Análisis de rendimiento y niveles de protección de protectores de software

Proyecto Fin de Carrera de Ingeniería en Informática

María Asunción Bazús Castán

Director: Ricardo J. Rodríguez
Ponente: José Merseguer

17 Marzo de 2014

Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Contenidos

Universidad Zaragoza
1542

Contenidos

Universidad Zaragoza
1542

Introducción (I): Ingeniería Inversa

¿Qué es?

Análisis de un programa a partir de su código binario

- **Análisis Estático** $\Downarrow$ No se ejecuta el código. $\Uparrow$ Se exploran todos los caminos
- **Análisis Dinámico** $\Uparrow$ Se ejecuta el código. $\Downarrow$ Sólo se explora uno de los posibles caminos

Universidad Zaragoza 1542

Introducción (I): Ingeniería Inversa

¿Qué es?

Análisis de un programa a partir de su código binario

- **Análisis Estático** $\Downarrow$ No se ejecuta el código. $\Uparrow$ Se exploran todos los caminos
- **Análisis Dinámico** $\Uparrow$ Se ejecuta el código. $\Downarrow$ Sólo se explora uno de los posibles caminos

Usos de la Ingeniería Inversa

- ✓ Analizar o encontrar errores en un programa **(búsqueda de bugs)**
- ✓ Conseguir interoperabilidad con otro sistema **(sistemas legados)**
- ✓ Detectar un programa potencialmente malicioso **(anti-virus)**
- X Detectar vulnerabilidades para crear software malicioso
- X Conseguir copia ilegal de un programa

Introducción (II): Herramientas de Ingeniería Inversa

Análisis Estático

- **Desensambladores**
  - Código máquina ⇒ código ensamblador. (IDAPro)
- **Decompiladores**
  - Código de bajo nivel ⇒ código de alto nivel

Análisis Dinámico

- **Depuradores** (*Debuggers*)
  - Tracear la ejecución. Incluir desensamblador. Puntos de ruptura. Visualizar registros, memoria y pila de ejecución (**OllyDbg**, WinDbg, IDAPro)

Otros

- Dumpers (permiten volcar la imgen en memoria de un proceso). Editor PE. Reconstructor de la tabla de importaciones

Introducción (III): Anti-Ingeniería Inversa

¿Qué es?

Técnicas para evitar el análisis estático y dinámico de un programa

- ⇑ Dificultan los ataques de ingeniería inversa
- ⇓ Impacto en el rendimiento de un programa (tiempo de ejecución, memoria consumida)

Usos de la Anti-Ingeniería Inversa

- ✓ Proteger un programa antes de ser distribuido
- X Evitar que un software malicioso sea detectado por un anti-virus

Universidad Zaragoza 1542

Introducción (IV): Herramientas Anti-Ingeniería Inversa

Protectores de Software: ¿Qué son?

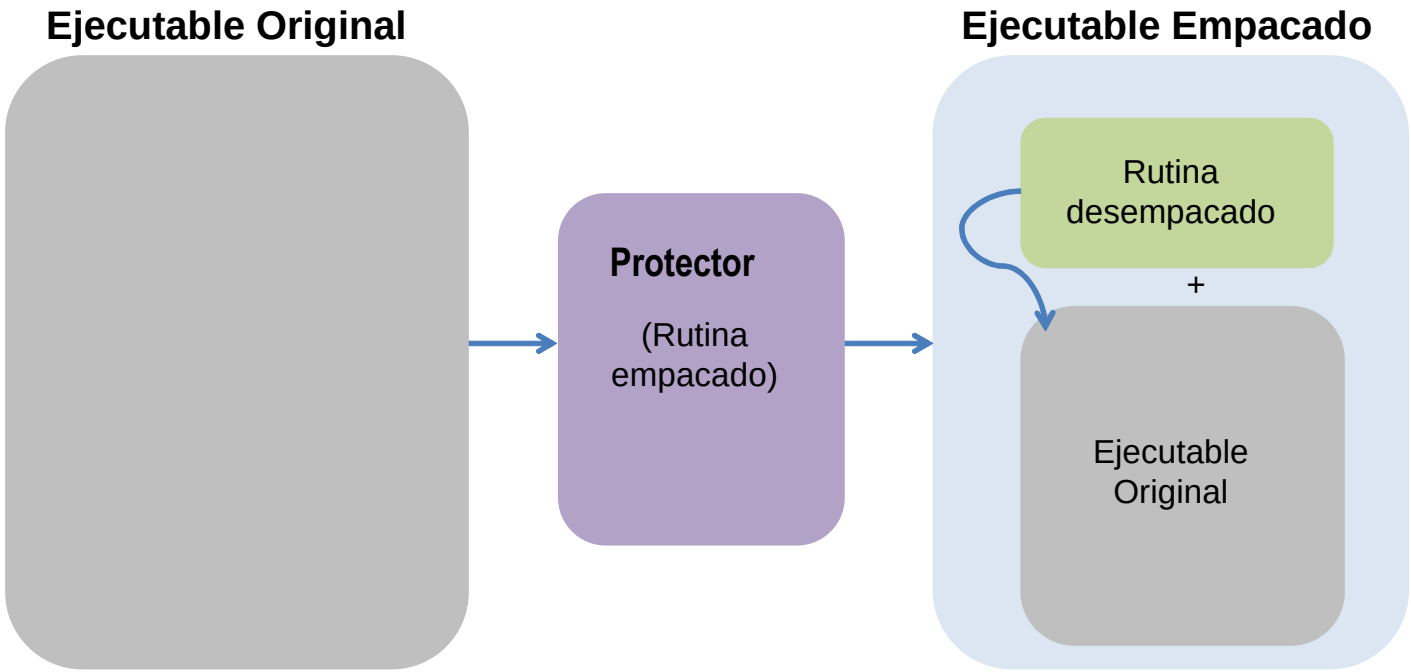
- Herramientas de **protección de ejecutables** (*Software Packers)*
- En un origen su objetivo era solamente comprimir el código (**compresores**)
- Evolucionaron para incluir diferentes técnicas de protección anti-ingeniería inversa (**protectores**)
- Utilizados principalmente en el **entorno Windows**

