Introducción a la Ingeniería Inversa: aplicaciones e investigación

Ricardo J. Rodríguez

rjrodriguez@unizar.es

tw: @RicardoJRdez – http://www.ricardojrodriguez.es

Universidad de Zaragoza
Zaragoza, Spain

17 de Diciembre, 2012

Diseño de Aplicaciones Seguras - Curso 2012/2013

Máster Universitario en Ingeniería de Sistemas e Informática

Universidad Zaragoza

Introducción a la Ingeniería Inversa (I)

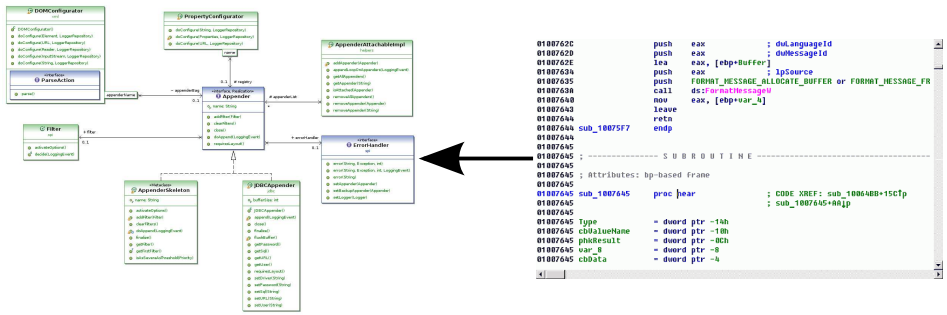
Ingeniería inversa (*reverse engineering*)

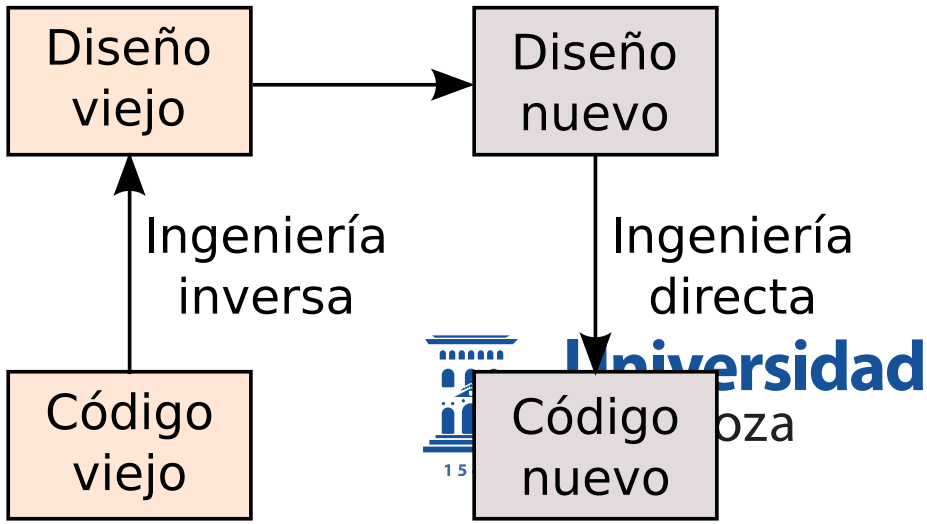
- Descubrir cómo funciona (algo) a partir de un análisis exhaustivo
- Mejora de productos/sistemas viejos
- Diferentes dominios de aplicación
  - *Hardware* (*legacy hardware*)
  - *Software* (e.g. Samba)

Introducción a la Ingeniería Inversa (I)

Ingeniería inversa (*reverse engineering*)

- Descubrir cómo funciona (algo) a partir de un análisis exhaustivo
- Mejora de productos/sistemas viejos
- Diferentes dominios de aplicación
  - *Hardware* (*legacy hardware*)
  - *Software* (e.g. Samba)
- Ir hacia atrás en el ciclo de desarrollo





Introducción a la Ingeniería Inversa (II)

Motivación

- Interoperabilidad
- Documentación inexistente
- Análisis de productos finales
- Auditoría de seguridad
- Espionaje industrial o militar (e.g. Segunda GM)
- Eliminación de anticopias o limitaciones de uso
- Creación de duplicados sin licencia
- Académicos
- Curiosidad innata
- Para aprender de los errores de otros

Zaragoza 1542

Introducción a la Ingeniería Inversa (II): Motivación (2)

Seguridad Informática

Encontrar vulnerabilidades en el software

- Chequeo de cotas de manera incorrecta (*buffer overflow*)
- Uso de entradas sin validación
- Rutinas cíclicas para entrada de datos
- Operaciones de copia a nivel de byte
- Aritmética de punteros basada en entradas dadas del usuario
- "Confianza" en sistemas seguros con entradas dinámicas

Universidad Zaragoza
1542

Introducción a la Ingeniería Inversa (III)

Aproximaciones

- Caja blanca
  - Analizar y entender el código fuente (o binario desensamblado)
  - Encontrar errores de programación y/o implementación
  - Analizador estático: búsqueda de patrones (¿falso positivo?)
  - Ejemplos: *WhiteBox SecureAssistant*, *IDAPro*, *SourceScope*...
- Caja negra
  - Analizar programa según diferentes entradas → software en ejecución
  - No es tan efectivo, pero requiere menos experiencia
  - Método habitual para detectar exploits
- Caja gris
  - Combinación de las dos anteriores
  - Ejecución de programa mediante debug...
  - ...alimentándolo con diferentes entradas
  - Ejemplos: Rational's Purify (análisis uso/consumo memoria), Valgrind

Introducción a la Ingeniería Inversa (III)

Reverse engineering code

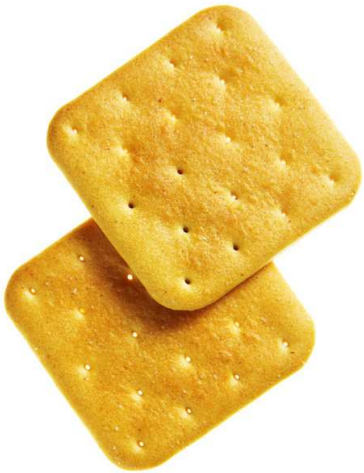
- También conocida como *cracking*
- Eliminar protecciones de código (*copyrights*)
- NO siempre es malo: detección de *bugs*, potenciales *exploits*, ...
- Lucha contra el malware

Universidad Zaragoza
1542

Introducción a la Ingeniería Inversa (III)

Reverse engineering code

- También conocida como *cracking*
- Eliminar protecciones de código (*copyrights*)
- NO siempre es malo: detección de *bugs*, potenciales *exploits*, ...
- Lucha contra el malware
- Crackers: algo más que unas galletas...
  - NO CONFUNDIR con los *criminal hackers*



ersidad
oza

Universidad Zaragoza
1542

Conocimientos previos (I)

