Análisis exhaustivo de malware en forense de memoria

Ricardo J. Rodríguez

Universidad de Zaragoza

Summer Boot Camp 2021

Profesorado

- **Ricardo J. Rodríguez**
  - Profesor Contratado Doctor en la Universidad de Zaragoza
  - **Investigador en temas de ciberseguridad**, especialmente en:
    - Seguridad de aplicaciones binarias
    - Análisis forense de memoria
    - Seguridad en sistemas basados en RFID/NFC
  - **Grupo de investigación DisCo**
    - RME-DisCo: https://reversea.me
    - Síguenos en Twitter y en Telegram! @reverseame
  - **Correo electrónico**: rjrodriguez@unizar.es
  - **Página web personal**: http://www.ricardojrodriguez.es

Departamento de Informática e Ingeniería de Sistemas

Universidad Zaragoza

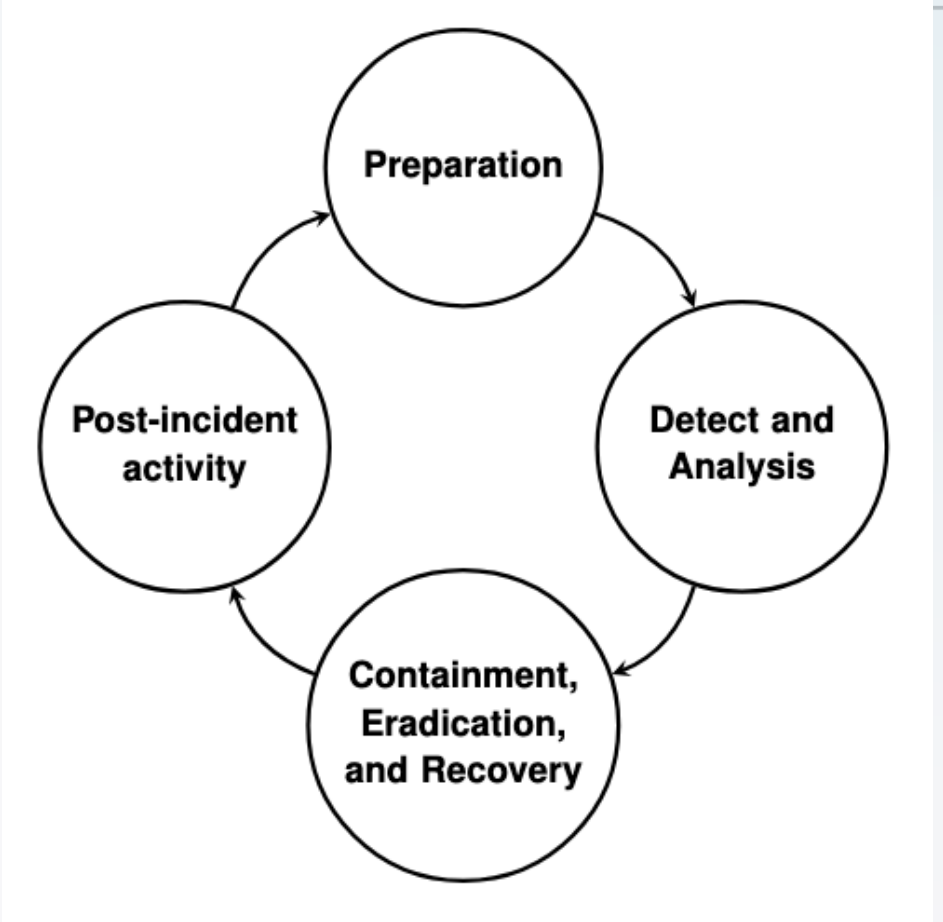
Tabla de contenidos - Índice

Tabla de contenidos - Índice

1. Introducción

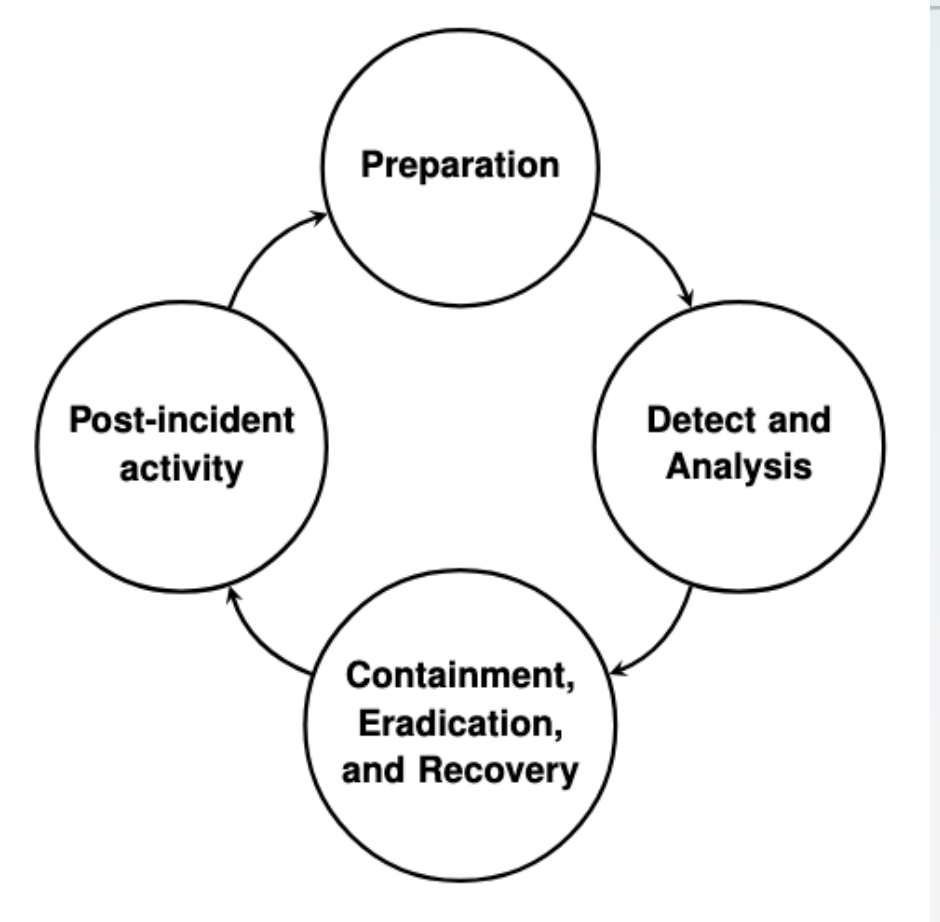
1. Introducción
Respuesta a incidentes

- **Fases de la respuesta a incidentes (NIST SP 800-61)**
  1. **Preparación**
     - Preparación para la gestión de incidentes
     - Prevención de incidentes
  2. **Detección y análisis**
     - Vectores de ataque
     - Señales de un incidente
     - Fuentes de precursores e indicadores
     - Análisis del incidente, documentación, priorización y notificación
  3. **Contención, erradicación y recuperación**
  4. **Actividad post-incidente**



