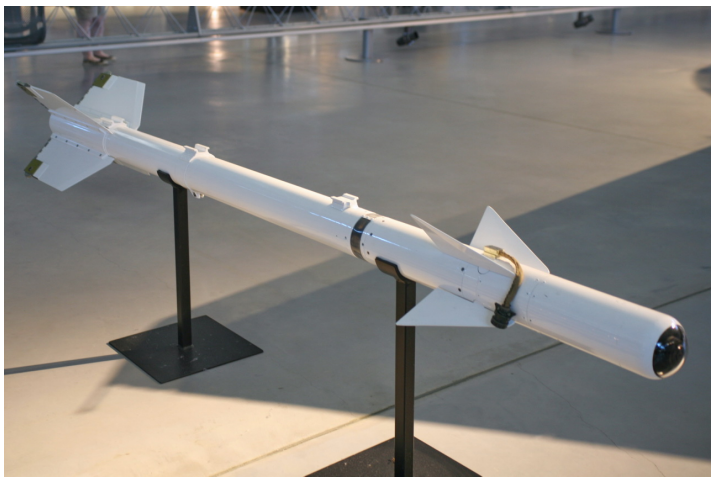
Introduccion a la Ingeniería Inversa

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...

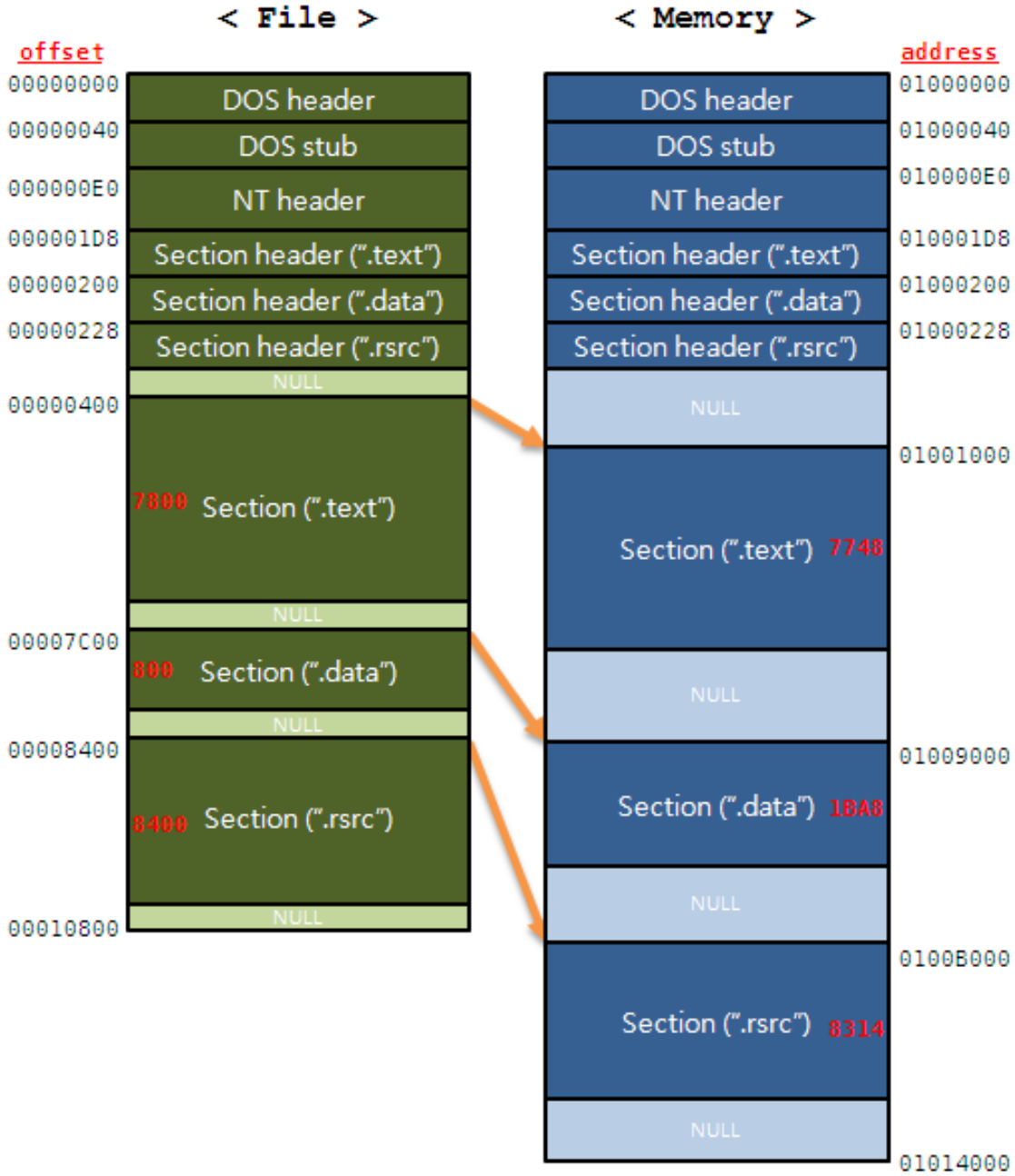
Tipos de análisis

Estático