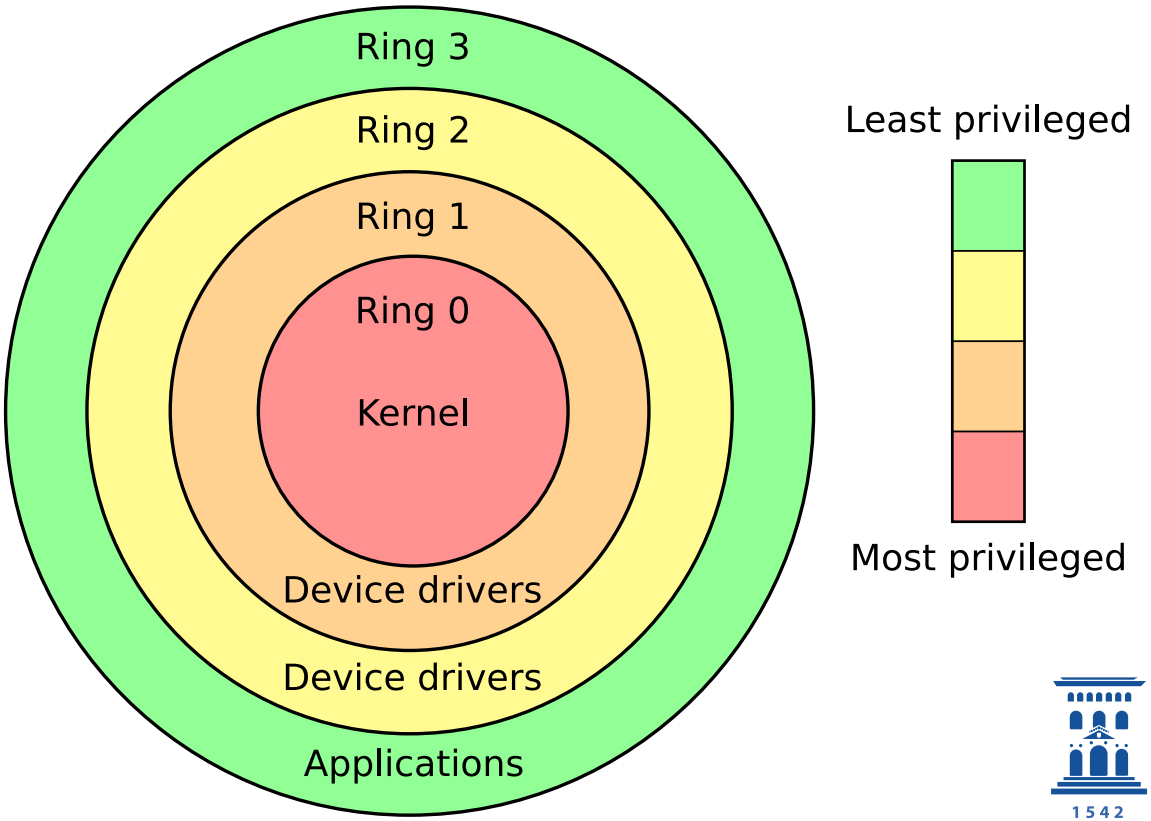
Ensamblador

- `http://www.intel.com/products/processor/manuals/`
- Diferentes tipos de registros, según uso:
  - Propósito general (`eax`, `ebx`, `ecx`, `edx`)
  - Segmento (`cs`, `ds`, `es`, `fs`)
  - Apuntador de instrucciones (`eip`)
  - Índice (`esi`, `edi`)
  - Bandera (*flags*, `CF`, `OF`...)
- Flujo del programa `jmp`, `call`
- La pila: LIFO (*Last In First Out*)
  - Paso de parámetros a procedimientos
  - `push` / `pop`; `call` / `ret`

Conocimientos previos (II)

Sistema Operativo (1)

- Anillos de privilegio $R0 \ldots R3$
- Más privilegios (*kernel*) a menos privilegios (aplicaciones)

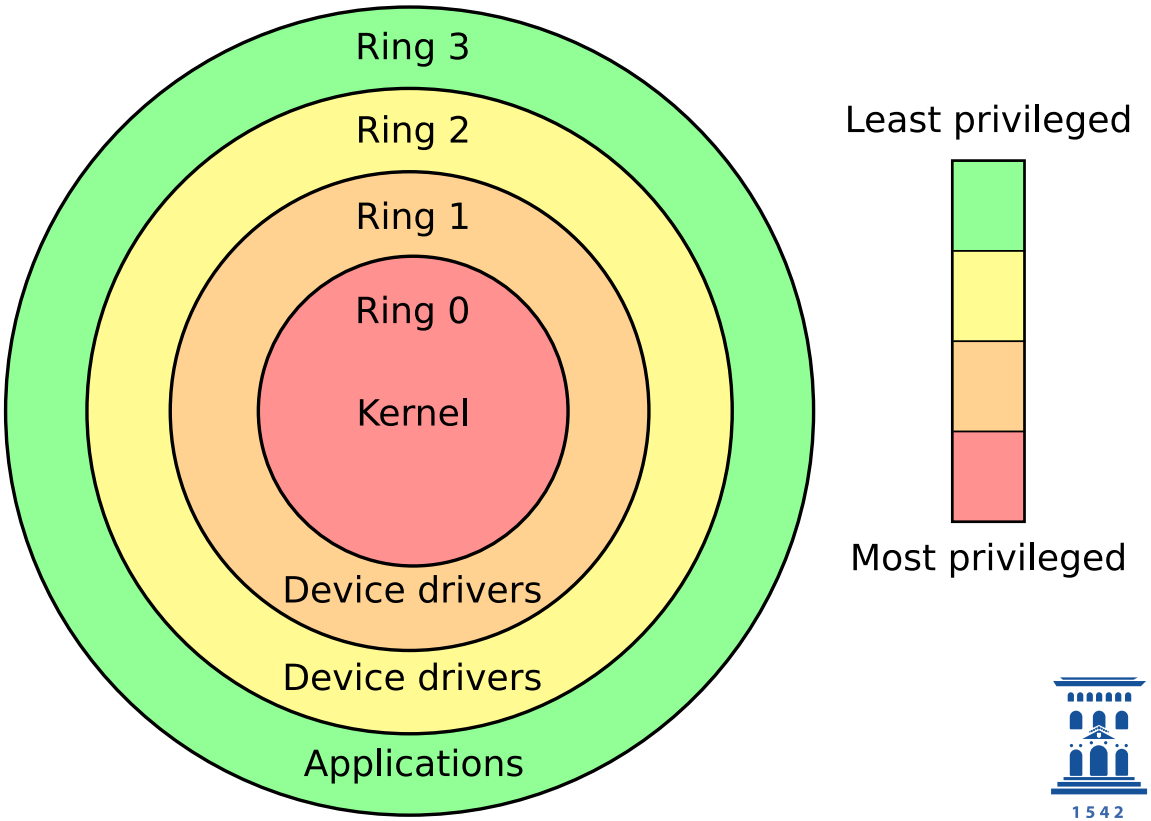


Universidad Zaragoza
1542

Conocimientos previos (II)

Sistema Operativo (1)

- Anillos de privilegio $R0 \ldots R3$
- Más privilegios (*kernel*) a menos privilegios (aplicaciones)