1. Introducción
Respuesta a incidentes

- **Fases de la respuesta a incidentes (NIST SP 800-61)**
  1. **Preparación**
  2. **Detección y análisis**
  3. **Contención, erradicación y recuperación**
     - Estrategias de contención
     - Recogida y gestión de evidencias
     - Identificación de atacantes
     - Erradicación y recuperación
  4. **Actividad post-incidente**
     - Lecciones aprendidas
     - Uso de información recogida del incidente
     - Retención de evidencias



1. Introducción
Respuesta a incidentes

- **Conocer qué ha ocurrido**, además de **preservar** toda la información relativa al incidente
- Dar respuesta a las conocidas 6 W's: *what, who, why, how, when, y where*
- **Incidente habitual**: presencia de software dañino (*malware*)
- **Diversas vertientes de análisis forense**:
  - ***Análisis forense de dispositivos***
    - Soportes digitales
    - ***Memoria***
  - Análisis forense de comunicaciones

1. Introducción
Respuesta a incidentes

- **Análisis forense de soportes digitales versus memoria**
  - Dificultad de acceso a los soportes digitales
  - Información cifrada
  - Información volátil
  - Cantidad de información excesiva

1. Introducción
Análisis forense de memoria

- **¿Puedo usar el análisis forense de memoria para detección de malware?**
  - **Sí. Y no.**
  - **Problemas relativos al contenido disponible en memoria**
    - Paginación
    - Carga bajo demanda
    - Contaminación (*smearing*)
  - Lo mejor sería usar el análisis forense de soportes digitales como **complemento**
    - Es decir, que no sea únicamente en lo que nos basamos

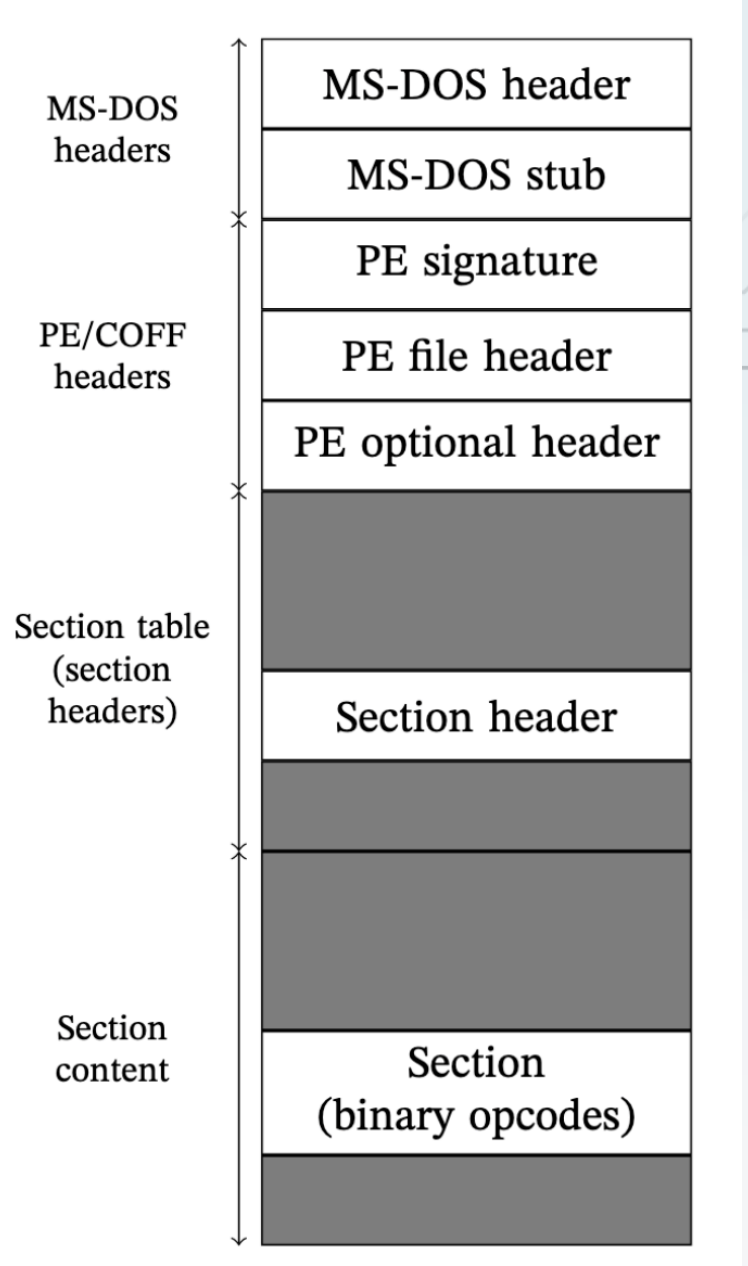
1. Introducción
Malware

- ***Malicious software***
  - **Software especialmente diseñado para realizar algún tipo de daño a un sistema informático**
  - **Diferentes tipos, según su funcionalidad**: keylogger, banker, ransomware, botnet, etc…
    - Puede tener varias funcionalidades a la vez
  - **Ciclo de vida:**
    1. Compromiso del sistema (ataques de ingeniería social, waterhole, insiders, etc.)
    2. Persistencia
    3. Comunicación con servidor de mando y control
    4. Movimiento lateral
    5. Exfiltración de datos / actividad dañina

