
Writeup: Análisis de archivo.exe

Introducción

Este writeup documenta el análisis de simpleServer.exe, un ejecutable sospechoso analizado mediante ingeniería inversa. El objetivo era extraer información clave como:

- 1-Dominio malicioso
- 2-Credenciales incrustadas (usuario y contraseña)
- 3-Método de cifrado utilizado

Para ello, se emplearon diversas herramientas de análisis estático y dinámico, destacando Radare2 y Python para la extracción y descifrado de datos.

Herramientas Utilizadas

Herramienta y comandos	Función
Radare2 (r2)	Ingeniería inversa del ejecutable, extracción de cadenas, desensamblado y análisis de funciones.
Strings (strings)	Extracción de texto legible del ejecutable para identificar información clave.
Python	Creación de un script para descifrar cadenas cifradas con XOR.
Grep (grep)	Filtrado de cadenas relevantes como dominios, contraseñas y comandos sospechosos.

1. Extracción de Cadenas de Texto

¿Por qué extraer cadenas?

Las cadenas de texto en un ejecutable pueden revelar información sensible, como:

Comandos internos (net user, cmd /c)

Dominios maliciosos

Credenciales embebidas

Mensajes de error que indican funcionalidad

Comando Utilizado

strings simpleServer.exe > extracted_strings.txt

```
(root@kali)-[/home/kali/Downloads]
# cat extracted_strings.txt
!This program cannot be run in DOS mode.
Rich
```

Esto genera un archivo extracted_strings.txt con todas las cadenas legibles.

Filtrar información clave con grep

Comando Utilizado

grep -Ei 'user|pass|auth|domain|info.txt' extracted_strings.txt

```
(root@kali)-[/home/kali/Downloads]
# grep -Ei 'user|pass|auth|domain|info.txt' extracted_strings.txt
net user > C:\Users\User\info.txt
```

2. Análisis del Código en Radare2

r2 -AA archivo.exe

```
(kali@kali)-[~/Downloads]
$ r2 -AA archivo.exe
INFO: Analyze all flags starting with sym. and entry0 (aa)
INFO: Analyze imports (af@ai)
```

Esto muestra todas las cadenas encontradas en el binario, confirmando la presencia de

```
[0x1400020f8]> iz
[Strings]
nth paddr vaddr len size section type string
-----
0 0x00002140 0x140003340 62 63 .rdata ascii abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
1 0x00002180 0x140003380 31 32 .rdata ascii CryptAcquireContext failed: %d\n
2 0x000021a0 0x1400033a0 27 28 .rdata ascii CryptCreateHash failed: %d\n
3 0x000021c0 0x1400033c0 25 26 .rdata ascii CryptHashData failed: %d\n
```

Resultado relevante:

```
25 0x00002380 0x140003580 34 35 .rdata ascii Command executed successfully. %s\n
26 0x000023b0 0x1400035b0 15 16 .rdata ascii Executing info\n
27 0x000023c0 0x1400035c0 33 34 .rdata ascii net user > C:\Users\User\info.txt
28 0x000023e8 0x1400035e8 29 30 .rdata ascii not_a_real_flag_just_trolling
29 0x00002408 0x140003608 14 15 .rdata ascii g`ww|g`dwv+ljf
30 0x00002418 0x140003618 22 23 .rdata ascii umlvm`wEg`ww|g`dwv+ljf
31 0x00002430 0x140003630 15 16 .rdata ascii 4hQm6I66qME}5w$
32 0x00002440 0x140003640 9 10 .rdata ascii N/A Error
```

Esto sugiere que el binario obtiene información de usuarios y contiene posibles cadenas cifradas.

3. Análisis de la Función de Descifrado

Se inspecciona la función identificada con:

pdf @ 0x140001640

```
[0x1400020f8]> pdf @ 0x140001640
160: fcn.140001640 (char *arg1, uint32_t arg2, uint32_t arg3);
; arg char *arg1 @ rcx
; arg uint32_t arg2 @ rdx
; arg uint32_t arg3 @ r8
; var uint32_t var_20h @ rsp+0x20
; var char *s @ rsp+0x40
; var uint32_t var_10h @ rsp+0x48
```

Hallazgos Clave en la Función

- Llama a strlen() para calcular la longitud de la cadena.
- Recorre la cadena carácter por carácter.
- Aplica una operación XOR con la constante 0x05:

```
0x1400016ae 0fbe00 movsx eax, byte [rax]
0x1400016b1 83f005 xor eax, 5
0x1400016b4 488b4c2420 mov rcx, qword [var_20h]
```

Esto confirma que las cadenas están cifradas con XOR y la clave 0x05.

4. Descifrado de las Cadenas en Python

Con la clave XOR identificada, se creó un script en Python para recuperar los valores en texto plano.

Código Python para Descifrar

4. **Descifrado en Python** → Recuperación de los valores en texto claro.

¿Qué significa este hallazgo?

- **El malware se comunica con `berrybears.ioc`**, indicando un servidor de comando y control (C2).
- **Las credenciales embebidas (`phisher@berrybears.ioc`) pueden ser usadas para autenticación.**
- **La contraseña `1mTh3L33tH@x0r!` probablemente se usa para acceder al servidor o cifrar datos antes de enviarlos.**

Conclusión Final

El análisis de `archivo.exe` demostró el uso de **XOR para cifrado de datos** y la existencia de un servidor remoto que recibe información potencialmente robada. Gracias a herramientas como **Radare2, Strings, Grep y Python**, se logró descifrar el contenido del malware y obtener respuestas clave. Este enfoque puede aplicarse a otros binarios maliciosos para extraer credenciales, identificar servidores de ataque y comprender mejor la funcionalidad del malware.