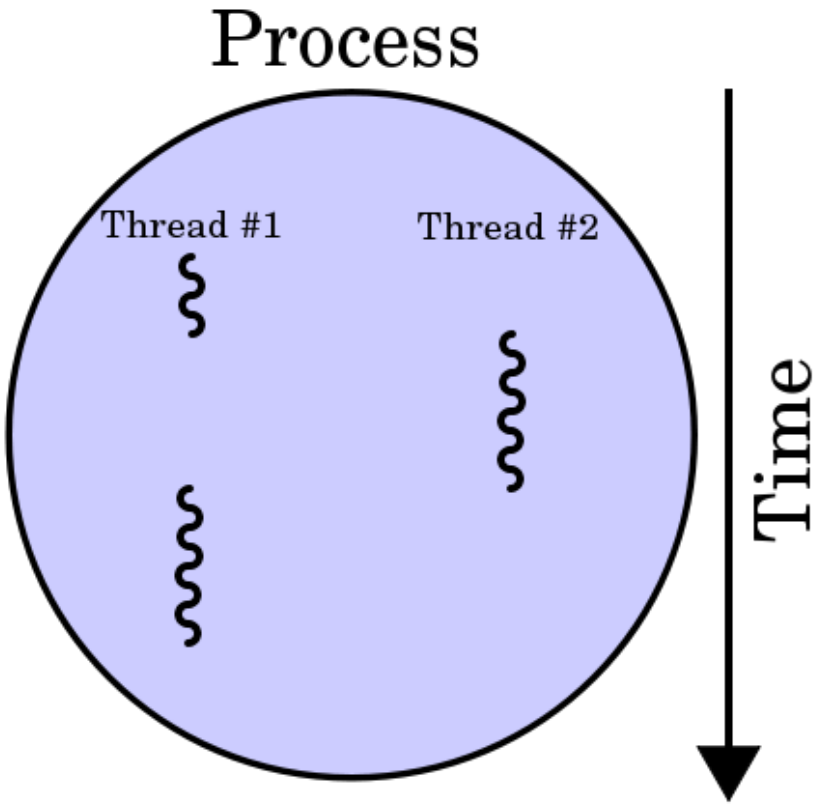
- Permite examinar todos los caminos y valores de variables
- Examen directo de ramas de ejecución poco frecuentes
- No efectivo contra código automodificable o muy ofuscado