2. Conceptos previos

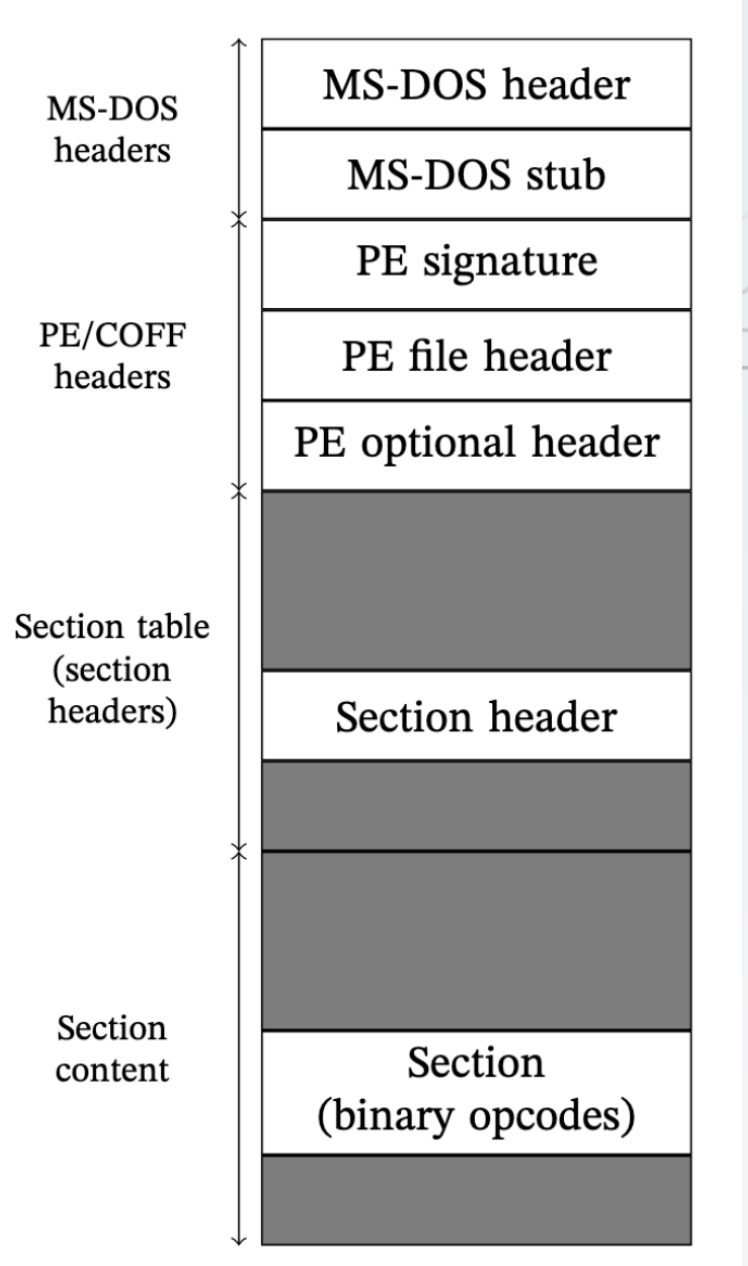
2. Conceptos previos
Estructura de ejecutables

- Desde Windows NT 3.1
- **PE: Portable Executable**
  - Estructura de datos definida en WinNT.h (Microsoft Windows SDK)
  - Tres partes: cabeceras MS-DOS, cabeceras PE/COFF, cabeceras de secciones
- **Cabeceras de MS-DOS**
  - Primeros 64 bytes
  - e_magic: MZ (Mark Zbikowski)
  - e_lfanew: offset a cabeceras PE/COFF



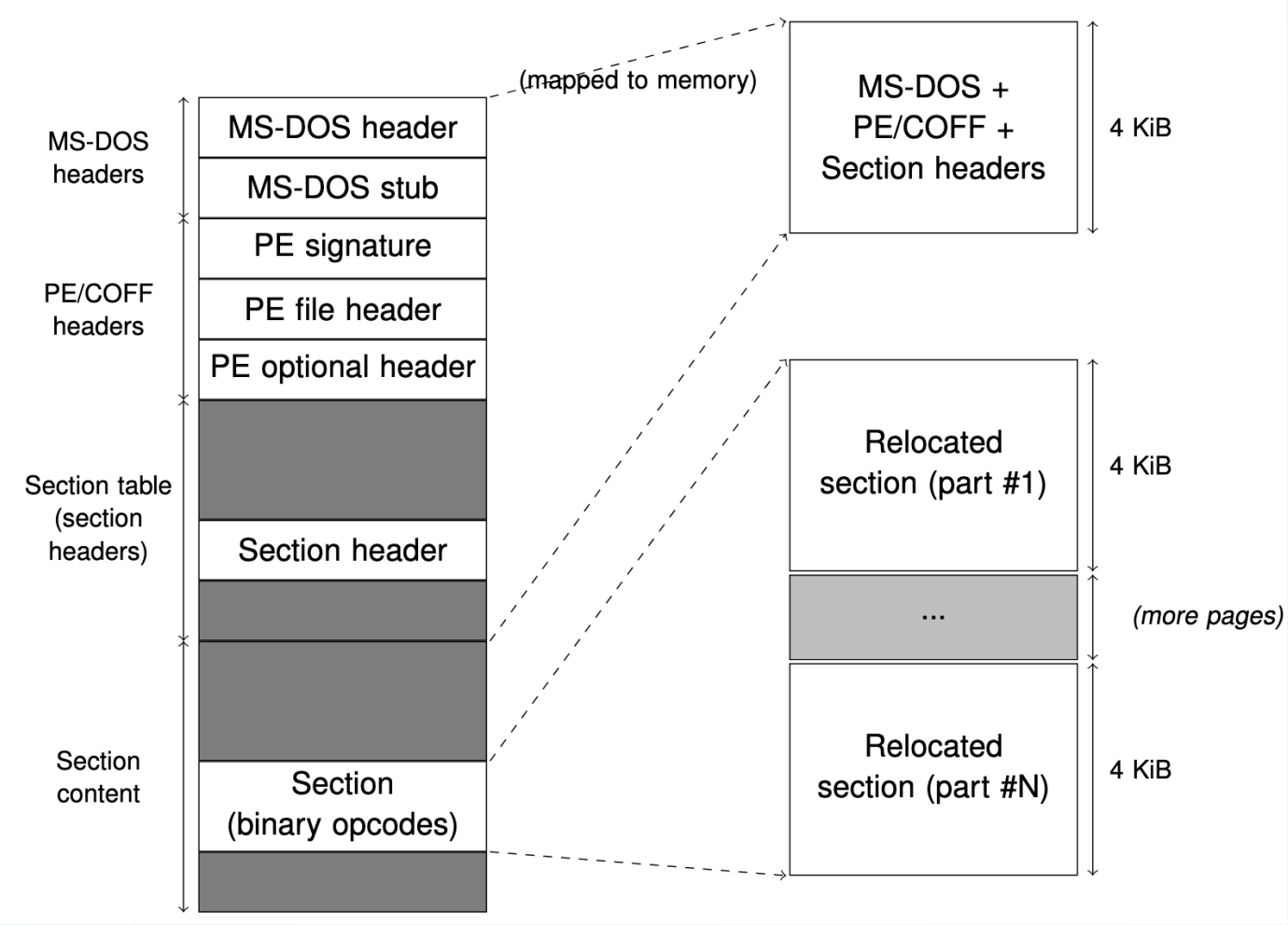
2. Conceptos previos
Estructura de ejecutables

- **Cabeceras PE/COFF**
  - **Signatura PE** (“PE\0\0”)
  - **Cabecera archivo PE**
    - Define máquina destino, número de secciones, características, etc.
  - **Cabecera PE opcional**
    - Opcional para algunos archivos objeto
    - Campos de interés: ImageBase, BaseOfCode, AddressOfEntryPoint
    - DataDirectory: tabla de directorios. Cada entrada tiene un significado
- **Cabeceras de sección**
  - Estructura IMAGE_SECTION_HEADER
  - Secciones comunes: .text/.code, .rdata/.rodata, .data, .reloc, …



