qword ptr [-><EXTERNAL>::__libc_start_main] undefined __libc_start_main
2a 00 00
00101521 f4       HLT
00101522 66       ??      66h   f
00101523 2e       ??      2eh   .
00101524 0f       ??      0fh
00101525 1f       ??      1fh
00101526 84       ??      84h
00101527 00       ??      00h
00101528 00       ??      00h
00101529 00       ??      00h
0010152a 00       ??      00h
0010152b 00       ??      00h
0010152c 0f       ??      0fh
0010152d 1f       ??      1fh
0010152e 40       ??      40h   @
0010152f 00       ??      00h
00101530 48       ??      48h   H
00101531 8d       ??      8dh
00101532 7d       ??      7dh

```

En esta sección del código se identifica el bucle principal de la máquina virtual implementada por el binario. Cada iteración lee una instrucción del bytecode y ejecuta una operación específica mediante un mecanismo de dispatch tipo switch, basado en el valor del opcode.

Se observan comparaciones contra valores constantes (CMP SIL, 0x6) y saltos a diferentes casos, lo que corresponde a operaciones como XOR, sumas, rotaciones y sustituciones mediante S-box. Este diseño introduce una capa adicional de abstracción que dificulta significativamente la inversión directa del algoritmo.

	LAB_00101380		XREF[1] :	001013d7(j)
00101380	41 0f b6 34 04	MOVZX	ESI,byte ptr [R12 + RAX*0x1]	
00101385	8d 50 01	LEA	EDX,[RAX + 0x1]	
00101388	40 80 fe ff	CMP	SIL,0xff	
0010138c	74 4b	JZ	LAB_001013d9	
0010138e	41 0f b6 14 14	MOVZX	EDX,byte ptr [R12 + RDX*0x1]	
00101393	8d 48 02	LEA	ECX,[RAX + 0x2]	
00101396	41 0f b6 0c 0c	MOVZX	ECX,byte ptr [R12 + RCX*0x1]	
0010139b	83 e2 1f	AND	EDX,0x1f	
0010139e	40 80 fe 06	CMP	SIL,0x6	
001013a2	77 2c	JA	switchD_001013ab::caseD_0	
001013a4	48 63 34 b7	MOVSXD	RSI,dword ptr [RDI + RSI*0x4]=>switchD_001013a... = FFFFF390h	
001013a8	48 01 fe	ADD	RSI,RDI	

Antes de ejecutar la máquina virtual, el binario aplica una transformación inicial a la entrada del usuario.

Esta lógica puede observarse claramente en Ghidra en un bucle que itera sobre cada carácter ingresado.

Tras ejecutar la máquina virtual, el binario obtiene un estado final de 32 bytes.

Este estado no se imprime, ni se compara directamente con la flag.

En su lugar, el binario realiza una comparación byte a byte contra un estado objetivo almacenado en vm.bin.

Para evitar la exposición directa del oracle durante la ejecución normal, el binario solo muestra el valor Matches cuando se habilita un modo de depuración mediante una variable de entorno.

Debido a que el binario no compara la flag directamente, sino que utiliza una métrica de similitud global (Matches), no es posible validar la entrada carácter por carácter ni aplicar fuerza bruta. El valor Matches actúa como una función de fitness ruidosa, ya que puede aumentar incluso con caracteres incorrectos y no es monotónico.

Para resolver el reto, se aborda el problema como una búsqueda heurística en un espacio grande de candidatos. Se utiliza una estrategia de beam search, en la que se mantienen varios candidatos simultáneamente y solo se conservan aquellos que presentan una mayor similitud con el estado objetivo. De esta manera, se reduce el impacto de los máximos locales y se permite una convergencia progresiva hacia la flag correcta sin invertir directamente la máquina virtual.

Solucion:

```

#!/usr/bin/env python3
import subprocess
import re
import string
from collections import defaultdict

BIN = "./chronovm"