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

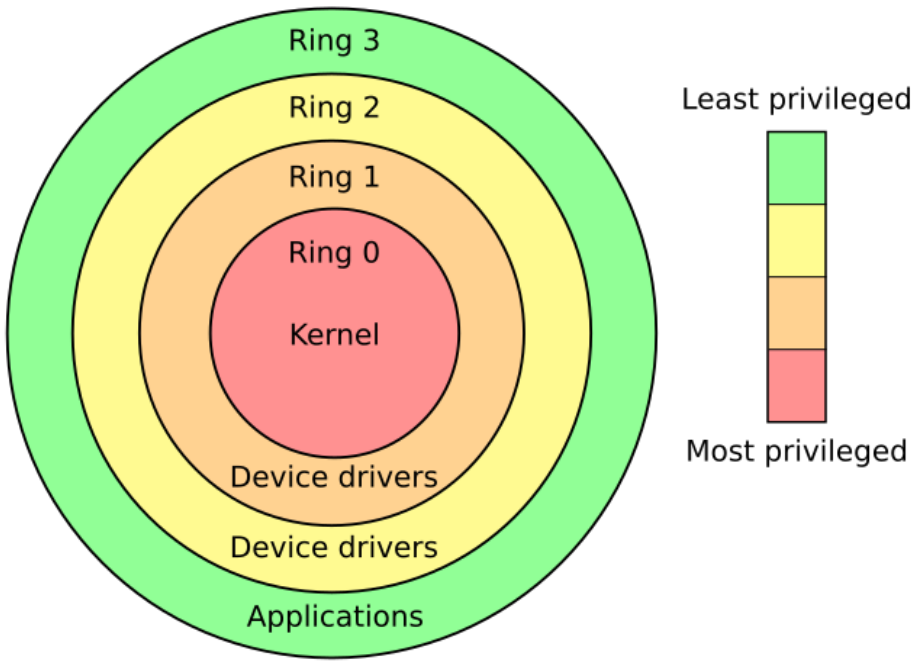

< File >
< Memory >
offset
address
00000000
DOS header
01000000
00000040
DOS stub
01000040
000000E0
NT header
010000E0
000001D8
Section header (".text")
010001D8
00000200
Section header (".data")
01000200
00000228
Section header (".rsrc")
01000228
NULL
00000400
7800
Section (".text")
01001000
Section (".text") 7748
NULL
00007C00
800
Section (".data")
NULL
01009000
Section (".data") 1BA8
00008400
8400
Section (".rsrc")
NULL
0100B000
Section (".rsrc") 8314
00010800
NULL
NULL
01014000

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

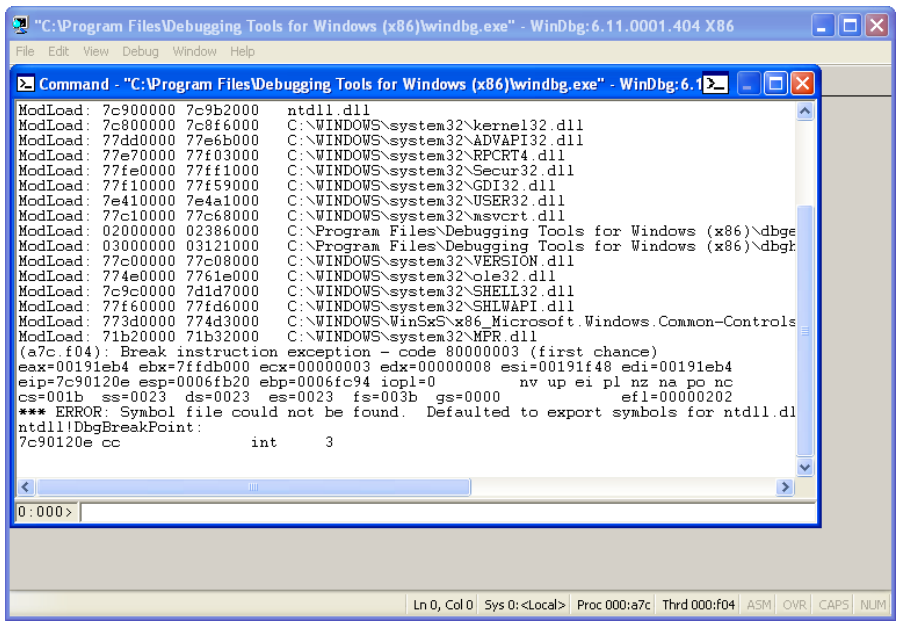
- Procesos: *CreateProcess*, *CreateThread*, *LoadLibrary*...
- Archivos: *CreateFile*, *CopyFile*, *GetFilesize*...
- Memoria: *VirtualAlloc*, *HeapAlloc*, *MapViewOfFile*...