2. Conceptos previos

Carga de ejecutables en memoria

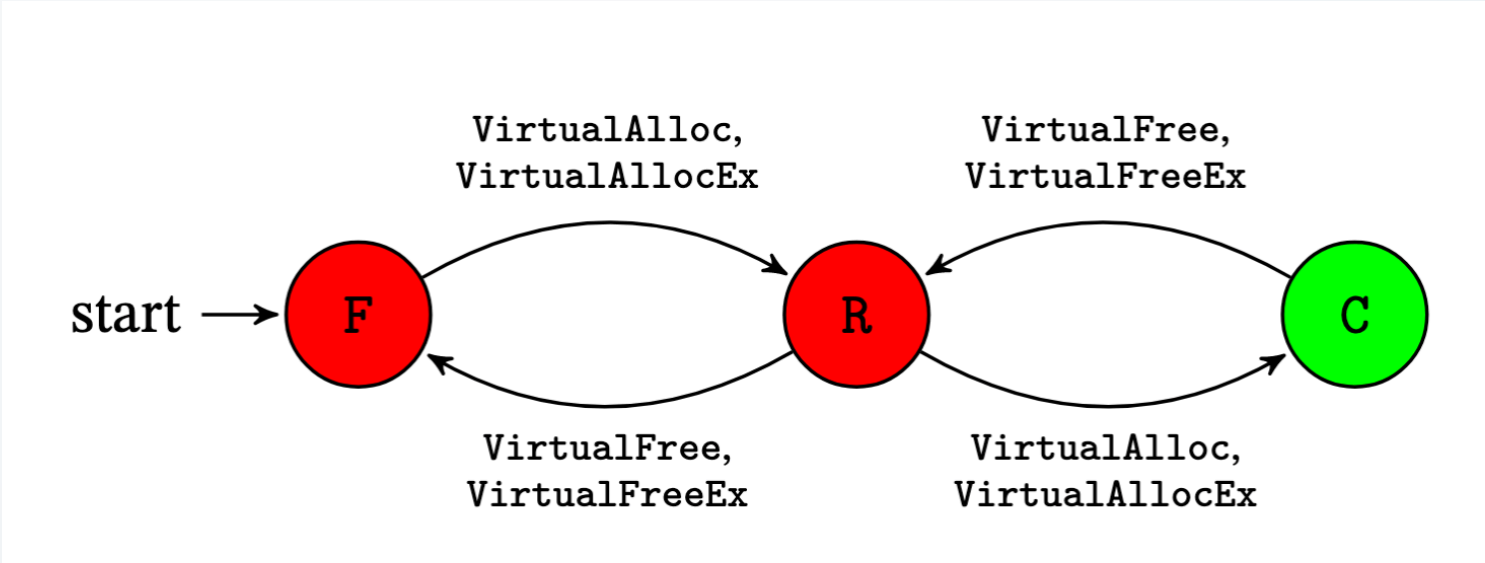


2. Conceptos previos
La memoria virtual

- **Dirección física** vs. **Dirección virtual**
  - Traducción realizada por la unidad de gestión de memoria (MMU)
  - PTE: *page table entries*
    - Cada proceso y el propio núcleo tienen sus propias tablas de páginas
    - Mapea dirección virtual a dirección física
- **Espacio de memoria virtual de un proceso**
  - **Regiones contiguas**
  - Diferentes usos: mapeado de ficheros (respaldo con fichero de disco), memoria no mapeada
- *Virtual Address Descriptor* (VAD)
  - Estructura del núcleo para representar una región contigua de memoria (puede contener varias páginas)
  - Árbol balanceado
  - Diferentes permisos (*comentaremos más tarde*…)

2. Conceptos previos
La memoria virtual: páginas

- **Página: <u>mínima granularidad de la memoria</u>**
  - Bloque de tamaño fijo y contiguo de memoria virtual
  - Pequeñas (4KiB) y grandes (por ejemplo, 2MiB en x86 y x64, 4MiB en ARM)
- **Estados:**
  - Libre: estado inicial
  - Reservada: para uso futuro
  - Entregada (*committed*)