Universidad Zaragoza
1542

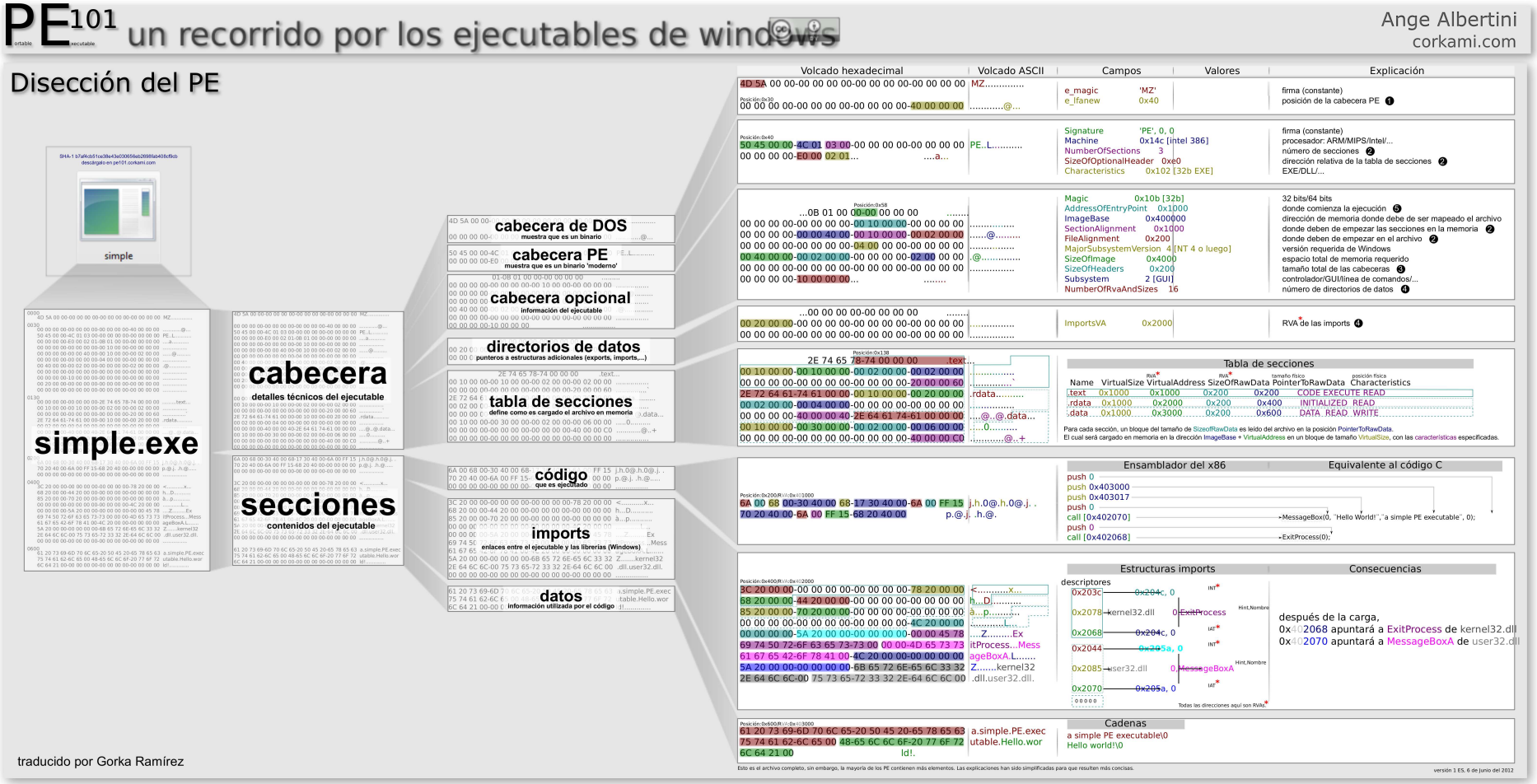
Conocimientos previos (III)

Sistema Operativo (2)

- Funcionamiento interno SS.OO.
  - *¿Qué ocurre al pulsar un botón?*
  - *¿Y al aceptar un* checkbox*?*
  - La biblia de APIs (*Application Programming Interface*) de Windows: WinXXAPI (32 ó 64 bits, `http://msdn.microsoft.com/en-us/library/Aa383723`)
- Estructura interna de un PE (*Portable Executable*)
  - Peering Inside the PE: A Tour of the Win32 Portable Executable File Format (`http://msdn.microsoft.com/en-us/library/ms809762.aspx`)
  - Microsoft PE and COFF Specification (`http://msdn.microsoft.com/en-us/windows/hardware/gg463119.aspx`)
  - The .NET File Format (`http://ntcore.com/files/dotnetformat.htm`)

Conocimientos previos (IV)

Sistema Operativo (3): Estructura PE



Conocimientos previos (V)

Manejo de Debuggers

- Código ensamblador
- Ejecución paso a paso
- Útil para detectar fallos en programas
- Comandos típicos:
  - *Breakpoint*: punto de ruptura
  - *Step into* / *Step over*
  - *Animate into* / *Animate over*
  - Ejecución hasta RET

Universidad Zaragoza
1542

Universidad Zaragoza
1542

Herramientas útiles (I)

Básicas

- Desensambladores
  - General: W32Dasm, IDA Pro
  - Específico: p32Dasm (VBasic), Reflector (.NET)
- Editor hexadecimal (e.g., HexWorkshop)
- Debuggers
  - Soft ICE
  - OllyDBG
  - IDA Pro

Herramientas útiles (II)

Otras...

- Identificadores y editores PE (PEiD, PEEditor)
- Visores/editores de recursos (XNResource Editor, Resource Hacker)
- Volcadores de memoria (LordPE Deluxe, ProcDump, SirPE)

Herramientas útiles (II)

Otras...

- Identificadores y editores PE (PEiD, PEEditor)
- Visores/editores de recursos (XNResource Editor, Resource Hacker)
- Volcadores de memoria (LordPE Deluxe, ProcDump, SirPE)
- Emuladores (HASP, Sentinel)
- Monitores de API (KaKeeware Application Monitor, Event2Address)
- Reparadores de IAT (ImportREC, ReVirgin)

Universidad Zaragoza

Herramientas útiles (III)

Documentación: manuales y tutoriales

- Hay que leer para aprender
- Internet, una herramienta útil y al alcance de cualquiera
  - Grupos de cracking
    - Hispano-hablantes (WkT, CLS, eCh)
    - Extranjeros (RZR, TNT!, ARTeam, RE)
  - Foros
    - elHacker (sección 'Programación→Ingeniería Inversa')
    - ExeTools
    - WoodMan
  - Páginas personales (Karpoff, Shoulck, Saccopharynx, +NCR, AbsSha)
  - Tuts4You (`http://www.tuts4you.com/`)

Herramientas útiles (III)

Documentación: manuales y tutoriales

- Hay que leer para aprender
- Internet, una herramienta útil y al alcance de cualquiera
  - Grupos de cracking
    - Hispano-hablantes (WkT, CLS, eCh)
    - Extranjeros (RZR, TNT!, ARTeam, RE)
  - Foros
    - elHacker (sección 'Programación→Ingeniería Inversa')
    - ExeTools
    - WoodMan
  - Páginas personales (Karpoff, Shoulck, Saccopharynx, +NCR, AbsSha)
  - Tuts4You (`http://www.tuts4you.com/`)
- Práctica, práctica y (un poco más de) práctica
  - Cualquier (pobre) programa que caiga en nuestras manos
  - Crackmes (`http://www.crackmes.us/`)

Universidad Zaragoza

Análisis de código muerto: descripción

- Programas sin protección (o protección mínima)
- Es raro que funcione
- Herramientas necesarias
  - Identificador PE
  - Desensamblador
  - Editor Hexadecimal
- Casos típicos
  - Salto JE/JNE (JZ/JNZ) para registro correcto
  - Número de registro embebido en la aplicación

Universidad Zaragoza

Análisis de código muerto: ejemplos (I)

NOPeo de salto de comprobación

- Una o varias rutinas de comprobación de *serial*
- NOPeo: sustituir código máquina por NOP (No OPeration)
  - JE/JNE (74/75) → NOP (90)
  - JE/JNE (74/75) → JMP (EB)[a]
  - Variantes: JX/JNX (cualquiera) → NOP (90) ó JMP (EB)

[a] Si el salto es largo (destino a más de 32 bits desde el lugar de origen), varía...

```
:004984AF 68488A5300          push 00538A48
:004984B4 E8CCFBFFFF          call 00498085
:004984B9 85C0                test eax, eax
:004984BB 0F8499000000        je 0049855A
:004984C1 BE887C5200          mov esi, 00527C88
:004984C6 BF488A5300          mov edi, 00538A48
:004984CB 33C0                xor eax, eax
:004984CD 83C9FF              or ecx, FFFFFFFF
:004984D0 F2                  repnz
```

Pasos

1. Identificar PE y desensamblar
2. Buscar mensajes de chico malo
3. Analizar camino hasta el mensaje
4. NOPear salto/desviar camino

Análisis de código muerto: ejemplos (II)

A la caza del *serial*

- Una o varias rutinas de comprobación de serial
- El código de registro (*serial*) es único y...
- ...está embebido en la aplicación ¡!