# charset típico de flags CTF
CHARSET = string.ascii_letters + string.digits + "_{}"

# Parámetros (puedes subirlos si quieres más potencia)
MAX_LEN = 40 # límite de longitud
BEAM_WIDTH = 200 # cuántos candidatos mantener por nivel
REPEATS = 1 # repetir ejecución para promedio (sube a 3 si quieres más estabilidad)

def get_matches(candidate: str) -> int:
    """
    Ejecuta el binario y extrae X de 'Matches: X/32'.
    Devuelve -1 si no encuentra la línea.
    """
    total = 0
    for _ in range(REPEATS):
        p = subprocess.run(
            [BIN],
            input=(candidate + "\n").encode(),
            stdout=subprocess.PIPE,
            stderr=subprocess.DEVNULL,
            check=False,
        )
        out = p.stdout.decode(errors="ignore")
        m = re.search(r"Matches:\s*(\d+)\s*/\s*32", out)
        if not m:
            return -1
        total += int(m.group(1))
    return total // REPEATS

def solve():
    # empezamos con algo razonable
    start = "FIS"
    s0 = get_matches(start)

```

```

if s0 < 0:
print("[-] No pude leer 'Matches: X/32'. ¿Estás usando el chronovm con oracle Matches?")
return

beam = [(start, s0)]
seen = set([start])

print(f"[*] Start beam: {beam}")

for step in range(MAX_LEN):
scored = defaultdict(int)

# expandimos el beam
for cand, sc in beam:
for ch in CHARSET:
nxt = cand + ch
if nxt in seen:
continue
seen.add(nxt)

m = get_matches(nxt)
if m < 0:
print("[-] Salida inesperada del binario (no encontré Matches).")
return

scored[nxt] = m

# si ya está perfecto, listo
if m == 32:
print("\n[✓] FLAG ENCONTRADA:", nxt)
return

if not scored:
print("[-] No hay expansiones nuevas (beam vacío).")
return

# seleccionamos top candidates por score y preferimos más cortos si empatan
beam = sorted(scored.items(), key=lambda x: (-x[1], len(x[0])))[:BEAM_WIDTH]

best_cand, best_score = beam[0]
print(f"[+] Step {step:02d} | best: ({best_cand!r}, {best_score}/32) | beam_size={len(beam)}")

```

```

# heurística: si ya llegamos a ')', prioriza terminación (opcional)
for cand, sc in beam:
    if cand.endswith("}") and sc == 32:
        print("\n[✓] FLAG ENCONTRADA:", cand)
        return

print("[-] No encontrada dentro del límite. Sube BEAM_WIDTH/MAX_LEN o amplía CHARSET.")

if __name__ == "__main__":
    solve()

```

```

└$ python3 solucion.py
[*] Start beam: [('FIS', 7)]
[+] Step 00 | best: ('FIS', 8/32) | beam_size=65
[+] Step 01 | best: ('FIS{u', 10/32) | beam_size=200
[+] Step 02 | best: ('FIS{us', 12/32) | beam_size=200
[+] Step 03 | best: ('FIS{usP', 13/32) | beam_size=200
[+] Step 04 | best: ('FIS{usPn', 14/32) | beam_size=200
[+] Step 05 | best: ('FIS{usPn_', 16/32) | beam_size=200
[+] Step 06 | best: ('FIS{usPn_c', 17/32) | beam_size=200
[+] Step 07 | best: ('FIS{us3n_cH', 18/32) | beam_size=200
[+] Step 08 | best: ('FIS{us3n_cHr', 19/32) | beam_size=200
[+] Step 09 | best: ('FIS{us3n_cHr0', 21/32) | beam_size=200
[+] Step 10 | best: ('FIS{us3n_cHr0n', 23/32) | beam_size=200
[+] Step 11 | best: ('FIS{us3n_cHr0n0', 25/32) | beam_size=200
[+] Step 12 | best: ('FIS{us3n_cHr0n0_', 27/32) | beam_size=200
[+] Step 13 | best: ('FIS{us3n_cHr0n0_v', 28/32) | beam_size=200
[+] Step 14 | best: ('FIS{us3n_cHr0n0_vM', 30/32) | beam_size=200

[✓] FLAG ENCONTRADA: FIS{us3n_cHr0n0_vM}

```