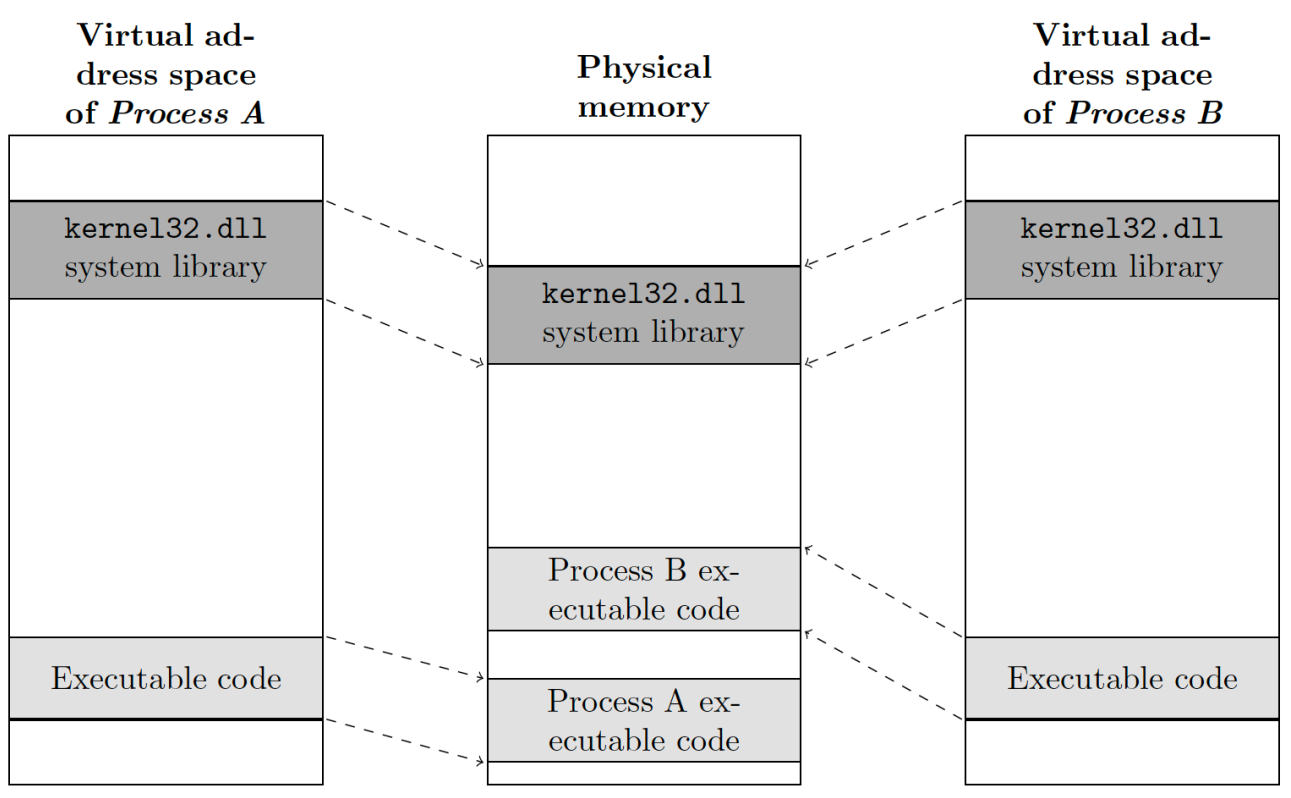


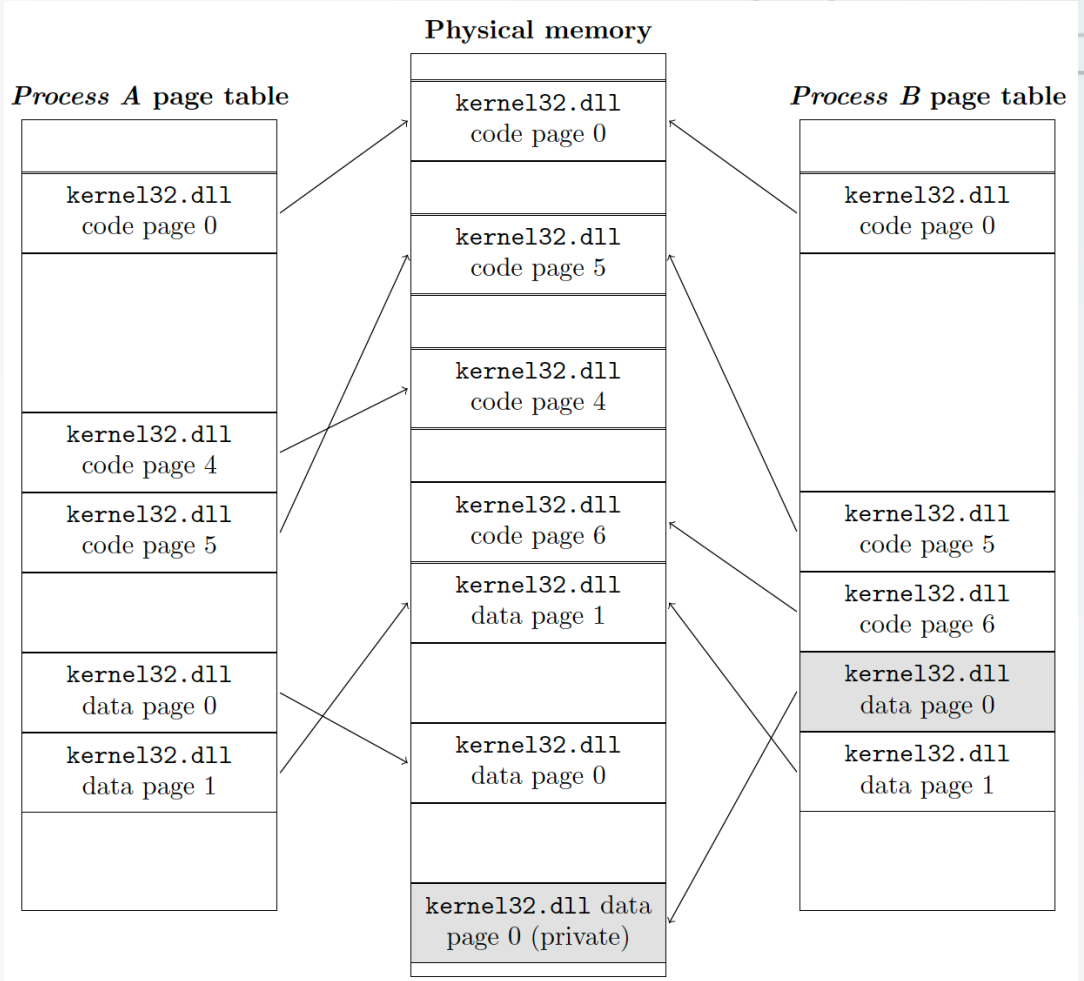
2. Conceptos previos
La memoria virtual: problemas

1. **Paginación**
   - Espacio de memoria disponible para **un** proceso en 32 bits: 2GiB
   - *¿Es físicamente posible?*
   - MMU gestiona las páginas de memoria que se acceden y están paginadas, recuperándolas de disco y colocándola de nuevo en memoria
2. **Carga bajo demanda**
   - Sólo se cargan las páginas de memoria que se necesiten, y cuando se necesiten (*lazy loading*)
   - Mecanismo Copy-on-Write (CoW)
3. **Contaminación** (*smearing*)
   - La memoria es un ente vivo, cambiando continúamente
   - Problema en captura de memoria en sistemas que están funcionando
     - Posibles referencias entre zonas de memoria muy alejadas