Pasos

1. Identificar PE (¿está protegido?)
2. Desensamblar
3. Buscar mensajes de chico malo
4. Husmear la zona
5. Comprobar cadenas sospechosas }:)

Zaragoza 1542

Análisis de código 'vivo': descripción

- Programas con (o sin) protección
- Herramientas necesarias
  - Identificador PE
  - Desensamblador
  - Debugger
  - (otras?)
- Más complicados (i.e., divertido)
- Cada aplicación es un reto nuevo y diferente
- Casos típicos
  - Mmm... ¿cualquiera?

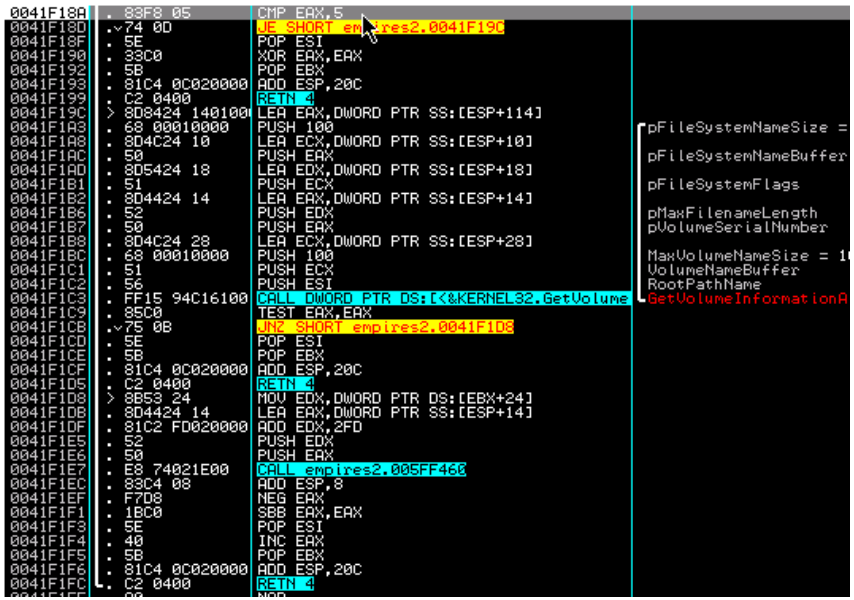
(luego veremos un ejemplo...)

Universidad Zaragoza 1542

Universidad Zaragoza

Técnicas de *cracking* (I): CD Check

- Verificación del CD presente en la unidad
- Fichero concreto en el CD de la unidad (algunas veces)
- Protecciones más avanzadas: SafeDisc, StarForce
- Uso de unidades virtuales: DaemonTools



APIs típicas

- `GetDriveTypeA`
  - $EAX = 5$ si hay CD
- `GetVolumeInformationA`

Universidad Zaragoza
1542

Técnicas de *cracking* (II): *Patching* y *loaders*

Patching

- Objetivo: cambiar flujo natural de ejecución del programa
  - Cambio de instrucciones máquina
    - Modificando (tras un CMP o TEST) o insertando saltos
    - Sustituyendo por NOPs
- Métodos habituales: búsqueda de cadenas o chequeo de APIs
- Cambios estáticos (i.e., permanentes)

Loaders

- Como el *patching*, pero "en caliente" $\rightarrow$ más elegante
- Dos tipos (básicos)
  - Simples
  - *Debuggers* (más complejos): útil para programas empacados
- Cambios dinámicos (i.e., temporales)

Técnicas de *cracking* (III): *Time-trials* y Registro

Time-trials

- Protección por tiempo (uso limitado X días/minutos)
- APIs típicas de chequeo
  - `GetLocalTime`
  - `GetFileTime`
  - `GetSystemTime`

Registro de Windows

- Guardan datos en el Registro de Windows
- APIs típicas
  - `RegCloseKey`
  - `RegCreateKeyEx`
  - `RegOpenKeyEx`
  - `RegSetValueEx`
  - `RegQueryValueEx`

Técnicas de *cracking* (V): Captura del *serial* y *Keygenning*

Captura del *serial*

- Objetivo: conseguir número de registro del programa
- Idéntico para todos los usuarios
- Embebido en la aplicación
- Fácil: búsqueda de cadenas con patrones conocidos...

Keygenning

- Objetivo: encontrar algoritmo de generación de claves
- Complejidad del algoritmo variable
- Cada usuario tiene un número de registro diferente
- Ingeniería inversa pura y dura

Zaragoza 1542

Técnicas de *cracking* (VI): Archivos de licencia

- Se registran mediante archivos de licencia
- Chequeos rutinarios contra servidor de la empresa (a veces)
- APIs típicas
  - Conexión: `connect`, `WSAConnect`
  - Recepción: `recv`, `recvfrom`, `WSARecv`, `WSARecvFrom`, `WSARecvMsg`
  - Envío: `send`, `sendto`, `WSASend`, `WSASendTo`, `WSASendMsg`
- Algunos usan criptografía (i.e., licencia codificada)
  - MUY complicados de conseguir licencia correcta
    $\rightarrow$ Dependerá del algoritmo criptográfico usado
- Solución: intentar parchear...

Universidad Zaragoza

Técnicas de *cracking* (VII): Desempacado (*unpacking*)

- Programas protegidos
- Pueden ser muy complicados (*anti-dumps*, *scrambling*, ...)
- Pasos a realizar
  1. Hallar el OEP (*Original Entry Point*)
     - *Stolen bytes*
     - Cambios en la cabecera PE
  2. Dumpear el proceso de memoria (estará desempacado!)
     - Secciones virtuales
     - Ofuscación de código
  3. Arreglar la IAT (*Import Address Table*)
     - Emulación de APIs
     - Redireccionamiento de APIs
- Lista: `http://en.wikipedia.org/wiki/Executable_compression`
- Existen *unpackers* automáticos: *tools* propias o *scripts*

Universidad Zaragoza
1542

Algunos métodos *antireversing*

- Técnicas anti-debugging
  - APIs
  - Windows *internals tricks*
  - Detección de herramientas
  - Lectura recomendada: `http://pferrie.tripod.com/`
- Técnicas anti-tracing
- Técnicas anti-dumping
- Técnicas de ocultación de OEP
- Otras técnicas:
  - Ofuscamiento de código (código basura)
  - Detección de modificaciones (CRC, APIs)
  - Criptografía

Universidad Zaragoza

Universidad Zaragoza
1542

¿Qué es un *malware*? (I)

Malicious software

- Software creado para dañar a(comprometer) tu ordenador
- Taxonomía malware:
  - Virus/Gusano
  - Backdoor
  - Trojano
  - Rootkits
  - Spyware
  - Hack tools (e.g. `netcat`)

FAQs

- ¿Qué puede hacer un malware?
- ¿Cómo puede entrar?
- ¿Cómo puedo evitarlo?

¿Qué es un *malware*? (II)

¿Qué puede hacer un malware?

Principales objetivos

- Botnets
  - The Lord is my Shepherd
  - E.g.: DDoS, spam...

Universidad Zaragoza
1542

¿Qué es un *malware*? (II)

¿Qué puede hacer un malware?

Principales objetivos

- Botnets
  - The Lord is my Shepherd
  - E.g.: DDoS, spam...
- Robo de información
  - User-content data (files)
  - Privacy data (keyloggers)
  - Pictures (by using webcam)

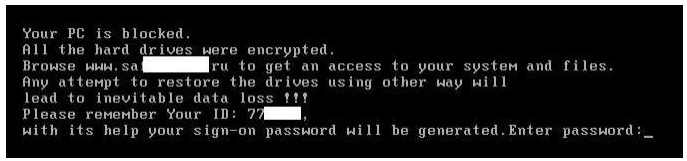
Universidad Zaragoza
1542

¿Qué es un *malware*? (II)

¿Qué puede hacer un malware?