Universidad Zaragoza 1542

Introducción (V): Herramientas Anti-Ingeniería Inversa

Protectores de Software: ¿Cómo funcionan?

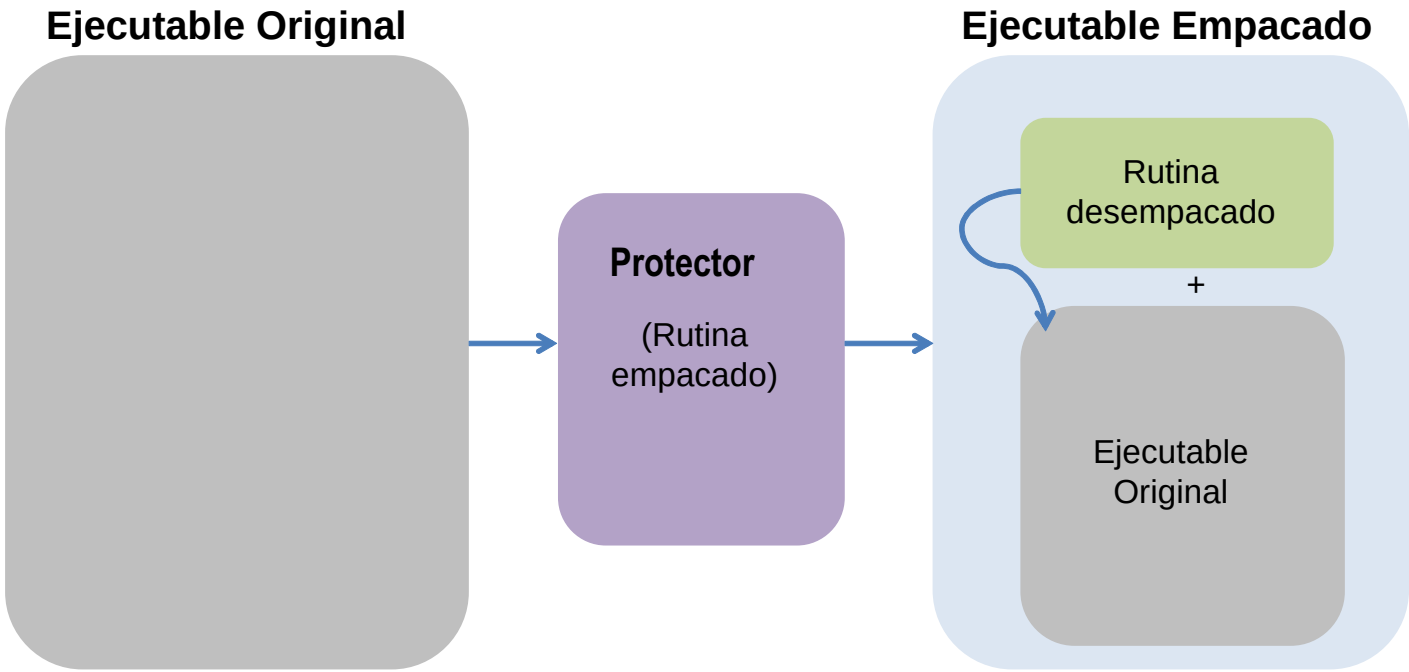
- Ejecutable Empacado = Ejecutable Original + Rutina desempacado



Introducción (V): Herramientas Anti-Ingeniería Inversa

Protectores de Software: ¿Cómo funcionan?