2. Conceptos previos

La memoria virtual: páginas y procesos





3. Análisis de malware en forense de memoria

3. Análisis de malware en forense de memoria

Fases de análisis de malware

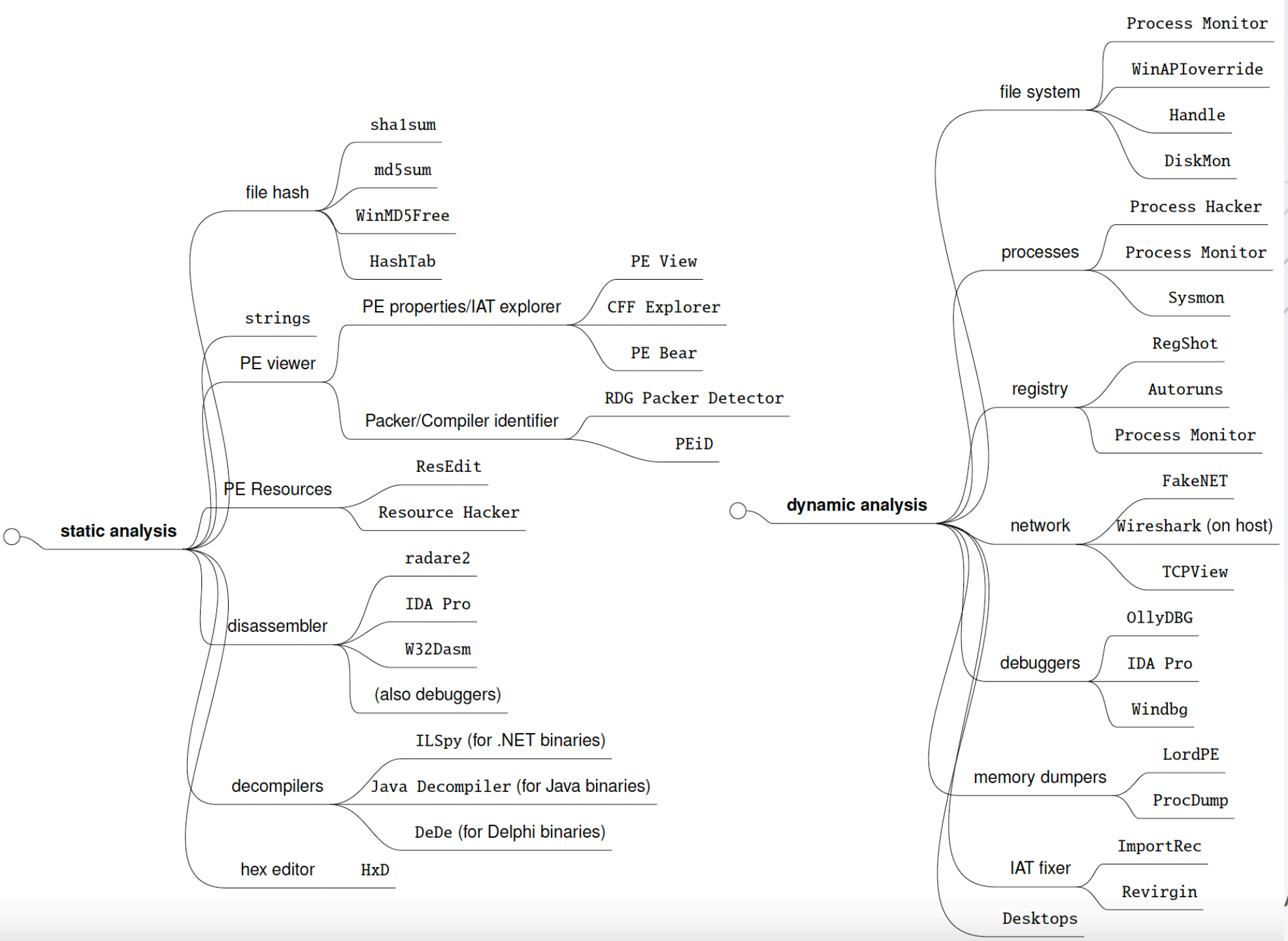
- **Análisis estático** (no se ejecuta)
  - Signaturas (MD5, SHA-1, SHA-256…)
    - HashTab, md5sum, sha1sum, WinMD5Free, …
  - Cadenas
    - strings
  - Propiedades del PE
    - Campos de interés (¿ofuscado? ¿empacado?)
    - Funciones declaradas en la *Import Address Table* (IAT)
    - Recursos del ejecutable

3. Análisis de malware en forense de memoria

Fases de análisis de malware

- **Análisis dinámico** (se ejecuta – normalmente en máquina virtual o aislada)
  - Interacción con el SO: ficheros
    - ¿Creación? ¿Acceso? ¿Modificación? ¿Eliminación?
  - Interacción con el SO: Registro de Windows
    - ¿Creación? ¿Acceso? ¿Modificación? ¿Eliminación?
  - Interacción con el SO: procesos
    - ¿Creación? ¿Acceso?
  - Interacción con el exterior: comunicaciones de red
    - Dirección IP
    - Nombres de dominio

3.
Fas


static analysis
file hash
sha1sum
md5sum
WinMD5Free
HashTab
strings
PE viewer
PE properties/IAT explorer
PE View
CFF Explorer
PE Bear
Packer/Compiler identifier
RDG Packer Detector
PEiD
PE Resources
ResEdit
Resource Hacker
disassembler
radare2
IDA Pro
W32Dasm
(also debuggers)
decompilers
ILSpy (for .NET binaries)
Java Decompiler (for Java binaries)
DeDe (for Delphi binaries)
hex editor
HxD
dynamic analysis
file system
Process Monitor
WinAPIoverride
Handle
DiskMon
processes
Process Hacker
Process Monitor
Sysmon
registry
RegShot
Autoruns
Process Monitor
network
FakeNET
Wireshark (on host)
TCPView
debuggers
OllyDBG
IDA Pro
Windbg
memory dumpers
LordPE
ProcDump
IAT fixer
ImportRec
Revirgin
Desktops

3. Análisis de malware en forense de memoria

Fases de análisis de malware en forense de memoria

- **Volcado de memoria**
  - Contiene **artefactos** de elementos que se estaban ejecutando en el momento de adquisición
    - **Procesos en ejecución, usuarios conectados, sockets abiertos, etc.**

Proceso: representación en memoria de un fichero ejecutable

1. **Fichero ejecutable mapeado a memoria**
   - Alineación a páginas → *firmas hash no concluyentes*
2. **Carga bajo demanda**
   - **Contenido parcial**: *problema para conocer actividad real realizada por la muestra*
   - La forma de adquirir la memoria puede afectar
3. **Tabla de funciones IAT resuelta**
   - *Dificultad de ejecución posterior en el mismo u otros entornos*

4. Recolección de evidencias de memoria

4. Recolección de evidencias de memoria

Adquisición de memoria

- Diversas técnicas de adquisición
  - Tobias Latzo, Ralph Palutke, Felix Freiling, “A universal taxonomy and survey of forensic memory acquisition techniques,” Digital Investigation, Volume 28, 2019, pp. 56-69, ISSN 1742-2876, https://doi.org/10.1016/j.diin.2019.01.001
- **Herramientas software para volcado completo de memoria**
  - WinPmem: https://github.com/Velocidex/WinPmem
    - Licencia Apache
    - Soporte para Windows XP hasta Windows 10, para 32 y 64 bits
    - Ejemplo: winpmem_mini_x64.exe physmem.raw
  - Linux Memory Extractor (LiME): https://github.com/504ensicsLabs/LiME
    - Licencia GNU/GPLv2
    - Soporte para Linux y Android
    - Extracción a través de conexión a puerto local
  - FTK Imager: https://accessdata.com/product-download/ftk-imager-version-4-2-1
    - Herramienta de pago
    - Soporte para Windows

4. Recolección de evidencias de memoria

Adquisición de memoria

- **Adquisición en máquinas virtuales**
  - VirtualBox
    - vboxmanage debugvm "Win7" dumpvmcore --filename test.elf
  - VMWare
    1. Crear una captura de la ejecución de la máquina virtual (se generan ficheros .vmss y .vmem)
    2. Herramienta vmss2core: https://flings.vmware.com/vmss2core??src=vmw_so_vex_mraff_549
- **Otras herramientas para extracción de procesos o módulos**
  - ProcDump: https://docs.microsoft.com/en-us/sysinternals/downloads/procdump
    - procdump -ma 4572
    - Volcado único (fichero .dmp)
  - Windows Memory Extractor: https://github.com/pedrofdez26/windows-memory-extractor
    - Licencia GNU/GPLv3
    - WindowsMemoryExtractor_x64.exe --pid 1234
    - Crea volcado por secciones de memoria del proceso