Principales objetivos

- Botnets
  - The Lord is my Shepherd
  - E.g.: DDoS, spam...
- Robo de información
  - User-content data (files)
  - Privacy data (keyloggers)
  - Pictures (by using webcam)
- *Computer-napping* (*ransomware*)
  - BIOS/MBR (Master Boot Record)



Universidad Zaragoza
1542

¿Qué es un *malware*? (II)

¿Qué puede hacer un malware?

Principales objetivos

- Botnets
  - The Lord is my Shepherd
  - E.g.: DDoS, spam...
- Robo de información
  - User-content data (files)
  - Privacy data (keyloggers)
  - Pictures (by using webcam)
- *Computer-napping* (*ransomware*)
  - BIOS/MBR (Master Boot Record)
  - O.S.

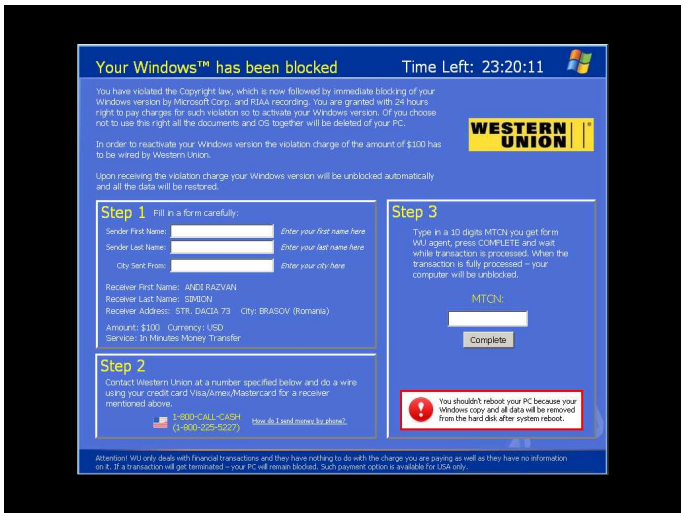
Universidad Zaragoza
1542

¿Qué es un *malware*? (II)

¿Qué puede hacer un malware?

Principales objetivos

- Botnets
  - The Lord is my Shepherd
  - E.g.: DDoS, spam...
- Robo de información
  - User-content data (files)
  - Privacy data (keyloggers)
  - Pictures (by using webcam)
- *Computer-napping* (*ransomware*)
  - BIOS/MBR (Master Boot Record)
  - O.S.



Universidad Zaragoza
1542

¿Qué es un *malware*? (II)

¿Qué puede hacer un malware?

Principales objetivos

- Botnets
  - The Lord is my Shepherd
  - E.g.: DDoS, spam...
- Robo de información
  - User-content data (files)
  - Privacy data (keyloggers)
  - Pictures (by using webcam)
- *Computer-napping* (*ransomware*)
  - BIOS/MBR (Master Boot Record)
  - O.S.

Universidad Zaragoza
1542

¿Qué es un *malware*? (II)

¿Qué puede hacer un malware?

Principales objetivos

- Botnets
  - The Lord is my Shepherd
  - E.g.: DDoS, spam...
- Robo de información
  - User-content data (files)
  - Privacy data (keyloggers)
  - Pictures (by using webcam)
- *Computer-napping* (*ransomware*)
  - BIOS/MBR (Master Boot Record)
  - O.S.
- Fraude (explícito)
  - Extra hits en ads (adware)
  - Porn diallers (modem)
  - Números premium, SMS (móviles)

Universidad Zaragoza
1542

¿Qué es un *malware*? (III)

¿Cómo puede entrar?

- Compartición de ficheros
  - Diskettes? :)
  - USB
  - Internet software...

¿Qué es un *malware*? (III)

¿Cómo puede entrar?

- Compartición de ficheros
  - Diskettes? :)
  - USB
  - Internet software...
- E-mail

¿Qué es un *malware*? (III)

¿Cómo puede entrar?

- Compartición de ficheros
  - Diskettes? :)
  - USB
  - Internet software...
- E-mail
- Redes P2P/torrent

¿Qué es un *malware*? (III)

¿Cómo puede entrar?

- Compartición de ficheros
  - Diskettes? :)
  - USB
  - Internet software...
- E-mail
- Redes P2P/torrent
- IRC (Internet Relay Chat)

Universidad Zaragoza

¿Qué es un *malware*? (III)

¿Cómo puede entrar?

- Compartición de ficheros
  - Diskettes? :)
  - USB
  - Internet software...
- E-mail
- Redes P2P/torrent
- IRC (Internet Relay Chat)
- Bluetooth (móviles)

Universidad Zaragoza

¿Qué es un *malware*? (III)

¿Cómo puede entrar?

- Compartición de ficheros
  - Diskettes? :)
  - USB
  - Internet software...
- E-mail
- Redes P2P/torrent
- IRC (Internet Relay Chat)
- Bluetooth (móviles)
- Android/iOS market (móviles)

Universidad Zaragoza

¿Qué es un *malware*? (III)

¿Cómo puedo evitarlo?

Técnicas de prevención

- Instalar algún AV & anti-spyware
  - Un AV de fiar...

¿Qué es un *malware*? (III)

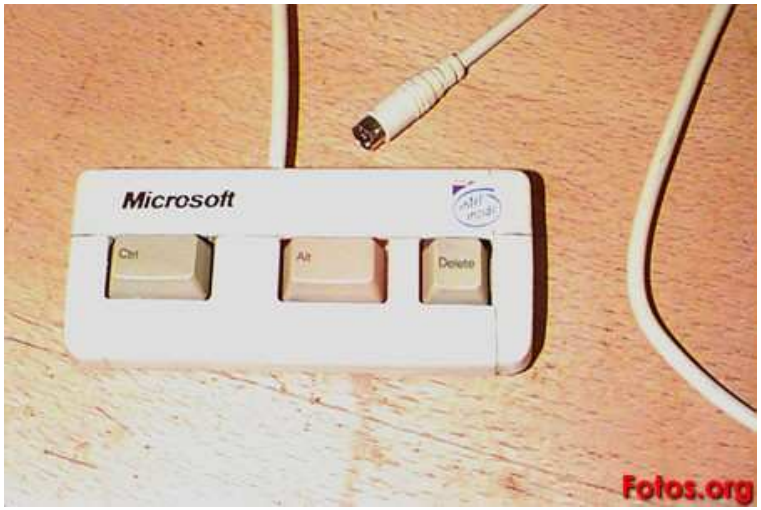
¿Cómo puedo evitarlo?

Técnicas de prevención

- Instalar algún AV & anti-spyware
  - Un AV de fiar...
- Evitar ciertas páginas
  - Cuidado con los ads!

¿Qué es un *malware*? (III)

¿Cómo puedo evitarlo?