- Ejecutable Empacado = Ejecutable Original + Rutina desempacado



Protectores de Software: Desprotección

Encontrar fin rutina desempacado ⇒ Volcado de memoria a disco

Introducción (VI): Objetivo del PFC

Evaluación cualitativa y cuantitativa de protectores de software

Hitos intermedios:

- Taxonomía de las técnicas de protección
- Selección de protectores para el estudio
- Creación de un benchmark
- Evaluación cualitativa: análisis protecciones
- Evaluación cuantitativa: fiabilidad y rendimiento

Universidad Zaragoza 1542

Trabajo relacionado

- ∄ análisis comparativo de protectores de software actuales
- Métodos de desempacado automático para analizar malware: [RHD+06], [KPY07], [MCJ07],[GFC08], [JCL+10]
- En [RG14] método de análisis de malware basado en DBI
- **Trabajo más cercano:** [KLC+10] rendimiento protectores para sistemas embebidos en Linux

Trabajo relacionado

- ∄ análisis comparativo de protectores de software actuales
- Métodos de desempacado automático para analizar malware: [RHD+06], [KPY07], [MCJ07],[GFC08], [JCL+10]
- En [RG14] método de análisis de malware basado en DBI
- **Trabajo más cercano:** [KLC+10] rendimiento protectores para sistemas embebidos en Linux

Contribución

- Categorización protecciones de software
- Comparativa protectores de software actuales
  - **Evaluación cualitativa**: análisis de protecciones
  - **Evaluación cuantitativa**: fiabilidad y rendimiento (Tiempo de ejecución, Consumo Memoria)

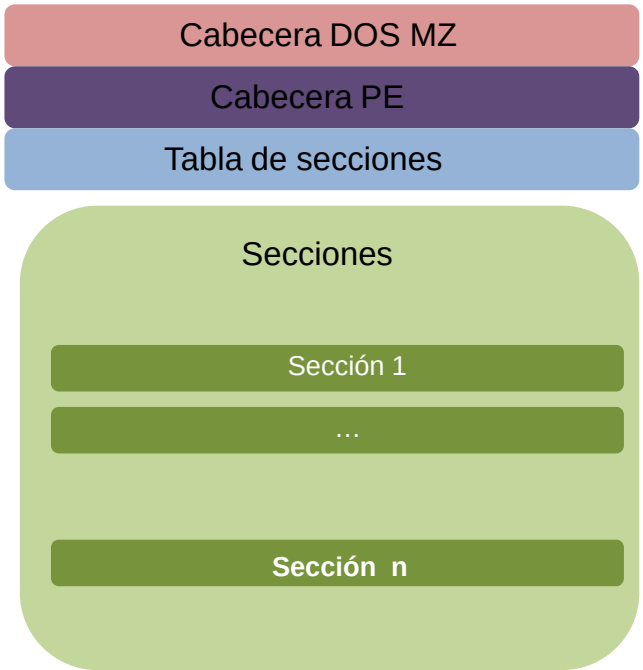
Contenidos

Universidad Zaragoza 1542

Conocimientos previos (I): Formato PE

¿Qué es?

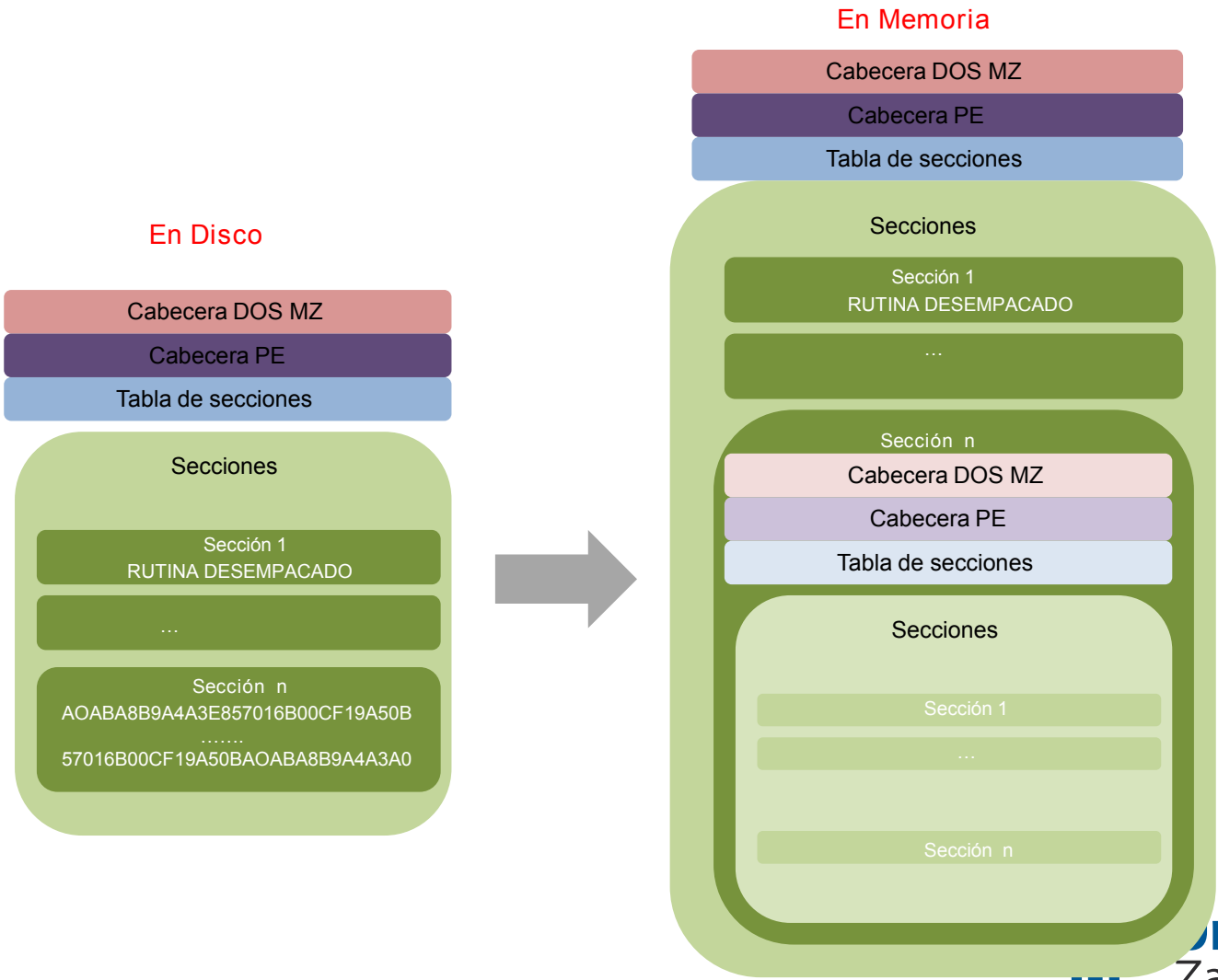
- Formato estándard de Windows (.exe, .dll, .sys,...)
- Cabecera (características) + Secciones (datos y código)