user32.dll

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

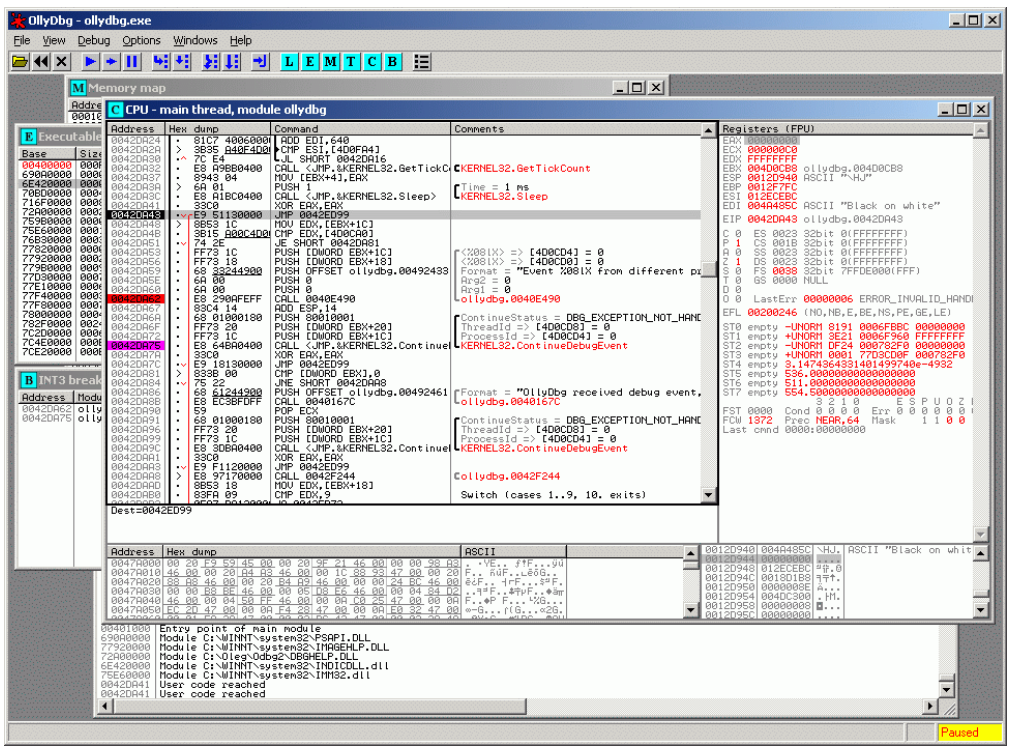
WinDBG

- 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



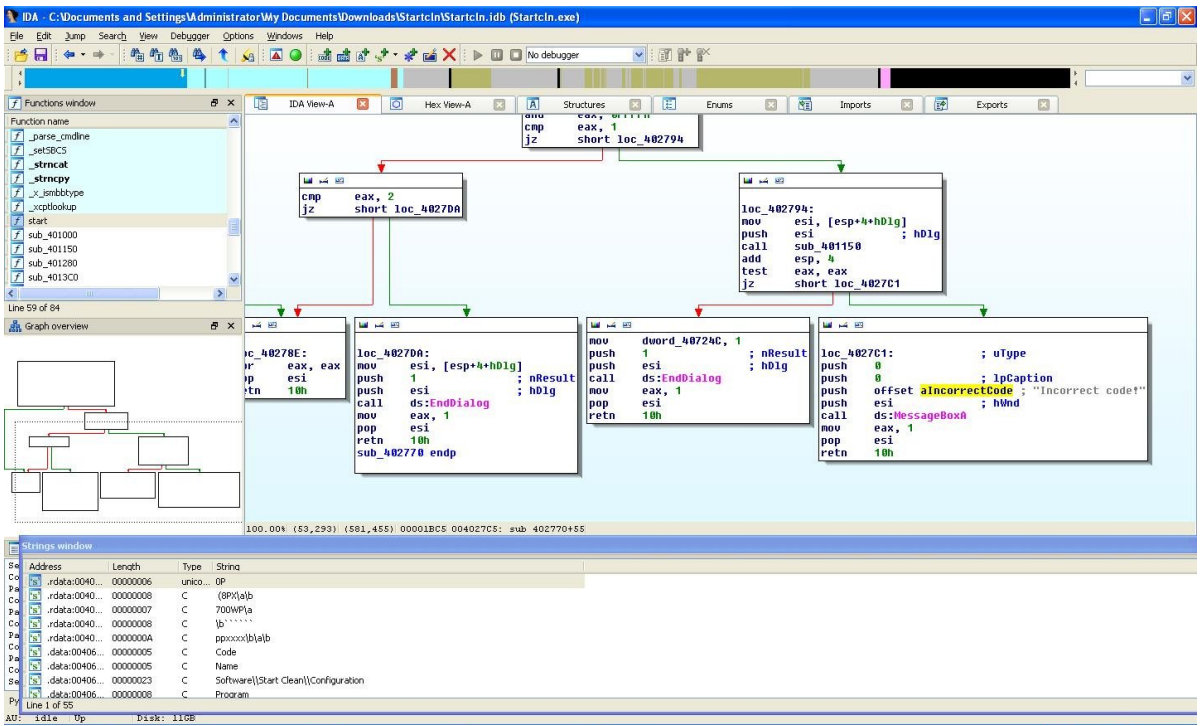
OllyDBG

