Malware Analysis for Incident Response

Ricardo J. Rodríguez
University of Zaragoza

Cybersecurity Summer Bootcamp

LEON - 2023

July 3 - 13, 2023
Leon, Spain
#CyberSBC2023

Organized by:

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA PRIMERA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

incibe_
INSTITUTO NACIONAL DE CIBERSEGURIDAD

OAS More rights for more people

With the collaboration of:

Canadä

AYUNTAMIENTO DE LEÓN

LEÓN CON GRE SOS

universidad de león

Agenda

1. Introduction
2. Malware Analysis Methodology
3. Hands-On: Malware Analysis
4. Incident Response Integration
5. Hands-On: Malware Analysis Integrated into Incident Response

Organizers:

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA PRIMERA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

incibe_
SPANISH NATIONAL CYBERSECURITY INSTITUTE

OAS More rights for more people

Partners:

Canada

AYUNTAMIENTO DE LEÓN

LEÓN CON GRE SOS

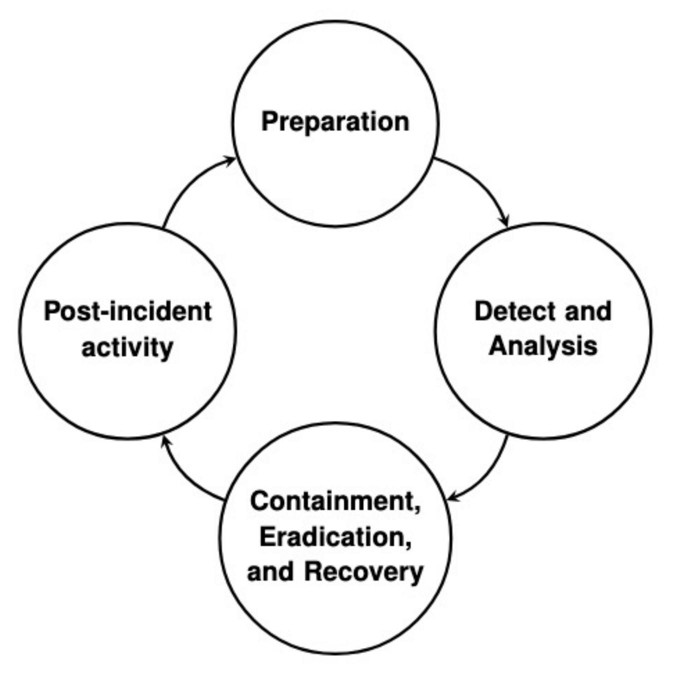
universidad de león

1. Introduction

1. Introduction

Incident Response

- Incident response phases (NIST SP 800-61)
  1. **Preparation**
     - Preparedness for incident management
     - Incident prevention
  2. **Detect and Analysis**
     - Attack vectors
     - Indicators of incidents
     - Sources of precursors and indicators
     - Incident analysis, documentation, prioritization and notification
  3. **Containment, Eradication, and Recovery**
  4. **Post-incident activity**