- **Cabecera DOS**:
  - *e_lfnew* offset Cabecera PE
- **Cabecera PE**:
  - *ImageBase*
  - *AddressOfEntryPoint*
  - *DataDirectory[1]*

Universidad Zaragoza 1542

Conocimientos previos (II): Proceso de carga



Contenidos

Universidad Zaragoza 1542

Anti-Análisis Estático

Protección Copia

- **Objetivo**: impedir la copia de un programa
- **Técnicas**: Gestión de Licencias, Clave registro, Hardware

Compresión y Cifrado

- **Objetivo**: impedir el análisis estático del código
- **Técnicas**: comprimir o cifrar el código original, comprimir recursos

Anti Desensamblado

- **Objetivo**: dificultar el proceso de desensamblado del código original
- **Técnicas**: insertar código basura (*junk code)* o código que nunca se ejecuta (*dummy code)*, permutación del código

Anti Análisis Dinámico (I) - Anti Debugging

Anti-Debugging

- **Objetivo**: impedir que un programa pueda ser debuggeado
- **Técnicas**:
  - **Basadas en APIs**: utilizan funciones del SO
  - **Procesos e Hilos**: utilizan información del proceso o hilo de ejecución
  - **Hardware y Registros**: utilizan los registros de procesador
  - **Tiempo y Latencia**: tiempo entre instrucciones
  - **Basadas en Excepciones**: basadas en el funcionamiento del manejo de excepciones

Universidad Zaragoza 1542

Anti Análisis Dinámico (II) - Anti Dumping

Anti Dumping

- **Objetivo**: impedir que volcado de memoria funcione correctamente
- **Técnicas**:
  - **Ofuscación Tabla de Importaciones**
  - **Nanomites**: sustituir los saltos por INT3
  - **Robo de Bytes**
  - **Páginas Guarda**
  - Virtualización: reescribir el código para que sea ejecutado por la máquina virtual embebida. El código original no está disponible en memoria en ningún momento

Universidad Zaragoza 1542

Anti Análisis Dinámico (III): Anti-VM y Anti-Patching

Anti-VM

- **Objetivo**: Impedir ejecución en máquinas virtuales
- **Técnicas**: técnicas de detección de máquinas virtuales específicas (p. ej VMWare, VirtualBox)

Anti Patching

- **Objetivo**: impedir modificación del código
- **Técnicas**: Comprobar la integridad mediante CRC o código hash

Universidad Zaragoza 1542

Contenidos

Universidad Zaragoza 1542

Selección de Protectores para el estudio

	Versión	Última Release	Licencia	Tipo
ACProtect	2.1.0	09/2008	149$	Compresor&Protector
Armadillo	9.62	06/2013	aprox. 299$	Compresor&Protector
ASPack	2.32	08/2012	34€- 208€	Compresor&Protector
ASProtect	1.69	09/2013	117€- 347€	Compresor&Protector
Enigma	3.7	06/2013	149$- 299$	Protector&Virtualizador
EXE Stealth	4.19	06/2011	296,31€	Compresor&Protector
ExeCryptor	2.3.9	03/2006	135$- 749$	Compresor&Protector
ExPressor	1.8	01/2010	14$- 360$	Compresor&Protector
FSG	2.0	05/2004	Gratuita	Compresor
MEW	11	11/2004	Gratuita	Compresor
Obsidium	1.5	10/2013	139€- 289€	Protector&Virtualizador
PECompact	3.02.2	02/2010	89,95$499$	Compresor&Protector
PELock	1.06	01/2012	89$- 289$	Compresor&Protector
PESpin	1.33	05/2011	Gratuita	Compresor&Protector
Petite	2.3	02/2005	170€	Compresor&Protector
Smart Packer	1.9	06/2013	59€- 399€	Compresor& Bundler
TeLock	0.98	08/2001	Gratuita	Compresor&Protector
Themida	2.2	10/2013	199€- 399€	Protector&Virtualizador
UPX	3.91	02/2013	Gratuita	Compresor