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



IDA Pro

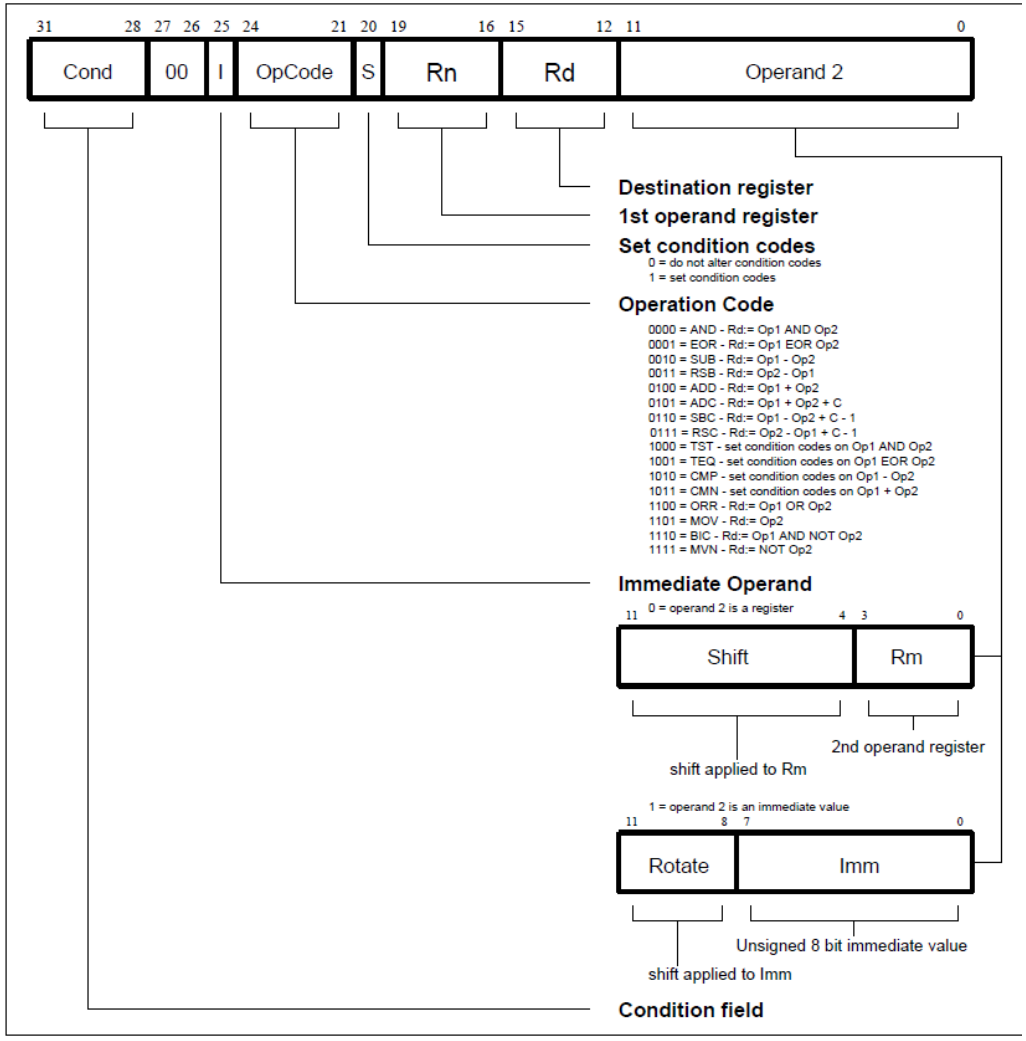
- Desemsamblador por excelencia, muchas funciones
- Destaca por analizar profundamente
- Dificultad media
- Posee GUI