4. Recolección de evidencias de memoria

Análisis de volcados de memoria: Volatility

- **Estándar de facto** para analizar volcados de memoria
- Licencia de código abierto GNU/GPLv2
- Publicado en 2007 en BH USA, llamada *Volatools*
- Soporte para Windows, Linux y macOS, en 32 y 64 buts
- API muy rica para implementaciones propias
- Versión 2.6 vs. Versión 3
  - Python2 vs Python3
  - Versión 3 ya es estable! https://github.com/volatilityfoundation/volatility3

4. Recolección de evidencias de memoria

Primeros pasos con Volatility

- Máquina virtual proporcionada: Debian 10.10
  - Volatility 2.6 y Volatility 3.0 ya instalados
  - Usuario/contraseña: alumno / alumno
- Ayuda:
  - python vol.py –h
- Volcado de memoria a analizar:
  - python vol.py --f mem.dmp --profile Win7SP1x86
  - El perfil es sólo necesario en la versión 2.6. Indica dónde están las estructuras internas del SO
- ¿Cómo conocer el profile a usar? → plugin imageinfo
  - python vol.py --f mem.dmp imageinfo
- Los plugins siempre se indican al final del comando

4. Recolección de evidencias de memoria

Detección de indicadores de compromiso con Volatility

https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

- **Procesos y DLLs**
  - pslist, pstree (psscan para posibles rootkits)
  - dlllist, dlldump
  - handles
  - enumfuncs (lista de funciones importadas y exportadas, por proceso/DLL)
- **Memoria de procesos**
  - Memmap, memdump
  - Procdump
  - Vadinfo, vadwalk, vadtree, vaddump
  - evtlogs
  - iehistory
- **Redes**
  - connections, connscan
  - sockets, sockscan
  - netscan (artefactos de redes en Win7)

4. Recolección de evidencias de memoria

Detección de indicadores de compromiso con Volatility

https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

- **Memoria del núcleo y otros objetos**
  - modules, modscan, moddump
  - driverscan
  - filescan
- **Registro**
  - hivescan, hivelist, hivedump
  - printkey
  - lsadump
  - userassist, shellbags, shimcache
  - Dumpregistry
- **Sistema de ficheros**
  - mbrparser, mftparser
- **Análisis de ficheros de hibernación u otros volcados**

4. Recolección de evidencias de memoria

Detección de indicadores de compromiso con Volatility: ejemplo

LABORATORIO 1

- **Volcado zeus.vmem** (del libro Malware Analyst's Cookbook)
- Sigue el guión de laboratorio proporcionado en la página web del taller: http://webdiis.unizar.es/~ricardo/sbc-2021/laboratorios/lab1_introduccion.pdf
  - Detalla de muchos plugins de Volatility de interés para el análisis de volcados de memoria

5. Detección avanzada de IoCs

Plugins no oficiales

- **Existen multitud de plugins adicionales que amplían la funcionalidad de Volatility**
- **Modo de uso**
  1. Instalación del plugin (p.e., descarga de repositorio)
  2. Ejecución: volatility --plugins="/path/to/plugin" -f file [OPTIONS] pluginname

5. Detección avanzada de IoCs

Plugins no oficiales

- **MalConfScan**: https://github.com/JPCERTCC/MalConfScan
  - Extrae configuración, cadenas descifradas o dominios DGA de algunas familias de malware
- **Malscan**: https://github.com/reverseame/malscan (para Volatility 2.6)
  - Licencia GNU/GPLv3
  - Integra malfind con clamav-daemon (sólo disponible en Linux). Menos falsos negativos
  - Modos de funcionamiento: normal (regiones +WX, cualquier módulo ejecutable, y memoria privada de tipo VadS) y full-scan (regiones con +X)
  - VADs sin ejecutables asociados, inicios de función y páginas vacías seguidas de código

```
→  ~ python2 volatility/vol.py --plugins=/home/alumno/malscan -f /mnt/volcados/alina1G.elf
--profile=Win7SP1x86 malscan
Volatility Foundation Volatility Framework 2.6.1

Process: ALINA_CJLXYJ.e Pid: 1828 Space Address: 0xc20000-0xc44fff
Vad Tag: Vad  Protection: PAGE_EXECUTE_WRITECOPY
Flags: CommitCharge: 5, Protection: 7, VadType: 2
Scan result: Win.Trojan.Alina-4

0x00c20000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00c20010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00c20020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00c20030  00 00 00 00 00 00 00 00 00 00 00 00 f0 00 00 00   ................
```

5. Detección avanzada de IoCs
Plugins no oficiales

- **Similarity Unrelocated Module**: https://github.com/reverseame/similarity-unrelocated-module (para Volatility 2.6)
  - Licencia GNU/GPLv3
  - Calcula firmas aproximadas sobre los módulos de un volcado:
    - Algoritmos: dcfldd, ssdeep, sdhash, TLSH
    - Un módulo es un fichero ejecutable o biblioteca de funciones cargada en memoria
  - Permite comparación entre módulos de diferentes volcados
  - Deshace los cambios hechos por el sistema operativo (relocation). Dos métodos:
    - Guided De-relocation
    - Linear Sweep De-relocation
  - Más información: M. Martín-Pérez, R. J. Rodríguez,D. Balzarotti, "*Pre-processing Memory Dumps to Improve Similarity Score of Windows Modules*", Computers & Security, vol. 101, p. 102119, 2021, https://doi.org/10.1016/j.cose.2020.102119

5. Detección avanzada de IoCs

Plugins no oficiales

- **Winesap**: https://github.com/reverseame/winesap (para Volatility 2.6)
  - Licencia AGPLv3
  - Busca todos los puntos de auto-inicio de Windows en el volcado de memoria
  - Claves de Registro binarias o desconocidas: se analizan como PE
  - Cadenas relativas a rutas de ficheros habituales de malware (%APPDATA%, %TMP%, %TEMP%, AppData), NTFS ADS, comandos shells (e.g., rundll32.exe shell32.dll,ShellExecute_RunDLL)

