Write-up: Reto pwn Wordle

Descripción del Reto

"Crees que eres bueno en el Wordle??? Prueba la versión difícil de Wordle, acierta las 100 palabras seguidas, muajjajajaja."

Al ejecutarlo, pide un nombre de usuario antes de comenzar el juego.

Análisis Inicial y Reconocimiento

Al ejecutar el binario proporcionado (wordle), confirma la descripción: pide un nombre de usuario y luego inicia el juego de adivinar palabras. Fallar una sola palabra termina el programa.

Ejecutar `strings ./wordle` revela información interesante:

- Mensajes del juego ("Introduce un nombre de usuario:", "Palabra %d:", "Correcto!", etc.)
- Un mensaje de debug crucial: `[DEBUG] Numero de caracteres: %d`
- Un mensaje de error relacionado con la flag: "No se pudo abrir el archivo flag"
- Una lista de palabras que parecen ser las usadas en el juego
- Funciones de libc como srand, rand, fgets, strcmp, fopen, printf

Análisis del Código (Ghidra)

El análisis de la función main revela la lógica clave:

Inicialización de la Lista de Palabras:

- Se declara un array local de 100 punteros
- Un bucle inicializa este array empezando con la palabra "mezquita"

Entrada del Usuario y Semilla del RNG:

- Se lee el nombre de usuario con fgets
- Se calcula la longitud del nombre de usuario

- Se imprime la longitud: `printf("\n[DEBUG] Numero de caracteres: %d\n",(ulong)local_10);`
- **¡Vulnerabilidad Principal!** Se inicializa el PRNG usando la longitud del nombre de usuario como semilla: `srand(local_10);`

Bucle del Juego:

- En cada iteración:
  - Se llama a rand() para obtener un número pseudoaleatorio
  - Se calcula un índice: iVar1 % 100
  - Se selecciona la palabra secreta de la lista usando ese índice
  - Se pide al usuario que introduzca una palabra
  - Se compara con strcmp; si no coinciden, el programa termina

Condición de Victoria:

- Si el bucle se completa 100 veces, se imprime "Imposible que hayas ganado..." y se muestra la flag

Identificación de la Vulnerabilidad

La vulnerabilidad principal es el uso de un PRNG predecible. La secuencia generada por rand() depende completamente de la semilla proporcionada a srand(). Como la semilla es la longitud del nombre de usuario, que controlamos y además el programa nos revela, podemos predecir toda la secuencia de números y, por tanto, las palabras requeridas.

Estrategia de Explotación

1. **Fijar la Semilla**: Elegir una longitud de nombre de usuario simple, como 1 (username = "A")
2. **Extraer la Lista de Palabras**: Usar Ghidra para examinar los 100 punteros consecutivos
3. **Predecir la Secuencia de Palabras**: Escribir un script que:
   - Use la misma semilla (srand(1))

- Llame a rand() % 100 cien veces para obtener la secuencia de índices
- Use estos índices para buscar las palabras correspondientes

4. **Automatizar la Interacción**: Usar pwntools para:
   - Conectarse al servicio remoto
   - Enviar el nombre de usuario
   - Enviar las 100 palabras predichas en orden
   - Leer la flag final

Desarrollo del Exploit

Se extrajo la lista completa de 100 palabras del binario, incluyendo algunas particularidades como palabras repetidas ("sierra"), siglas ("UCO") y una cadena vacía que resultó ser "flag" en el servidor.

Se creó un script Python que:

1. Conecta al servicio remoto
2. Envía el nombre de usuario "A" (longitud 1)
3. Inicializa el mismo generador pseudoaleatorio con la semilla 1
4. Predice y envía cada una de las 100 palabras requeridas
5. Recibe y muestra la flag

```
#!/usr/bin/env python3
from pwn import *
import ctypes

# Configuración
HOST = "ctf.hackademics-forum.com"
PORT = 53921
USERNAME = "A"  # Longitud 1

# Lista de 100 palabras extraídas del binario
wordlist = [
```

```
    "mezquita", "califato", "medina", "alcazar", "patios",
    # ... (lista completa de palabras)
]

# Generar secuencia usando libc local
libc = ctypes.CDLL(None)
libc.srand(1)  # Semilla = longitud del username
sequence = [wordlist[libc.rand() % 100] for _ in range(100)]

# Interactuar con el servicio remoto
p = remote(HOST, PORT)
p.recvuntil(b"Introduce un nombre de usuario: ")
p.sendline(USERNAME.encode())
p.recvuntil(b'Adivina las 100 palabras secretas.\n')

# Enviar las 100 palabras
for i, word in enumerate(sequence):
    round_num = i + 1
    p.recvuntil(f"Palabra {round_num}: ".encode())
    p.sendline(word.encode())
    p.recvuntil(b"Correcto!")

# Recibir la flag
p.recvuntil(b"Imposible que hayas ganado")
flag = p.recvall().decode().strip()
print(flag)
```

Ejecución y Flag

Al ejecutar el script, envió con éxito todas las palabras predichas y recibió la flag:

```
hfctf{Pl4nt4st3_l4_s3m1lla_c0rr3ct4}
```