Técnicas de prevención

- Instalar algún AV & anti-spyware
  - Un AV de fiar...
- Evitar ciertas páginas
  - Cuidado con los ads!
- Mirar procesos activos
  - Ctrl + Alt + Del (Windows)
  - Apple → "Force Quit"... (MacOS)
  - $ps | aux (MacOS & Linux)

¿Qué es un *malware*? (III)

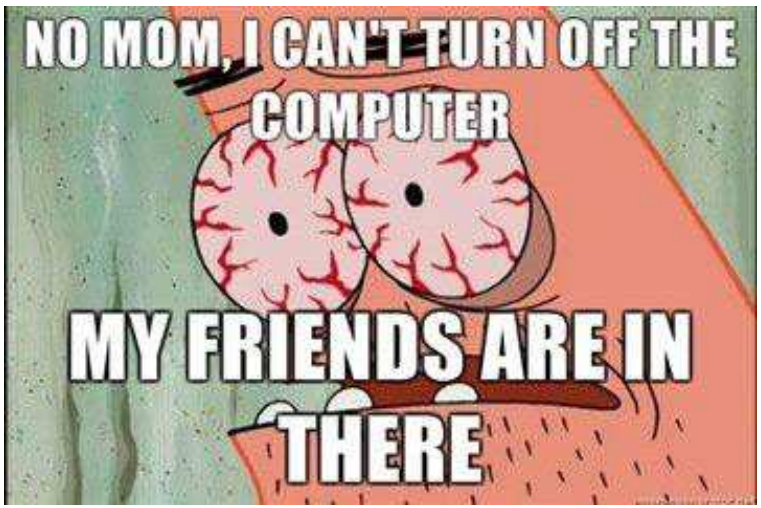
¿Cómo puedo evitarlo?

Técnicas de prevención

- Instalar algún AV & anti-spyware
  - Un AV de fiar...
- Evitar ciertas páginas
  - Cuidado con los ads!
- Mirar procesos activos
  - Ctrl + Alt + Del (Windows)
  - Apple → "Force Quit"... (MacOS)
  - $ps | aux (MacOS & Linux)
- No fiarse de los correos electrónicos!
  - Fotos de tu amigo (qué amigo?)
  - Tienes un tío en Nigeria?
  - Has ganado la lotería y no juegas?

¿Qué es un *malware*? (III)

¿Cómo puedo evitarlo?



Técnicas de prevención

- Instalar algún AV & anti-spyware
  - Un AV de fiar...
- Evitar ciertas páginas
  - Cuidado con los ads!
- Mirar procesos activos
  - Ctrl + Alt + Del (Windows)
  - Apple → "Force Quit"... (MacOS)
  - $ps | aux (MacOS & Linux)
- No fiarse de los correos electrónicos!
  - Fotos de tu amigo (qué amigo?)
  - Tienes un tío en Nigeria?
  - Has ganado la lotería y no juegas?
- Pregunta a tu amigo *computer-geek*

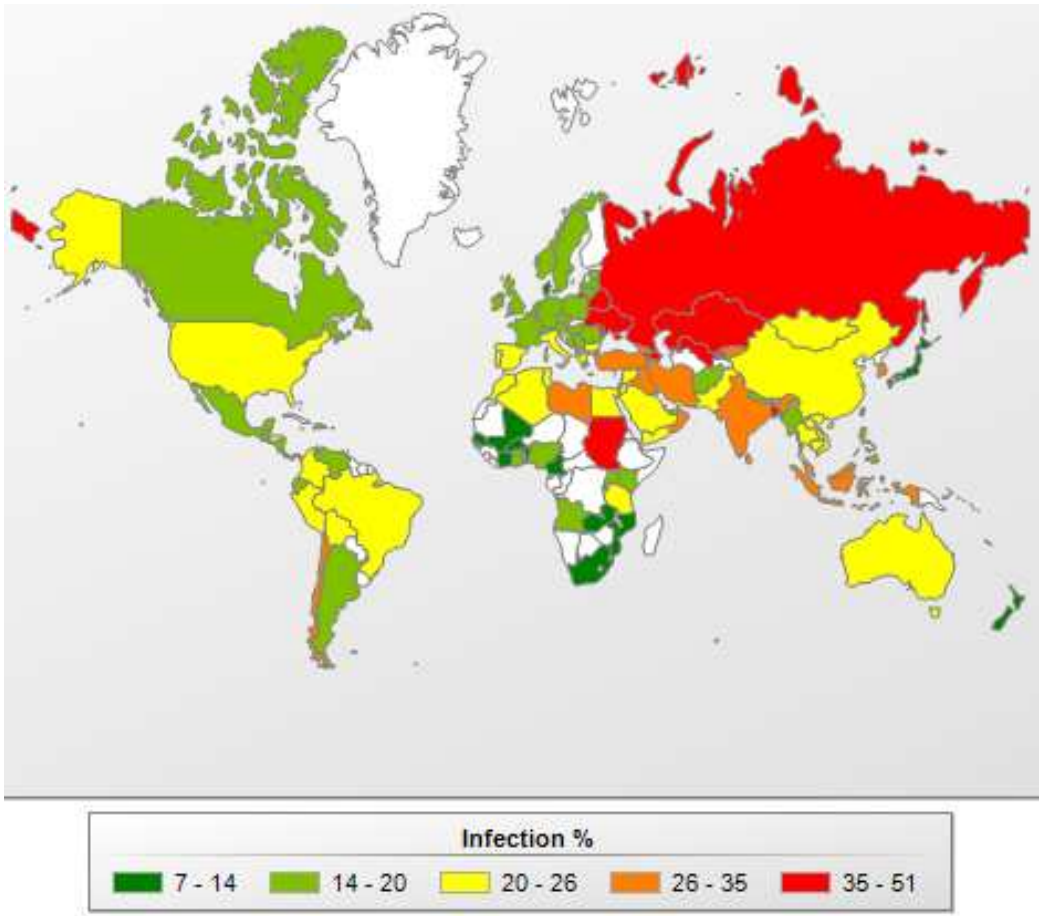
Estadísticas de malware (I)

Software malware threats

(informe de `http://www.threatexpert.com/`, Mayo 2012)

Estadísticas de malware (II)

Infection of web malware threats



(informe mensual de Kaspersky[1], Abril 2012)

[1] De información recogida de sus productos comerciales.

Estadísticas de malware (III)

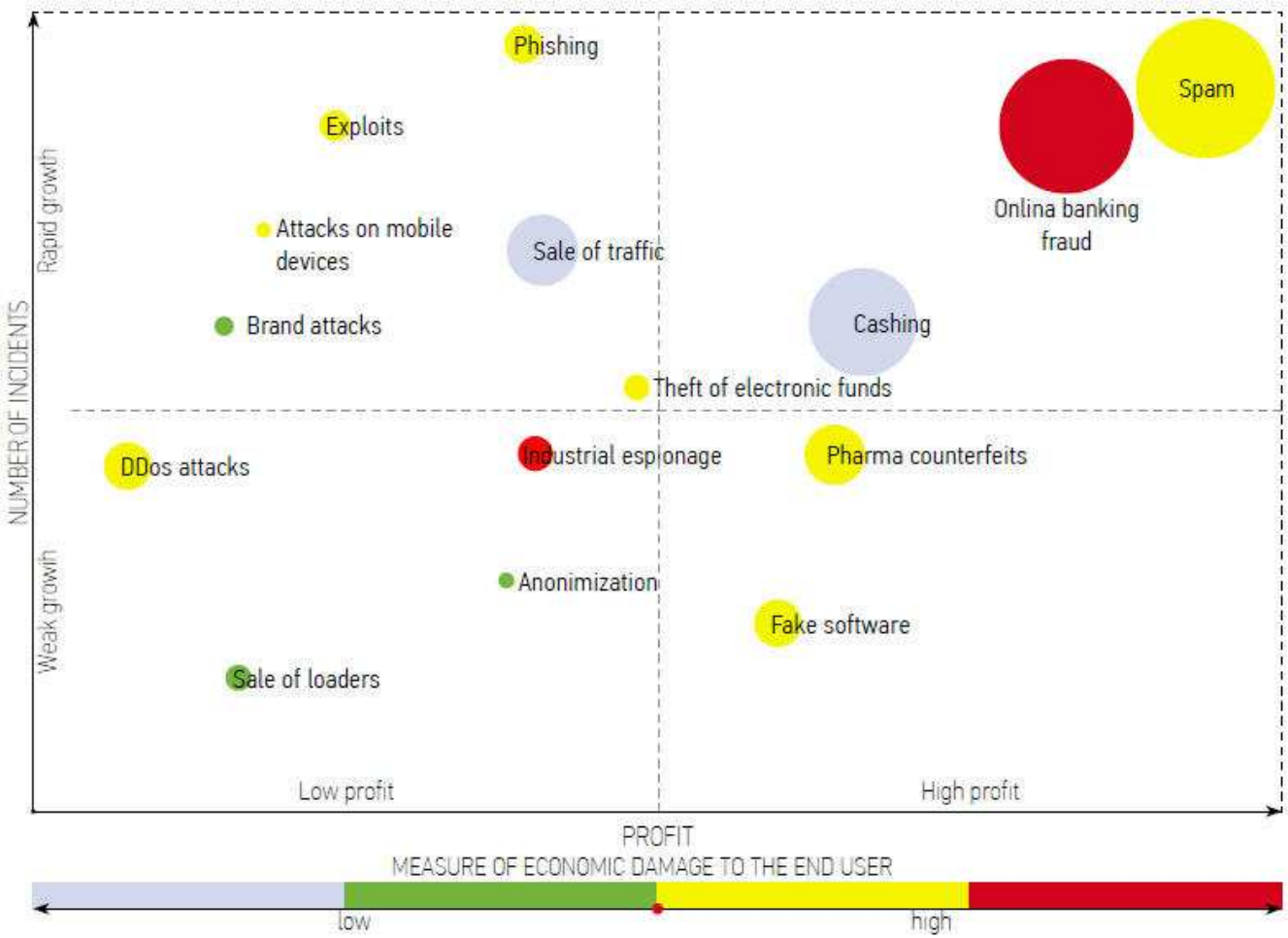
Un mercado muy rentable: estimación de beneficios de 2011