Windows Auto-Start Extensibility Points	*Characteristics*					
	Write permissions	Execution privileges	Tracked down in memory forensics†	Freshness of system	Execution scope	Configuration scope
System persistence mechanisms						
Run keys (HKLM root key)	yes	user	yes	user session	application	system
Run keys (HKCU root key)	no	user	yes	user session	application	user
Startup folder (%ALLUSERSPROFILE%)	yes	user	no	user session	application	system
Startup folder (%APPDATA%)	no	user	no	user session	application	user
Scheduled tasks	yes	any	no	not needed‡	application	system
Services	yes	system	yes	not needed‡	application	system
Program loader abuse						
Image File Execution Options	yes	user	yes	not needed	application	system
Extension hijacking (HKLM root key)	yes	user	yes	not needed	application	system
Extension hijacking (HKCU root key)	no	user	yes	not needed	application	user
Shortcut manipulation	no	user	no	not needed	application	user
COM hijacking (HKLM root key)	yes	any	yes	not needed	system	system
COM hijacking (HKCU root key)	no	user	yes	not needed	system	user
Shim databases	yes	any	yes	not needed	application	system
Application abuse						
Trojanized system binaries	yes	any	no	not needed	system	system
Office add-ins	yes	user	yes	not needed	application	user
Browser helper objects	yes	user	yes	not needed	application	system
System behavior abuse						
Winlogon	yes	user	yes	user session	application	system
DLL hijacking	yes	any	no	not needed	system	system
AppInit DLLs	yes	any	yes	not needed	system	system
Active setup (HKML root key)	yes	user	yes	user session	application	system
Active setup (HKCU root key)	no	user	yes	user session	application	application

†*If the memory is paging to disk, it would be not possible to track down these ASEPs in memory forensics.*
‡*Depends on the trigger conditions defined to launch the program.*

- Más información: D. Uroz, R. J. Rodríguez, “Characteristics and Detectability of Windows Auto-Start Extensibility Points in Memory Forensics”, Digital Investigation, vol. 28, p. S95-S104, 2019, https://doi.org/10.1016/j.diin.2019.01.026

5. Detección avanzada de IoCs
Plugins no oficiales

- **Sigcheck**: https://github.com/reverseame/sigcheck (para Volatility 2.6)
  - Licencia GNU/GPLv3
  - Verifica ficheros PE firmados digitalmente con Microsoft Authenticode
  - Dos métodos de firma: embebida (en el PE), por catálogo (en fichero externo)
  - **IMPORTANTE**: verifica que el fichero ejecutable que se inició era original
    - Si un malware hace *process hollowing* no se detectaría con este método
  - Más información: D. Uroz, R. J. Rodríguez, "On Challenges in Verifying Trusted Executable Files in Memory Forensics", Forensic Science International: Digital Investigation, vol. 32, p. 300917, 2020, https://doi.org/10.1016/j.fsidi.2020.300917

5. Detección avanzada de IoCs

Desarrollo de herramientas propias

LABORATORIO 2

- **Volcado alina1G.elf**
- Sigue el guión de laboratorio proporcionado en la página web del taller: http://webdiis.unizar.es/~ricardo/sbc-2021/laboratorios/lab2_desarrollo_plugins.pdf

5. Detección avanzada de IoCs

Ejemplo: WannaCry

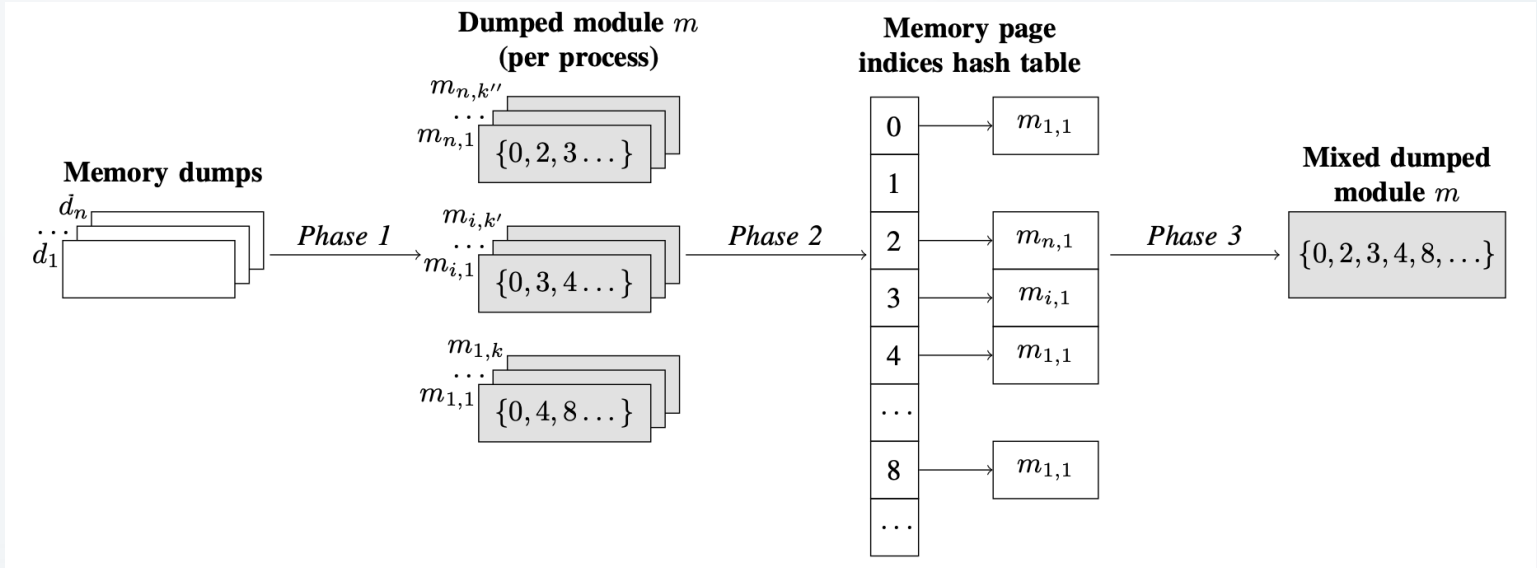
LABORATORIO 3

- **Volcado wannacry.elf**
- Sigue el guión de laboratorio proporcionado en la página web del taller: http://webdiis.unizar.es/~ricardo/sbc-2021/laboratorios/lab3_ejemplo_wannacry.pdf

6. Diseño de workflows de análisis

Montaje del flujo de análisis e intercambio de información

- **Definir claramente qué se desea obtener**
  - Buscar en los plugins de la comunidad de Volatility si ya está hecho (*no hay que reinventar la rueda!*)
- **Desarrollo del pipeline**
  - ¿Python? ¿Bash?
  - Multi-threading
  - Extracción de módulos y análisis
    - Sandbox, VT, pefile
- **Intercambio de información**
  - Formatos estándar:
    - JSON, CSV, etc.
- **Informe final de análisis:**
  - ¿JSON? ¿Markdown?



7. Bibliografía recomendada

- The Art of Memory Forensics
  - Material adicional disponible aquí
- Practical Malware Analysis. The Hands-On Guide to Dissecting Malicious Software
- Malware Analyst's Cookbook
- Documentación de Volatility 3

Análisis exhaustivo de malware en forense de memoria

Ricardo J. Rodríguez

Universidad de Zaragoza

Summer Boot Camp 2021