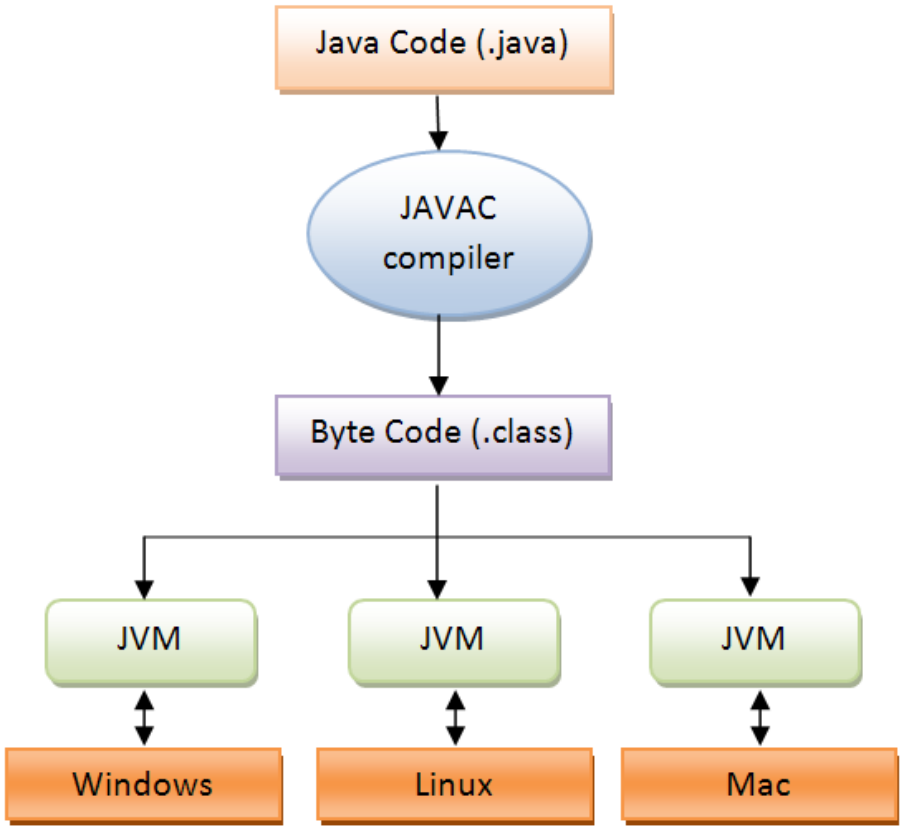
Más herramientas

- **Análisis estático de ejecutables**
  PEview, Dependency Walker, Strings, PEiD...
- **Monitores**
  Process Explorer, Process Monitor, Wireshark, Regshot...
- **Análisis de malware**
  Cuckoo Sandbox, AndroGuard, VirusTotal, Koodous...

Opcodes


31 28 27 26 25 24 21 20 19 16 15 12 11 0
Cond
00
I
OpCode
S
Rn
Rd
Operand 2
Destination register
1st operand register
Set condition codes
0 = do not alter condition codes
1 = set condition codes
Operation Code
0000 = AND - Rd:= Op1 AND Op2
0001 = EOR - Rd:= Op1 EOR Op2
0010 = SUB - Rd:= Op1 - Op2
0011 = RSB - Rd:= Op2 - Op1
0100 = ADD - Rd:= Op1 + Op2
0101 = ADC - Rd:= Op1 + Op2 + C
0110 = SBC - Rd:= Op1 - Op2 + C - 1
0111 = RSC - Rd:= Op2 - Op1 + C - 1
1000 = TST - set condition codes on Op1 AND Op2
1001 = TEQ - set condition codes on Op1 EOR Op2
1010 = CMP - set condition codes on Op1 - Op2
1011 = CMN - set condition codes on Op1 + Op2
1100 = ORR - Rd:= Op1 OR Op2
1101 = MOV - Rd:= Op2
1110 = BIC - Rd:= Op1 AND NOT Op2
1111 = MVN - Rd:= NOT Op2
Immediate Operand
0 = operand 2 is a register
11 4 3 0
Shift
Rm
2nd operand register
shift applied to Rm
1 = operand 2 is an immediate value
11 8 7 0
Rotate
Imm
Unsigned 8 bit immediate value
shift applied to Imm
Condition field