Contenidos

Universidad Zaragoza
1542

Evaluación Cualitativa (I): Anti-Análisis Estático

Protector	Compresión	Cifrado	Anti-Desensamblado
ACProtect	✓	–	✓
Armadillo	✓	✓	✓
ASPack	✓	✓	–
ASProtect	✓	✓	✓
Enigma	✓	✓	✓
EXE Stealth	✓	–	–
ExeCryptor	✓	–	✓
ExPressor	✓	–	–
FSG	✓	–	–
MEW	✓	✓	–
Obsidium	✓	✓	✓
PECompact	✓	–	–
PELock	✓	–	–
PESpin	✓	✓	✓
Petite	✓	–	–
SmartPacker	✓	✓	–
TeLock	✓	–	–
Themida	✓	✓	✓
UPX	✓	–	–
VMProtect	✓	–	✓
Yodas Protector	✓	–	✓

iversidad
agoza

Evaluación Cualitativa (II): Anti-Análisis Dinámico

Protector	Anti-Debugging	Anti-Dumping	Anti-VM	Anti-Patching
ACProtect	✓	✓	–	–
Armadillo	✓	✓	–	✓
ASPack	✓	–	–	–
ASProtect	✓	✓	–	✓
Enigma	✓	✓(Alto)	✓	✓
EXE Stealth	✓	✓	✓	✓
ExeCryptor	✓(Alto)	✓(Alto)	✓	✓
ExPressor	✓	✓	✓	✓
FSG	–	–	–	–
MEW	–	–	–	–
Obsidium	✓(Alto)	✓(Alto)	✓	✓
PECompact	✓	✓	–	✓
PELock	✓	✓	–	–
PESpin	✓	✓	–	–
Petite	✓	✓	–	–
SmartPacker	–	–	–	–
TeLock	✓	✓	–	–
Themida	✓(Alto)	✓(Alto)	✓	✓
UPX	–	–	–	–
VMProtect	✓	✓(Alto)	✓	✓
Yodas Protector	✓	✓	–	–

lad

Contenidos

Universidad Zaragoza 1542

Evaluación Cuantitativa (I): Creación de benchmark

Aplicaciones seleccionadas

Nombre	Versión	Lenguaje	Tipo	Tipo de Cálculo	Origen
GNU go	3.8	C	IA - Juegos	Entero	SPEC CINT 2006
hmmer	3.0	C	Genética	Entero	SPEC CINT 2006
h264ref	18.5	C	Compresión video	Entero	SPEC CINT 2006
mcf	1.3	C	Combinatoria	Entero	SPEC CINT 2006
namd	2.8	C++	Biología, simulación de moléculas	Real	SPEC CFP 2006
povray	3.7	C++	Renderización	Real	SPEC CFP 2006
calculix	2.6	Fotran90 & C	Mecánica Estructural	Real	SPEC CFP 2006
palabos	1.4.	C++	Dinámica de fluidos	Real	Propio
aescrypt	3.0.9	C	Criptografía, cifrado	E/S	Propio
bzip2	1.0.5	C	Compresión	E/S	SPEC CINT 2006
ffmpeg	0.10	C	Conversión de formatos video/audio	E/S	Phoronix Test Suite
md5	1.2	Delphi 7	Criptografía, Hash	E/S	Propio

Universidad Zaragoza

Evaluación Cuantitativa (II): Creación de benchmark

Algoritmo del benchmark

1: $aplicacion = [bzip2, gnugoo, hmmer, h264ref, mcf, namd, povray...]$
2: $protector = [ACProtect, ASPack, Armadillo, Enigma, EXEStealth, ...]$
3: **for** $repeticion = 1$ to 45 **do**
4: **for** *aplicacion* **do**
5: ejecutar *aplicacion* {Ejecutable original}
6: medicion proceso *GetProcessTimes*(); *GetProcessMemoryInfo*()
7: **end for**
8: **for** *protector* **do**
9: **for** *aplicacion* **do**
10: ejecutar *aplicacion.protector* {Ejecutable protegido}
11: medicion proceso *GetProcessTimes*(); *GetProcessMemoryInfo*()
12: **end for**
13: **end for**
14: **end for**

