

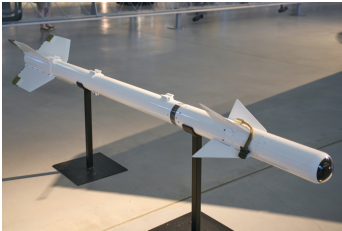
Introduccion a la Ingeniería Inversa

Análisis de malware, cracking de software...

Carlos Ledesma Peña Fernando Díaz Urbano

Grupo de Desarrolladores de Google, Málaga

Antecedentes



Misil K-13 soviético aire-aire



CDP MPC 1600-1

Aplicaciones típicas en informática

- **Mantenimiento de programas sin código fuente**
Interacción de componentes, creación de documentación...
- **Análisis de malware**
Conocer objetivos y procedimientos de un malware
- **Búsqueda de vulnerabilidades**
En programas de código cerrado
- **Modding y cracking**
Añadir o remover funcionalidades

... Entre otras.

Aplicaciones típicas en informática

- **Mantenimiento de programas sin código fuente**
Interacción de componentes, creación de documentación...
- **Análisis de malware**
Conocer objetivos y procedimientos de un malware
- **Búsqueda de vulnerabilidades**
En programas de código cerrado
- **Modding y cracking**
Añadir o remover funcionalidades

... Entre otras.

¿Qué debemos conocer?

- **El entorno**
Proceso, ejecutable, thread, librería...
- **Las herramientas**
Desensamblador, depurador, monitor de recursos...
- **Los lenguajes**
Ensamblador (x86), bytecode (JavaVM), scripting...
- **Los patrones típicos**
Reconocer un bucle for, uso típico de la API Win32...

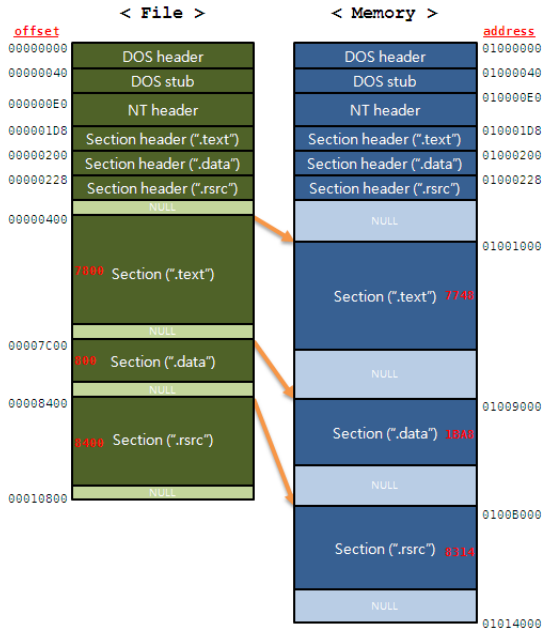
Tipos de análisis

Estático

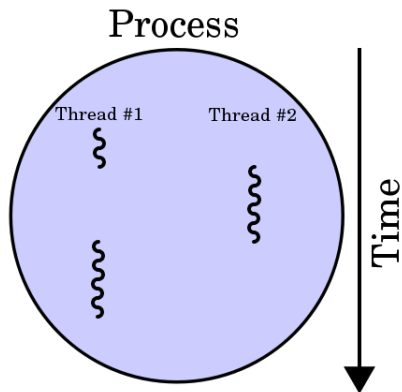
- Permite examinar todos los caminos y valores de variables
- Examen directo de ramas de ejecución poco frecuentes
- No efectivo contra código automodificable o muy ofuscado

Dinámico

- Sólo se ejecuta un camino cada vez, dependiente del entorno
- Más fácil entender lógica complicada viendo la ejecución
- Efectivo contra código automodificable u ofuscado

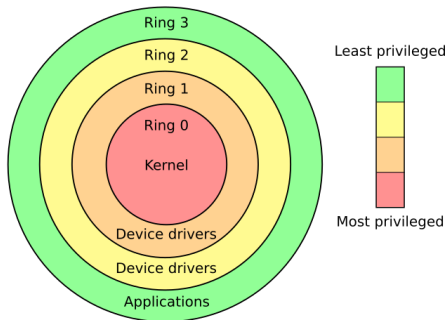


Ejecutables, procesos y threads



- **Ejecutable**
Contenedor de código,
"plantilla" de procesos
- **Proceso**
Rango de memoria,
contenedor de threads
- **Thread**
Contexto de ejecución,
corren en paralelo

Anillos de protección



Win32 API

Casi desde Windows 95 hasta Windows 10...

kernel32.dll

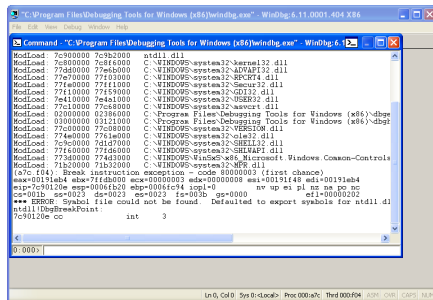
- Procesos: *CreateProcess, CreateThread, LoadLibrary...*
- Archivos: *CreateFile, CopyFile, GetFileSize...*
- Memoria: *VirtualAlloc, HeapAlloc, MapViewOfFile...*

user32.dll

- Iniciar GUI's: *CreateWindowEx, MessageBox, LoadCursor...*
- Gestionar GUI's: *GetMessage, PeekMessage, PostMessage...*