Bytecode


Java Code (.java)
JAVAC compiler
Byte Code (.class)
JVM
JVM
JVM
Windows
Linux
Mac

Seguridad

- **2005:** Versión mejorada de *BlueSniper*.
- **2007:** Ataque sobre teclados inalámbricos.
- Ataque sobre el pasaporte europeo.
- **2008:** Michael Ossmann repasa en Black Hat sobre el estado de la seguridad de las radiocomunicaciones, y advierte que SDR accesible es peligroso.
- Ataque sobre el sistema de tarjetas del metro de Boston y de pago remoto en peajes.
- **2009:** Ataque práctico sobre *GSM*.
- **2010:** Lectura de *RFID* a larga distancia.
- **2011:** Ataque sobre *GRPS/EDGE* y *UMTS/HSPA*.

Estado actual

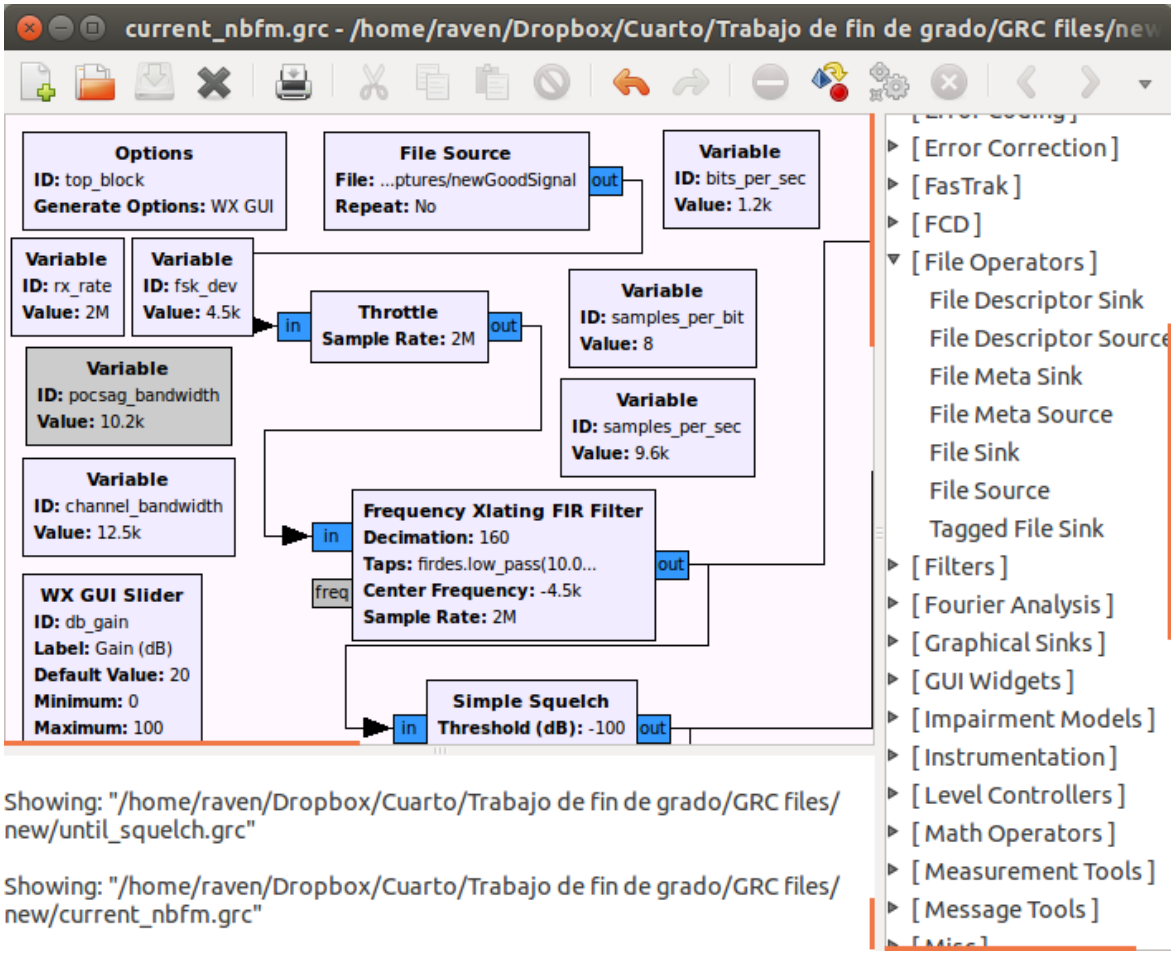
- En 2010, Eric Fry se da cuenta de algo extraño al realizar ingeniería inversa a un *driver* de un dispositivo *USB* para recepción *FM* y *DAB+*. Lo que viaja del dispositivo al PC no es audio, sino muestras de la señal en una etapa intermedia entre la señal de radiofrecuencia y el audio.
- En 2012 nace el proyecto *rtl-sdr*, que proporciona una interfaz para usar estos dispositivos como SDR's (sólo recepción, pero muy asequibles).
- Interés en integrar SDR en la comunidad de pentesting, con nuevas herramientas que permiten inyectar paquetes de diversos protocolos al vuelo.

