La máquina dentro de la máquina

Al ejecutar el binario:

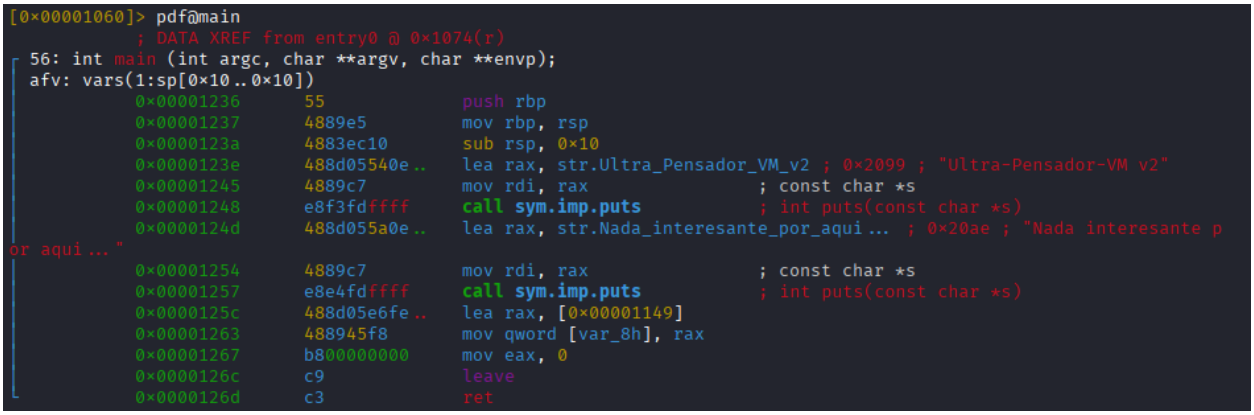
```
└─$ ./Ultra-Pensador-VM-hard
Ultra-Pensador-VM v2
Nada interesante por aqui ...
```

El programa no solicita entrada ni imprime información sensible, lo que indica que la lógica real del reto está oculta y requiere análisis estático.

Al usar strings, no aparece ninguna flag ni texto sospechoso.
Esto sugiere que la flag no está en texto plano.

El análisis de la función main revela que el programa únicamente imprime dos mensajes y finaliza su ejecución. No existe ninguna llamada directa a funciones que procesen los datos observados en .rodata.

Sin embargo, se identifica una instrucción lea que carga la dirección 0x1149 en una variable local, sin que dicha dirección sea ejecutada mediante una instrucción call. Esto indica la posible existencia de código relevante que nunca es invocado durante la ejecución normal del programa.



Al analizar el código ubicado en la dirección 0x1149, se observa una función completa que no es invocada desde main. Esta función accede directamente al bloque de datos ubicado en la sección .rodata, descifra cada byte aplicando una operación XOR con 0x5A y ejecuta un bucle de interpretación.

Las comparaciones contra los valores 0x01, 0x05 y 0xFF indican la implementación de instrucciones personalizadas, confirmando que se trata de una máquina virtual encargada de ejecutar el bytecode previamente identificado.

Se observa la sección:

.rodata vaddr=0x00002000 size=0xcb

Nos posicionamos en dicha sección:

Una vez identificado el bloque de datos no ASCII dentro de la sección .rodata, se procede a extraer el bytecode cifrado para su posterior análisis y ejecución fuera del binario.

A partir del análisis previo se determinó que el bytecode comienza en la dirección virtual 0x2020. Utilizando radare2, se extrae dicho bloque directamente a un archivo binario:

s 0x2020

wtf enc.bin 0x79

El bytecode extraído se encuentra ofuscado mediante una operación XOR simple. Cada byte del programa virtual es descifrado aplicando la siguiente operación:

byte_descifrado = byte_cifrado ^ 0x5A

Para obtener la flag, se implementó un pequeño intérprete en Python que replica el comportamiento de la máquina virtual identificada en el binario.

```
KEY = 0x5A

def run_vm(enc):
regs = [0, 0, 0, 0]
pc = 0
out = []

while pc < len(enc):
op = enc[pc] ^ KEY
pc += 1

if op == 0x01: # LOADI
if pc + 1 >= len(enc):
break
r = (enc[pc] ^ KEY) & 0x03
pc += 1
imm = enc[pc] ^ KEY
pc += 1
regs[r] = imm

elif op == 0x05: # PRINT
if pc >= len(enc):
break
r = (enc[pc] ^ KEY) & 0x03
pc += 1
out.append(chr(regs[r]))
```

```
elif op == 0xFF: # HALT
break
else:
break

return "".join(out)

if __name__ == "__main__":
with open("enc.bin", "rb") as f:
enc = f.read()

print("Flag:", run_vm(enc))
```

```
└─$ python3 solucion.py
Flag: FIS{VM_r3v_h4rd_but_fun}
```