WinDBG

- Depurar ("debuggear") a nivel de kernel (ring 0)
- Debuggear el propio sistema operativo, drivers...
- Curva de aprendizaje empinada, tedioso...
- No tiene GUI, funciona a través de una consola



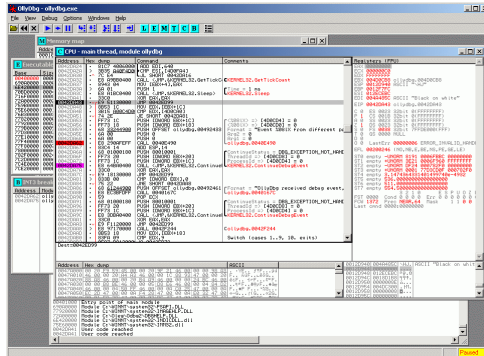
```
"C:\Program Files\Debugging Tools for Windows (x86)\windbg.exe" - WinDbg 6.11.0001.404 X86
File Edit View Debug Windows Help

Command "C:\Program Files\Debugging Tools for Windows (x86)\windbg.exe" - WinDbg 6.11.0001.404 X86
ModLoad: 7c900000 7c903000 ntdll.dll
ModLoad: 7c800000 7c804000 C:\WINDOWS\system32\kernel32.dll
ModLoad: 77d40000 77d4b000 C:\WINDOWS\system32\ADVAPI32.dll
ModLoad: 77e70000 77e73000 C:\WINDOWS\system32\RPCRT4.dll
ModLoad: 77f10000 77f11000 C:\WINDOWS\system32\Secur32.dll
ModLoad: 77f10000 77f19000 C:\WINDOWS\system32\GDI32.dll
ModLoad: 7e410000 7e411000 C:\WINDOWS\system32\USER32.dll
ModLoad: 77c10000 77c18000 C:\WINDOWS\system32\USER32.dll
ModLoad: 02000000 02386000 C:\Program Files\Debugging Tools for Windows (x86)\dbgpe
ModLoad: 03000000 03121000 C:\Program Files\Debugging Tools for Windows (x86)\dbgpe
ModLoad: 77c00000 77c08000 C:\WINDOWS\system32\VERSION.dll
ModLoad: 774e0000 774e0000 C:\WINDOWS\system32\ole32.dll
ModLoad: 7c9c0000 7c9c7000 C:\WINDOWS\system32\GHELL32.dll
ModLoad: 77f60000 77f6d000 C:\WINDOWS\system32\GHELLAPI.dll
ModLoad: 77340000 77343000 C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls
ModLoad: 71b30000 71b32000 C:\WINDOWS\system32\NFI.dll
(s/c f04): Break instruction exception - code 80000003 (first chance)
eax=00191ab4 ebx=71f4b800 ecx=00000003 edx=00000008 esi=00191f48 edi=00191ab4
eip=7c90120e esp=0006fb20 ebp=0006ic94 iopl=0         nv up ei pl zr na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for ntdll.dll
ntdll!DbgBreakPoint:
7c90120e cc                int     3

0:000>
```

OllYDBG

- Debuggear a nivel de aplicaciones (ring 3)
- Más sencillo para principiantes
- Posee GUI



Más herramientas

- **Análisis estático de ejecutables**

PEview, Dependency Walker, Strings, PEiD...

- **Monitores**

Process Explorer, Process Monitor, Wireshark, Regshot...

- **Análisis de malware**

Cuckoo Sandbox, AndroGuard, VirusTotal, Koodous...

Un poco más de entorno...

Simplificadamente, las partes de un programa en memoria:

- **Código**

Instrucciones en código máquina

- **Datos**

Variables globales, constantes...

- **Stack**

Variables locales, anidamiento de llamadas...

- **Heap**

Reserva de memoria bajo demanda

Registros principales en x86

- **De propósito general**
EAX, EBX, ECX, EDX
- **Stack**
ESP y EBP
- **Instruction pointer**
EIP
- **Registro de estado**
EFLAGS

Tipos de instrucciones

- **Aritméticas y transferencia**
mov, add, sub, inc, dec, xor...
- **Stack**
push, pop, pushad, popad...
- **Comparación y saltos**
test, cmp, jmp, je, jne, jl, jge...
- **Llamadas a funciones**
call, return...

Convenios de llamada: *stdcall*

- Se empujan los **argumentos** de la función en la **pila** en **orden inverso**
- El responsable de **limpiar la pila** de los argumentos es la **función llamada**
- Lo retornado por la función (**return**) se devuelve en el **registro EAX**

Bucle for en x86

```
401000 mov ecx,10      #Mueve el valor 10 a ecx
401005 push FF          #Comienzo del cuerpo del bucle
...
# Cuerpo del bucle
...
401039 sub ecx,1        #Resta 1 a ecx
40103C jnz 401005
#Salta a 401005 -> el resultado != cero
```

Recursos recomendados

- **Modding y cracking**

The Legend Of Random

<http://thelegendofrandom.com/>

- **Análisis de malware**

Practical Malware Analysis

Michael Sikorski y Andrew Honig

¡Gracias por vuestra atención!