¿Qué es GNU Radio?

GNU Radio es un entorno de desarrollo *open source* multiplataforma de procesamiento de señales en general, si bien está especializado en SDR, pero no limitado a ello.

- Ofrece una interfaz gráfica, *GRC*, además de las interfaces para *Python* y *C++*. La de *Python* es una envoltura de la de *C++*, y la gráfica una envoltura de la de *Python*.
- La interfaz gráfica sirve para crear diagramas de flujo, con conexiones entre bloques que representan funciones de procesamiento de señales.
- Los bloques pueden ser de entrada o salida, para interactuar con el exterior (parte hardware de SDR, tarjeta de audio, disco duro...), o de entrada y salida, implementando funciones en sí.

GRC



Showing: "/home/raven/Dropbox/Cuarto/Trabajo de fin de grado/GRC files/new/until_squelch.grc"

Showing: "/home/raven/Dropbox/Cuarto/Trabajo de fin de grado/GRC files/new/current_nbfm.grc"

¿Qué es un pager POCSAG?

Al ataque

- Tras escanear en el rango de frecuencias que menciona el *pager* en su parte de atrás, y los avisos que alcanzo a capturar se emiten en la misma frecuencia.
- Construyo el diagrama de flujo (no sin mucho esfuerzo) para decodificar con arreglo al estándar *POCSAG* y efectivamente, se ajusta al estándar. Cada disposivo tiene un ID, y suena cuando se emite el suyo.
- Modifico el diagrama de flujo y creo un bloque personalizado para *GRC* para imprimir en consola los ID's según se capturan los avisos.

Dificultades encontradas

- Dominio completamente nuevo para mí, y falta de base sólida a la hora de resolver los problemas (días de diagnóstico por problema).
- *GRC* no está hecho para aprender a base de prueba y error desde el principio, no es fácil saber qué está fallando ni por qué (curva de aprendizaje elevada).
- Limitaciones del hardware, mi portátil usa *USB 2.0*, lo que limita el ancho de banda capturable de una vez, además de no tener potencia de procesamiento suficiente y descartar muestras si se usaban varias operaciones simultáneamente.

Conclusiones

- Desde el punto de vista económico y humano, es necesario invertir en la seguridad de los sistemas informáticos. No sólo se protege de las malas intenciones, sino de las buenas intenciones equivocadas.
- La "seguridad" por oscuridad no es seguridad, si un sistema necesita que su forma de funcionar no sea pública para ser seguro, no es seguro igualmente. *Sólo la clave debe ser desconocida para el resto*.
- No se le ha prestado suficiente atención a la seguridad de las radiocomunicaciones en el pasado, y ahora se dispone de herramientas basadas en SDR que facilitan aprovecharse de sistemas vulnerables. Hay que prestarle atención desde ya.

Utilidades

- Aprender conocimientos básicos de radio (inquietud personal).
- Incorporar una nueva herramienta de trabajo (es posible que se incorpore en labores de pentesting).
- Servir de guía de inicio rápido a SDR a los investigadores del departamento.
- Obtener el título de Graduado en Ingeniería Informática.

¡Gracias por vuestra atención!