Universidad Zaragoza 1542

Evaluación Cuantitativa (III): Fiabilidad

	Cálculo entero	Cálculo real	Gran demanda E/S	Total
ACProtect	25%	50%	75%	50%
Armadillo	50%	100%	75%	75%
ASPack	100%	100%	75%	91%
ASProtect	100%	100%	75%	91%
Enigma	75%	75%	100%	83%
EXE Stealth	100%	75%	100%	91%
ExeCryptor	25%	50%	0%	25%
ExPressor	100%	100%	100%	100%
FSG	0%	50%	100%	50%
MEW	100%	50%	100%	50%
Obsidium	100%	100%	100%	100%
PECompact	25%	100%	100%	75%
PELock	25%	100%	75%	66%
PESpin	0%	50%	75%	41%
Petite	50%	25%	50%	41%
Smart Packer	50%	75%	100%	75%
TeLock	25%	50%	50%	41%
Themida	25%	100%	100%	75%
UPX	100%	100%	100%	100%
VMProtect	100%	100%	100%	100%
YodasProtector	25%	25%	50%	33%
	61%	**76%**	**83%**	

dad

Evaluación Cuantitativa (IV): Fiabilidad

Consideraciones

- **Obsidium, EXEStealth**: ejecutables protegidos proporcionados por el mismo fabricante
- **FSG, MEW, PESPin, Petite, TELock, YodasProtector**: Error "*Runtime error: R6002 floating point support not loaded*" $\Rightarrow$ Programas compilados con MSVC++5 o superior
- **ACProtect, Armadillo, PELock, PESpin**: ejecutable correcto sin todas las protecciones
- **VMProtect, Themida, EXECryptor, Armadillo y PELock**: se ha trabajado con la versión demo

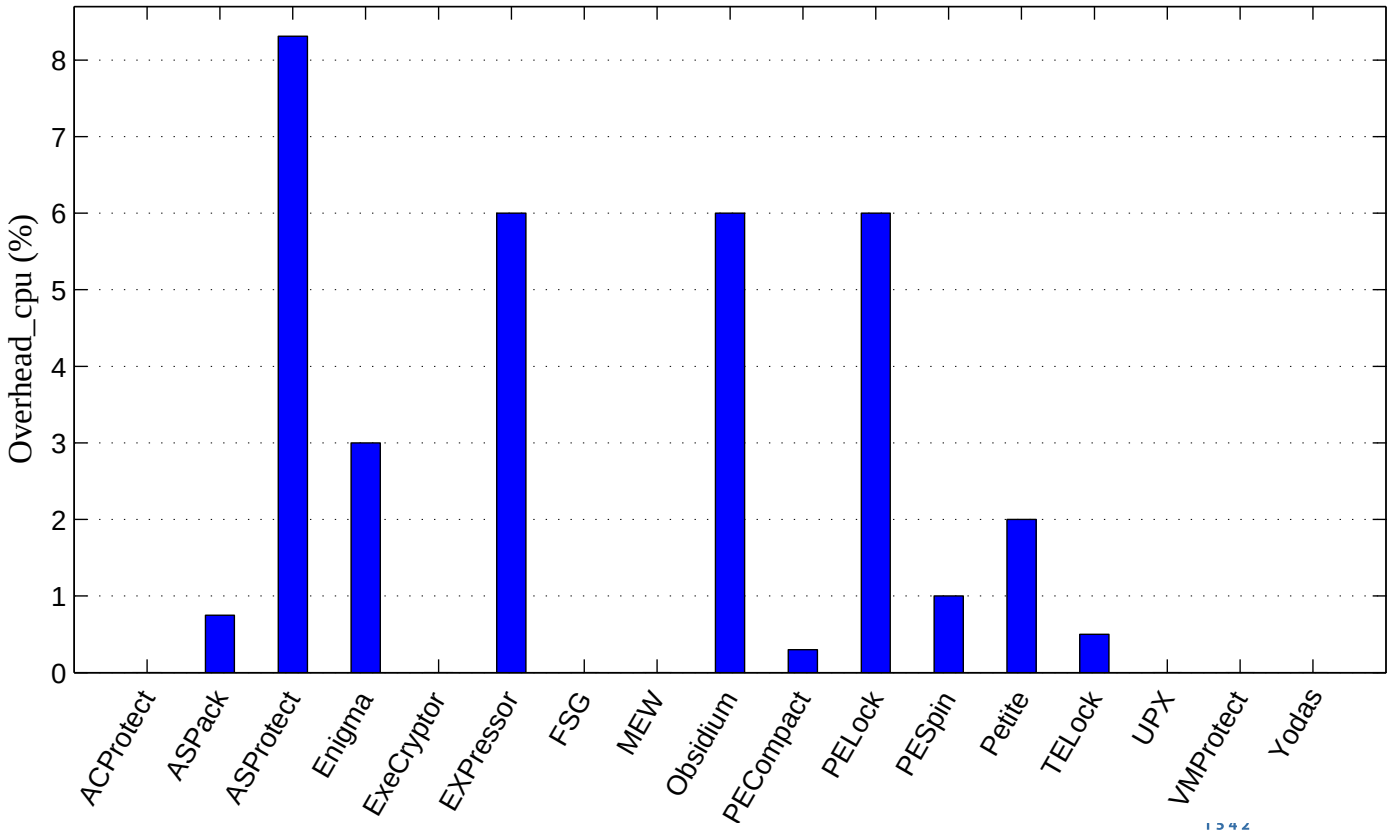
Universidad Zaragoza 1542

Evaluación Cuantitativa (V): Rendimiento

Resultados Overhead CPU

0% **UPX,FSG, MEW** (sólo compresores). **EXECryptor, VMProtect** (versión demo). **ACProtect, YodasProtector**