TREND	TOTAL MARKET SHARE	AMOUNT
ONLINE FRAUD		
Online banking fraud	21.3 %	490 million $
Cashing	16 %	367 million $
Phishing	2.4 %	55 million $
Theft of electronic funds	1.3 %	30 million $
Total:	41 %	942 million $
SPAM		
Spam	24 %	553 million $
Pharma and counterfeits	6.2 %	142 million $
Fake software	5.9 %	135 million $
Total:	36.1 %	830 million $
INTERNAL MARKET (C2C)		
Sale of traffic	6.6 %	153 million $
Sale of exploits	1.8 %	41 million $
Sale of loaders	1.2 %	27 million $
Anonymization	0.4 %	9 million $
Total:	10 %	230 million $
DDOS ATTACKS		
DDoS attacks	5.6 %	130 million $
Total:	5.6 %	130 million $
OTHER		
Other	7.3 %	168 million $
Total:	7.3 %	168 million $

(cogida de `http://www.securityaffairs.co/`)

Estadísticas de malware (IV)

Un mercado muy rentable: impacto de daños



(cogida de `http://www.securityaffairs.co/`)

Laboratorio para análisis *malware*

- Máquina virtual (recomendado)
  - Puede que el malware detecte la máquina virtual...
- Disco duro congelado o usar snapshots!
- Software instalado:
  - AVs: si se quiere probar detección in situ
    - Usar servicios tipo VirusTotal, Jotti
  - Herramientas básicas de reversing (debugger, analizador PE, visor APIs...)
  - Analizador de memoria física
  - Analizador de Registro de Windows
- Alternativa: uso de sandboxs tipo Cuckoo (o `http://www.malwr.com`)
- En el host: Wireshark, simuladores de servicios de Internet (DNS, servidor web...)

Fases del análisis

Análisis estático (código muerto)

- Propiedades del PE (TLS? Protegido?)
- Firma MD5, SHA1 $\rightarrow$ búsqueda en VT, Jotti...
- Chequeo de tabla de imports (APIs usadas)
- Comando `strings`

Universidad Zaragoza

Fases del análisis

Análisis estático (código muerto)

- Propiedades del PE (TLS? Protegido?)
- Firma MD5, SHA1 → búsqueda en VT, Jotti...
- Chequeo de tabla de imports (APIs usadas)
- Comando `strings`

Análisis dinámico (código en ejecución)

- Actividad con el S.O. (e.g., `Process Monitor`)
  - ¿Crea ficheros nuevos? → empezamos con análisis estático
  - Claves de registro modificadas
- Actividad con el exterior (e.g., `Wireshark`)
  - Análisis con `whois`, `ipdomaintools`, ... (posible C&C?)
  - Análisis del tráfico de red capturado

Fases del análisis

Análisis estático (código muerto)

- Propiedades del PE (TLS? Protegido?)
- Firma MD5, SHA1 $\rightarrow$ búsqueda en VT, Jotti...
- Chequeo de tabla de imports (APIs usadas)
- Comando `strings`

Análisis dinámico (código en ejecución)

- Actividad con el S.O. (e.g., `Process Monitor`)
  - ¿Crea ficheros nuevos? $\rightarrow$ empezamos con análisis estático
  - Claves de registro modificadas
- Actividad con el exterior (e.g., `Wireshark`)
  - Análisis con `whois`, `ipdomaintools`, ... (posible C&C?)
  - Análisis del tráfico de red capturado

Zaragoza 1542

Inferir patrón de comportamiento

Demostración de análisis *malware*

It's demo time!

Universidad Zaragoza
1542

Conclusiones y otras aplicaciones

- Cualquier protección es *crackeable*
- Mundo de constante evolución → nuevas protecciones, nuevos métodos
- Leer y practicar mucho
- Usar y programar más *software* libre
  - Que no 'hacer' más *software* 'libre' }:)

Otras aplicaciones

- Análisis de malware
- DBI (Dynamic Binary Instrumentation) reversing
- Exploiting

Universidad Zaragoza

Universidad Zaragoza

Investigación en Ingeniería Inversa (I)

- Comparativa de rendimiento con/sin protecciones
- Nuevas técnicas de protección
  - Basadas en Redes de Petri
  - ?
- Búsqueda de nuevas vulnerabilidades con nuevos métodos

Conferencias académicas y revistas

- Working Conference on Reverse Engineering (WCRE), CORE B (2010)
- ACM SIGPLAN Program Protection and Reverse Engineering Workshop (PPREW)
- IEEE Security & Privacy, JCR Q2 (2011)
- Empirical Software Engineering, JCR Q1 (2011)

Investigación en Ingeniería Inversa (II)

DBI reversing

Definición DBI

- Instrumentación: Qué está pasando...
- Dinámica: DURANTE la ejecución...
- (de) Ejecutables: de un binario

Ventajas

- Independiente de lenguaje de programación
- Visión modo máquina
- Instrumentación de software propietario
- No se necesita recompilar/reenlazar cada vez

Desventajas

- Sobrecarga
- $\Downarrow$ rendimiento

Universidad Zaragoza 1542

Investigación en Ingeniería Inversa (III)

DBI: ¿cómo funciona? (I)

- Inserción de código arbitrario durante ejecución de un programa

Código en ejecución

Universidad Zaragoza
1542

Investigación en Ingeniería Inversa (III)

DBI: ¿cómo funciona? (I)

- Inserción de código arbitrario durante ejecución de un programa
- ¿Qué inserto? → función de instrumentación

