Análisis de malware

Depuración

Gustavo Romero López - gustavo@ugr.es

Updated: 3 de marzo de 2025

Departamento de Ingeniería de Computadores, Automática y Robótica

Por donde vamos...

Parte 1: Análisis básico
- Capítulo 1: Técnicas estáticas básicas
- Capítulo 2: Análisis de malware en máquinas virtuales
- Capítulo 3: Análisis dinámico básico

Parte 2: Análisis estático avanzado
- Capítulo 4: Curso intensivo de ensamblador x86
- Capítulo 5: IDA Pro
- Capítulo 6: Reconocimiento de construcciones de C en ensamblador
- Capítulo 7: Análisis de programas maliciosos para Windows

Parte 3: Análisis dinámico avanzado
- Capítulo 8: Depuración
- Capítulo 9: OllyDbg
- Capítulo 10: Depuración del núcleo con WinDbg

Parte 4: Funcionalidad del malware

Parte 5: Anti-ingeniería inversa

Parte 6: Temas especiales

1. Código fuente vs ensamblador

2. Modo usuario vs modo núcleo

3. Uso de un depurador

4. Excepciones

5. Modificando la ejecución en la práctica

Código fuente vs ensamblador

Código fuente:

- Suelen estar presentes en la mayoría de los IDEs.
- Avance y detención línea a línea de código fuente.
- Permiten consultar el estado de las variables.

Ensamblador:

- Avance y detención en cada instrucción.
- Permiten inspeccionar registros y memoria.
- No requieren del código fuente.

Modo usuario vs modo núcleo

- Modo usuario:
  - Solo se necesita un sistema.
  - un proceso depura a otro con la ayuda del SO.
- Modo núcleo:
  - Requiere dos sistemas: uno con el depurador y otro con el SO.
  - El SO debe estar preparado para ser depurado o ejecutarse en una máquina virtual.
- Más utilizados:
  - WinDbg: soporta ambos modos.
  - OllyDbg: solo modo usuario, el más usado.
  - IDA Pro: permite elegir entre el interno, más limitado, u otro externo.

Uso de un depurador

- Hay dos formas de iniciar la depuración:
  - Ejecutar el programa desde el interior del depurador.
  - Vincular el depurador a un proceso ya en ejecución.
- Todas las hebras se pausarán y podremos examinarlas.
- Permite vincularnos a un programa afectado por malware para su estudio.

“Paso a paso” (*“Single stepping”*)

- Función más básica.
- Ejecuta una única instrucción y devuelve el control al depurador.
- Siempre posible.
- Evitar con programas largos...
  - No suele ser posible ni recomendable.
  - “Que los árboles no te impidan ver el bosque.”
- Imprescindible para descubrir cosas ocultas al análisis estático, ej: zona de memoria codificada con `XOR`.

“Pasar por encima” frente a “entrar” (*“Stepping-Over” vs “Stepping-Into”*)

- Como paso a paso salvo en llamadas a función.
- **“Pasar por encima”** (step-over) ejecuta la función y se detiene en la siguiente instrucción
  - Vital para avanzar rápido sobre funciones conocidas.
- **“Entrar”** (step-into) se detiene en la primera instrucción de la función
  - Peligroso, podemos caer en paso a paso.
  - Solución: **“salir”** (step-out): se detiene en la primera instrucción tras retornar o en la instrucción de retorno.
- Existen funciones que no finalizan.
- Algunas máquinas virtuales permiten grabar y retroceder (VMware).

Puntos de ruptura (*"breakpoints"*)

- No podemos examinar un programa mientras se ejecuta.
- Permiten elegir donde detenernos.
- Uso típico: romper en `call ...` y examinar parámetros.
- Imprescindibles, ej: `call %eax`.
- Ejemplo: capturamos paquetes de red cifrados
  - Localizar la función de envío (send).
  - Añadir punto de ruptura antes del cifrado.
  - Mostrar el texto en claro.
- Tipos:
  - Software: sobrescribir primer byte con `int $0x3` = `0xcc`.
    - Problemas: código automidificable y sumas de verificación.
  - Hardware: en x86, registros con direcciones de memoria.
    - Ventajas: ruptura al acceder y no sólo ejecutar (rwx).
  - Condicionales: ruptura sólo si se cumple cierta condición.

Excepciones

- Mecanismo de los depuradores para conseguir el control.
- 2 oportunidades para gestionar una excepción:
  1. 1ª oportunidad: el depurador elige resolver o pasar al programa.
  2. 2ª oportunidad: el programa no ha podido resolver y el depurador debe hacerlo.
- La 2ª oportunidad sugieren un fallo o la intención del malware de escapar.
- Excepciones más comunes:
  - `int $0x3`: genera excepción para depuración.
  - Bandera "trap": genera excepción tras cada instrucción.
  - Acceso a dirección de memoria inválida o protegida.
  - Instrucción privilegiada en modo no privilegiado (`cli`).

Modificando la ejecución en la práctica

- Los depuradores nos permiten cambiar cualquier cosa dentro de un proceso.
- Ejemplo: evitar la ejecución de una función
  - punto de ruptura en la función
  - al ser alcanzado cambiamos el puntero de instrucción por la dirección de la primera instrucción tras la función
- Analizando malware aparecen muchas situaciones tipo "que pasaría si..."

Modificando la ejecución en la práctica

- Ejemplo: el virus xenófobo
  - ingles: mensaje gracioso
  - japonés: borra el disco duro
  - indonesio: borra el disco duro
  - chino: desinstala el virus

```
00411349   call    GetSystemDefaultLCID
0041134F ❶mov     [ebp+var_4], eax
00411352   cmp     [ebp+var_4], 409h
00411359   jnz     short loc_411360
0041135B   call    sub_411037
00411360   cmp     [ebp+var_4], 411h
00411367   jz      short loc_411372
00411369   cmp     [ebp+var_4], 421h
00411370   jnz     short loc_411377
00411372   call    sub_41100F
00411377   cmp     [ebp+var_4], 0C04h
0041137E   jnz     short loc_411385
00411380   call    sub_41100A
```