6-8% ASProtect, EXPressor, Obsidium, PELock



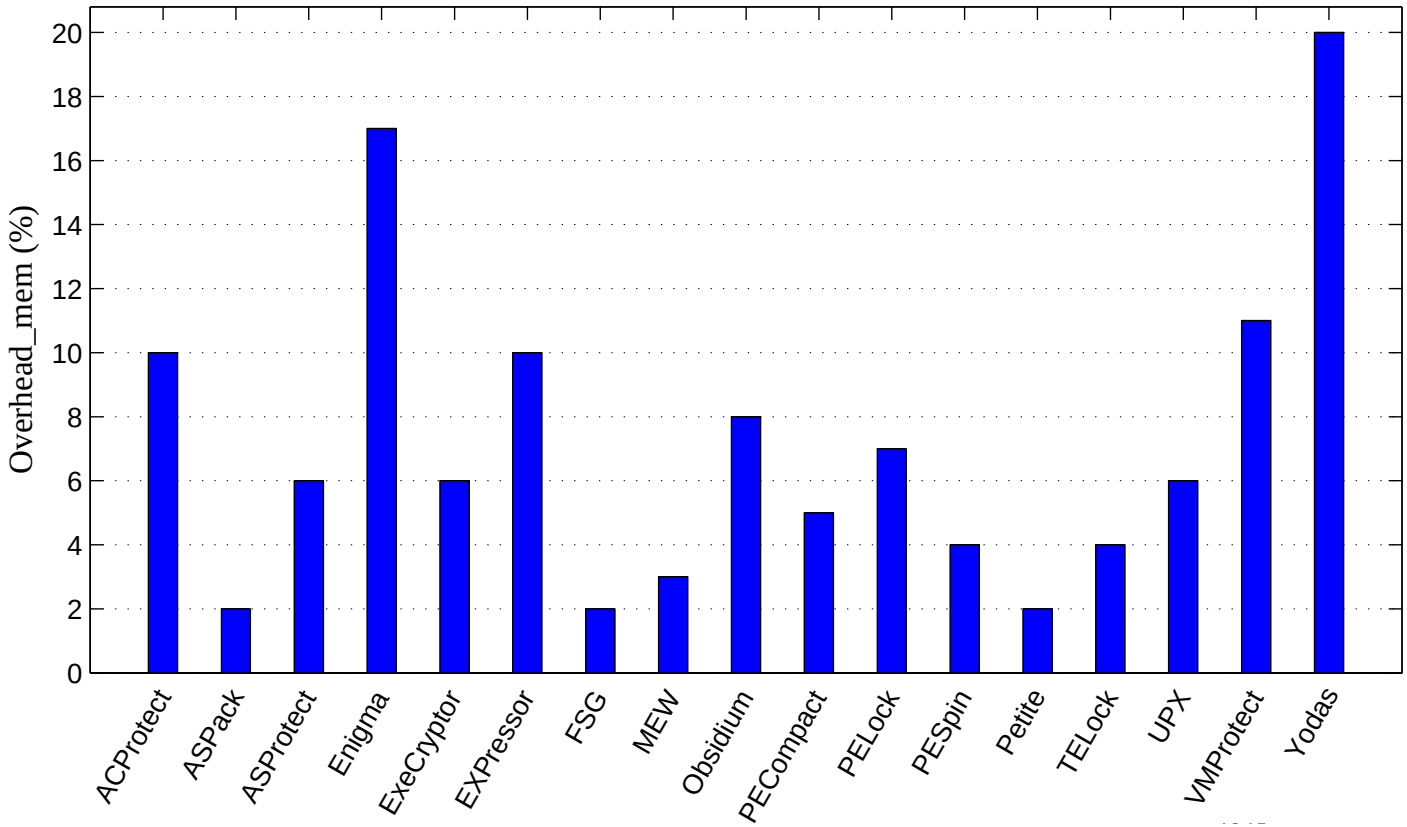
Evaluación Cuantitativa (VI): Rendimiento

Resultados Overhead Mem

$< 6\%$ **ASPack, FSG, MEW, PECompact, PESpin, Petite, TELock**

16% Enigma (virtualización)

20% Yodas (no Overhead CPU)



Contenidos

Universidad Zaragoza 1542

Conclusiones y trabajo futuro (I): Conclusiones

✓ Se han analizado 252 ejecutables ⇒ 21 herramientas x 12 aplicaciones

- Protecciones
- Fiabilidad
- Rendimiento

Universidad Zaragoza 1542

Conclusiones y trabajo futuro (I): Conclusiones

✓ Se han analizado 252 ejecutables ⇒ 21 herramientas x 12 aplicaciones

- Protecciones
- Fiabilidad
- Rendimiento

Análisis Protectores de Software: Conclusiones

✓ Protectores de software dificultan ataques de ingeniería inversa (no los evitan). Desventajas:

Conclusiones y trabajo futuro (I): Conclusiones

✓ Se han analizado 252 ejecutables ⇒ 21 herramientas x 12 aplicaciones

- Protecciones
- Fiabilidad
- Rendimiento

Análisis Protectores de Software: Conclusiones

✓ Protectores de software dificultan ataques de ingeniería inversa (no los evitan). Desventajas:

⇓ **Efectividad disminuye con el paso del tiempo** → técnicas de desprotección conocidas y difundidas por la red

Conclusiones y trabajo futuro (I): Conclusiones

✓ Se han analizado 252 ejecutables ⇒ 21 herramientas x 12 aplicaciones

- Protecciones
- Fiabilidad
- Rendimiento

Análisis Protectores de Software: Conclusiones

✓ Protectores de software dificultan ataques de ingeniería inversa (no los evitan). Desventajas:

⇓ **Efectividad disminuye con el paso del tiempo** → técnicas de desprotección conocidas y difundidas por la red

⇓ Fiabilidad: **algunas herramientas no funcionan** bien con determinados ejecutables

Conclusiones y trabajo futuro (I): Conclusiones

✓ Se han analizado 252 ejecutables ⇒ 21 herramientas x 12 aplicaciones

- Protecciones
- Fiabilidad
- Rendimiento

Análisis Protectores de Software: Conclusiones

✓ Protectores de software dificultan ataques de ingeniería inversa (no los evitan). Desventajas:

⇓ **Efectividad disminuye con el paso del tiempo** → técnicas de desprotección conocidas y difundidas por la red

⇓ Fiabilidad: **algunas herramientas no funcionan** bien con determinados ejecutables

⇓ Rendimiento del programa protegido → ↑ **consumo de memoria y, en algunos casos,** ↑ **tiempo de ejecución**

Conclusiones y trabajo futuro (I): Conclusiones

✓ Se han analizado 252 ejecutables ⇒ 21 herramientas x 12 aplicaciones

- Protecciones
- Fiabilidad
- Rendimiento

Análisis Protectores de Software: Conclusiones

✓ Protectores de software dificultan ataques de ingeniería inversa (no los evitan). Desventajas:

⇓ **Efectividad disminuye con el paso del tiempo** → técnicas de desprotección conocidas y difundidas por la red

⇓ Fiabilidad: **algunas herramientas no funcionan** bien con determinados ejecutables

⇓ Rendimiento del programa protegido → ↑ **consumo de memoria y, en algunos casos,** ↑ **tiempo de ejecución**

X Programas legítimos pueden ser detectados por anti-virus (**falsos positivos**)

Conclusiones y trabajo futuro (II): Líneas futuras

Líneas futuras

- **Ampliación herramientas protectoras** de software
- **Ampliación aplicaciones para benchmark**
- Utilización de **versiones completas** de las herramientas protectoras (Armadillo, EXECryptor, Themida, PELock, VMProtect)

Universidad Zaragoza
1542

Tiempo dedicado: Desglose Horas

Tareas	Horas
Reuniones	10
Estudio de Formato PE	32
Estudio de las técnicas de protección de software	45
Análisis de las herramientas de protección (calidad, fiabilidad y rendimiento)	98
Creación del benchmark	117
Documentación memoria	159
Total	**461**

Referencias

- **[RHD+06]** Paul Royal, Mitch Halpin, David Dagon, Robert Edmonds & Wenke Lee. **PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware**. *Georgia Institute of Technology*, 2006.
- **[KPY07]** Min Gyung Kang, Pongsin Poosankam & Heng Yin. **Renovo: A Hidden Code Extractor for Packed Executables**. Carnegie Mellon University, 2007.
- **[MCJ07]** Lorenzo Martignoni, Mihai Christodorescu & Somesh Jha. **OmniUnpack: Fast, Generic, and Safe Unpacking of Malware**. In Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC'07). IEEE Computer Society, 2007.
- **[GFC08]** Fanglu Guo, Peter Ferrie & Tzi-Cker Chiueh. **A Study of the Packer Problem and Its Solutions**. Procs. 11th Int. Symp. on RAID, 2008.
- **[JCL+10]** Guhyeon Jeong, Euijin Choo, Joosuk Lee, Munkhbayar Bat-Erdene & Heejo Lee. **Generic Unpacking using Entropy Analysis**. IEEE: 5th International Conference on Malicious and Unwanted Software, 2010.
- **[RG14]** Ricardo J. Rodríguez & Iñaki Rodríguez Gastón. **Hardering Sandbox Environments with Dynamic Binary Instrumentation Techniques**. Submitted IEEE TIFS 2014.
- **[KLC+10]** Min-Jae Kim, Jin-Young Lee, Hye-Young Chang, SeongJe Cho, Minkyu Park, Yongsu Park & Philip A. Wilsey. **Design and Performance Evaluation of Binary Code Packing for Protecting Embedded Software against Reverse Engineering**. IEEE Int. Symp. on Object/Component/Service-Oriented Real-Time Distributed Computing, 2010.

Universidad Zaragoza
1542