Código arbitrario

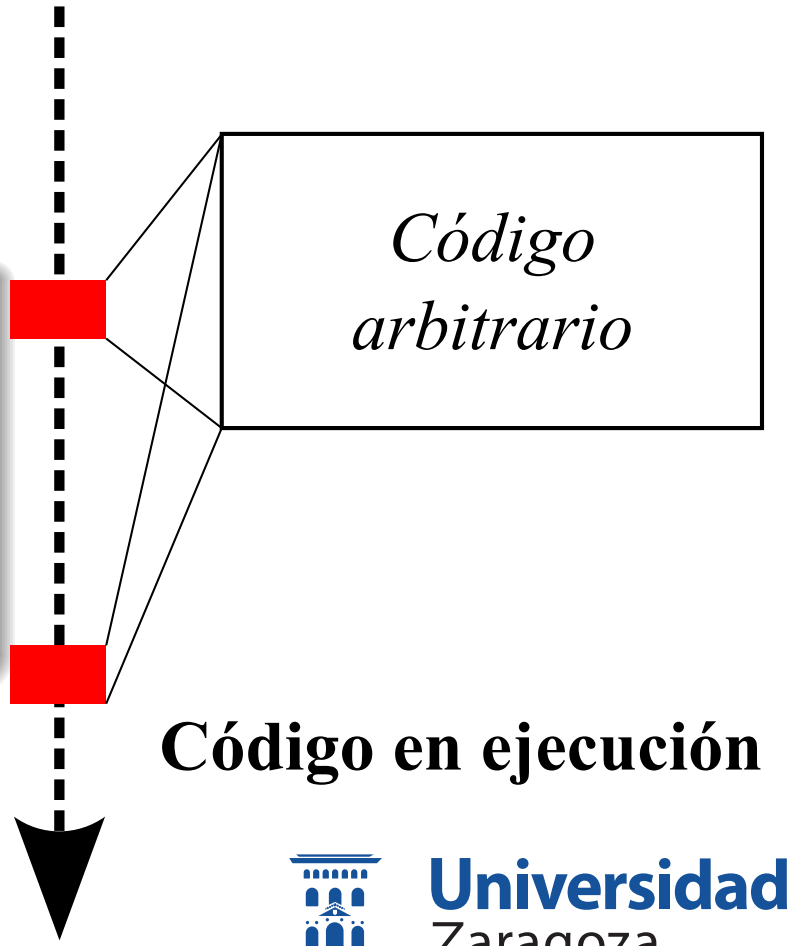
Código en ejecución

Universidad Zaragoza 1542

Investigación en Ingeniería Inversa (III)

DBI: ¿cómo funciona? (I)

- Inserción de código arbitrario durante ejecución de un programa
- ¿Qué inserto? → función de instrumentación
- ¿Dónde? → lugares de inserción



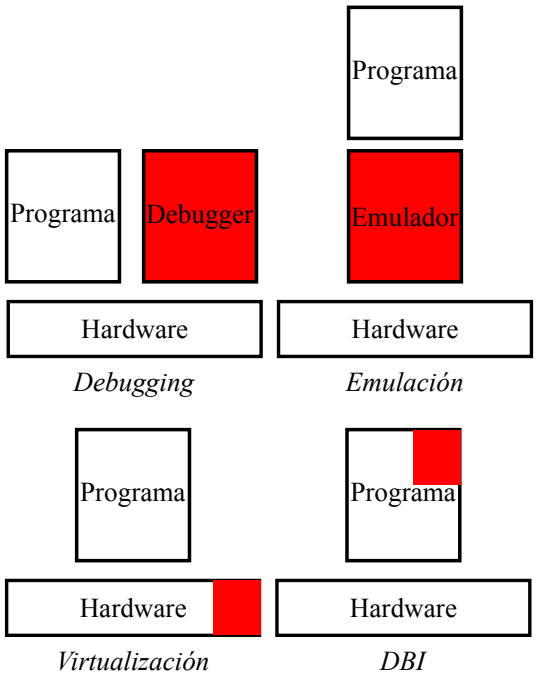
Universidad Zaragoza 1542

DBI: ¿cómo funciona? (II)

DBI en el contexto de análisis dinámico

Definición (informal)

- Transformación del ejecutable
- Control total sobre la ejecución
- Sin necesidad de soporte arquitectural



- Virtualización
  - ¿Control total?
- Emulación
  - ¿Transformación del programa?
- Debugging
  - Soporte arquitectural

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:
  - Generación de trazas

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:
  - Generación de trazas
  - Modelado de predictores de saltos y caches

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:
  - Generación de trazas
  - Modelado de predictores de saltos y caches
  - Recuperación de fallos de memoria

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:
  - Generación de trazas
  - Modelado de predictores de saltos y caches
  - Recuperación de fallos de memoria
  - Emulación de estrategias de especulación

Usos de DBI (I)

Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:
  - Generación de trazas
  - Modelado de predictores de saltos y caches
  - Recuperación de fallos de memoria
  - Emulación de estrategias de especulación
- . . .

Usos de DBI (II)

Usos relacionados con seguridad

- Análisis del flujo de datos (de control)

Usos de DBI (II)

Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades

Usos de DBI (II)

Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing

Universidad Zaragoza

Usos de DBI (II)

Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing
- Monitorización avanzada

Universidad Zaragoza

Usos de DBI (II)

Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing
- Monitorización avanzada
- Ingeniería Inversa

Universidad Zaragoza

Usos de DBI (II)

Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing
- Monitorización avanzada
- Ingeniería Inversa
- Monitorización de privacidad

Universidad Zaragoza

Usos de DBI (II)

Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing
- Monitorización avanzada
- Ingeniería Inversa
- Monitorización de privacidad
- Sandboxing

Universidad Zaragoza

Usos de DBI (II)

Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing
- Monitorización avanzada
- Ingeniería Inversa
- Monitorización de privacidad
- Sandboxing
- . . .

Universidad Zaragoza

Una pequeña demo de DBI reversing (I)

It's demo time (2)!

Aplicación DBI para detección de *buffer overflow*

Una pequeña demo de DBI reversing (II): Buffer Overflow

Demo: `ProtectRetAddrDBA.dll`

Vulnerabilidad

- CWE-120 (`http://cwe.mitre.org/data/definitions/120.html`)
- Copia a buffers sin restricciones → ejecución de código arbitrario
- *"Buffer overflows often can be used to execute arbitrary code [...]. Buffer overflows generally lead to crashes [...]."*

DBA desarrollada con Pin (`ProtectRetAddrDBA.dll`)

- ¿Dónde? → toda `CALL` (antes) o `RETN` (antes) en la sección `.text`
- ¿Qué?
  - `CALL` → guarda dirección legítima de retorno ($EIP + size(CALL)$)
  - `RETN` → si no está en la lista raro...
- Detectados 6 cambios de en librería `ntdll.dll`!!

Zaragoza

Una pequeña demo de DBI reversing (II): Buffer Overflow

Demo: `ProtectRetAddrDBA.dll`

Vulnerabilidad

- CWE-120 (`http://cwe.mitre.org/data/definitions/120.html`)
- Copia a buffers sin restricciones → ejecución de código arbitrario
- *"Buffer overflows often can be used to execute arbitrary code [...]. Buffer overflows generally lead to crashes [...]."*

DBA desarrollada con Pin (`ProtectRetAddrDBA.dll`)

- ¿Dónde? → toda `CALL` (antes) o `RETN` (antes) en la sección `.text`
- ¿Qué?
  - `CALL` → guarda dirección legítima de retorno ($EIP + size(CALL)$)
  - `RETN` → si no está en la lista raro...
- Detectados 6 cambios de en librería `ntdll.dll`!!

Recuerda: Hacer demo...

Introducción a la Ingeniería Inversa: aplicaciones e investigación

Ricardo J. Rodríguez

rjrodriguez@unizar.es

tw: @RicardoJRdez – http://www.ricardojrodriguez.es

Universidad de Zaragoza
Zaragoza, Spain

17 de Diciembre, 2012

Diseño de Aplicaciones Seguras - Curso 2012/2013

Máster Universitario en Ingeniería de Sistemas e Informática