1. Introduction

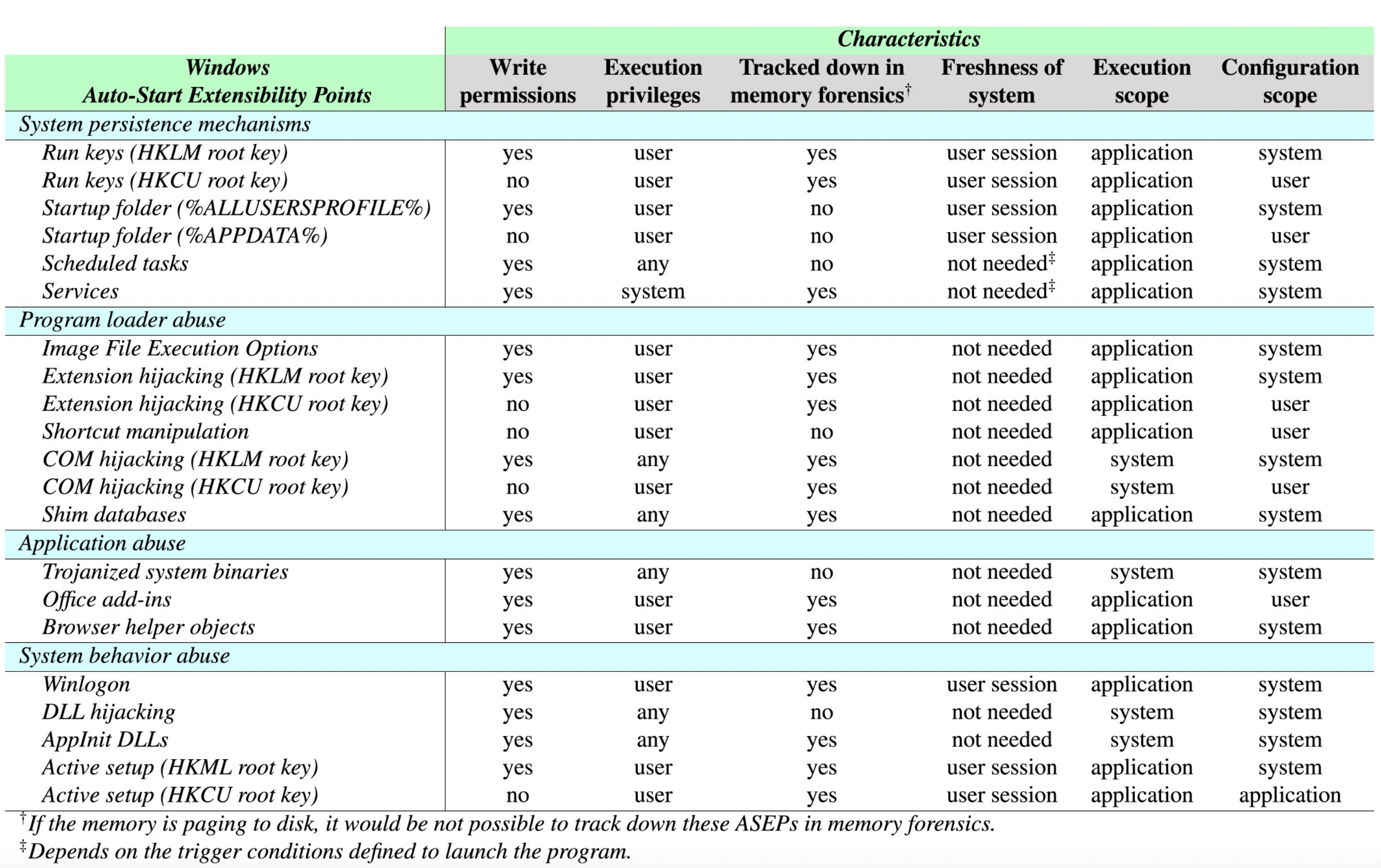
Incident Response

- **Know what has happened, preserving all the information** related to the incident
- Respond to the well-known 6 W's: *what, who, why, how, when*, and *where*
- **Usual incident**: presence of malicious software (*malware*)
- **Various aspects of forensic analysis**:
  - Device forensics
    - Digital drive (digital media)
    - Memory
  - Network forensics

1. Introduction

Malware

- *Malicious software*
  - **Software specially designed to do some k**
  - **Different types, depending on their funct**
    - They can have several functionalities at th
  - **Lifecycle**
    1. Initial compromise (social engineering at
    2. Persistence
    3. Communication with C&C servers
    4. Lateral movement
    5. Data exfiltration / malicious activity



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.*

More details: Uroz, D. & Rodríguez, R. J. **Characteristics and Detectability of Windows Auto-Start Extensibility Points in Memory Forensics**. Digital Investigation, 2019, 28, S95-S104, Elsevier. https://doi.org/10.1016/j.diin.2019.01.026

1. Introduction

Malware and Incident Response

- **Identification and Classification**:
  - We need to understand the specific characteristics of the malware
- **Behavior and Impact Analysis:**
  - How it spreads, communicates, and interacts with the compromised system/network
- **Indicators of Compromise (IOCs):**
  - Valuable clues for detecting and mitigating the presence of the malware across the systems and networks
  - They help identify affected assets, patterns of malicious activity, and potential entry points for future attacks
- **Root Cause Analysis:**
  - How the malware entered the environment (phishing emails, malicious downloads, or other means)
- **Mitigation and Remediation:**
  - Specific actions required to mitigate the impact of the malware and remove it from compromised systems
  - Identify the necessary patches, security updates, or configuration changes needed to prevent further propagation and restore the affected systems to a secure state
- **Threat Intelligence and Information Sharing:**
  - Enhance collective defenses and improve incident response across the industry

1. Introduction

Importance of Malware Analysis in Incident Response

- **Threat Understanding**
  - Understanding helps incident responders assess the severity of the threat, determine its potential impact on affected systems, and make informed decisions
- **Incident Triage and Prioritization**:
  - Malware analysis aids in the initial triage and prioritization of security incidents
  - Categorize incidents based on their severity, potential for damage, and the level of risk they pose to critical assets
  - More efficient allocation of resources and the ability to prioritize the most critical incidents
- **Indicators of Compromise (IOCs):**
  - Malware analysis helps identify and extract indicators of compromise (IOCs) associated with the malware
  - File hashes, network signatures, behavior patterns, and other identifiable artifacts
  - Crucial role in threat hunting, proactive defense, and future incident prevention

1. Introduction

Importance of Malware Analysis in Incident Response

- **Incident Containment and Eradication:**
  - Insights into the techniques and mechanisms used by the malware to propagate and persist within the compromised environment
  - Effective strategies for containing the incident, isolating affected systems or networks, and taking appropriate steps to eradicate the malware
- **Post-Incident Analysis and Learning:**
  - Identify the entry point of the malware, determine the vulnerabilities or security gaps exploited, and gain insights into the attacker's tactics, techniques, and procedures (TTPs)
  - Enhance preventive measures, strengthen defenses, and improve future incident response capabilities.
- **Threat Intelligence and Information Sharing:**
  - Enhances collective defenses, enables early detection of similar threats, and facilitates a more proactive approach to incident response

2. Malware Analysis Methodology

2. Malware Analysis Methodology

- **Static program analysis** (also called *dead code* or *cold analysis*)
  - **The program does not run**
  - You should take a look at…
    - PE properties
    - Import functions (which APIs are used?)
    - Hash computation (e.g., MD5, SHA1)
    - Retrieve strings from the binary file: strings
  - **Disadvantage:**
    - All possible execution paths are explored (*state explosion problem*)
      - You might be analyzing infeasible code

2. Malware Analysis Methodology

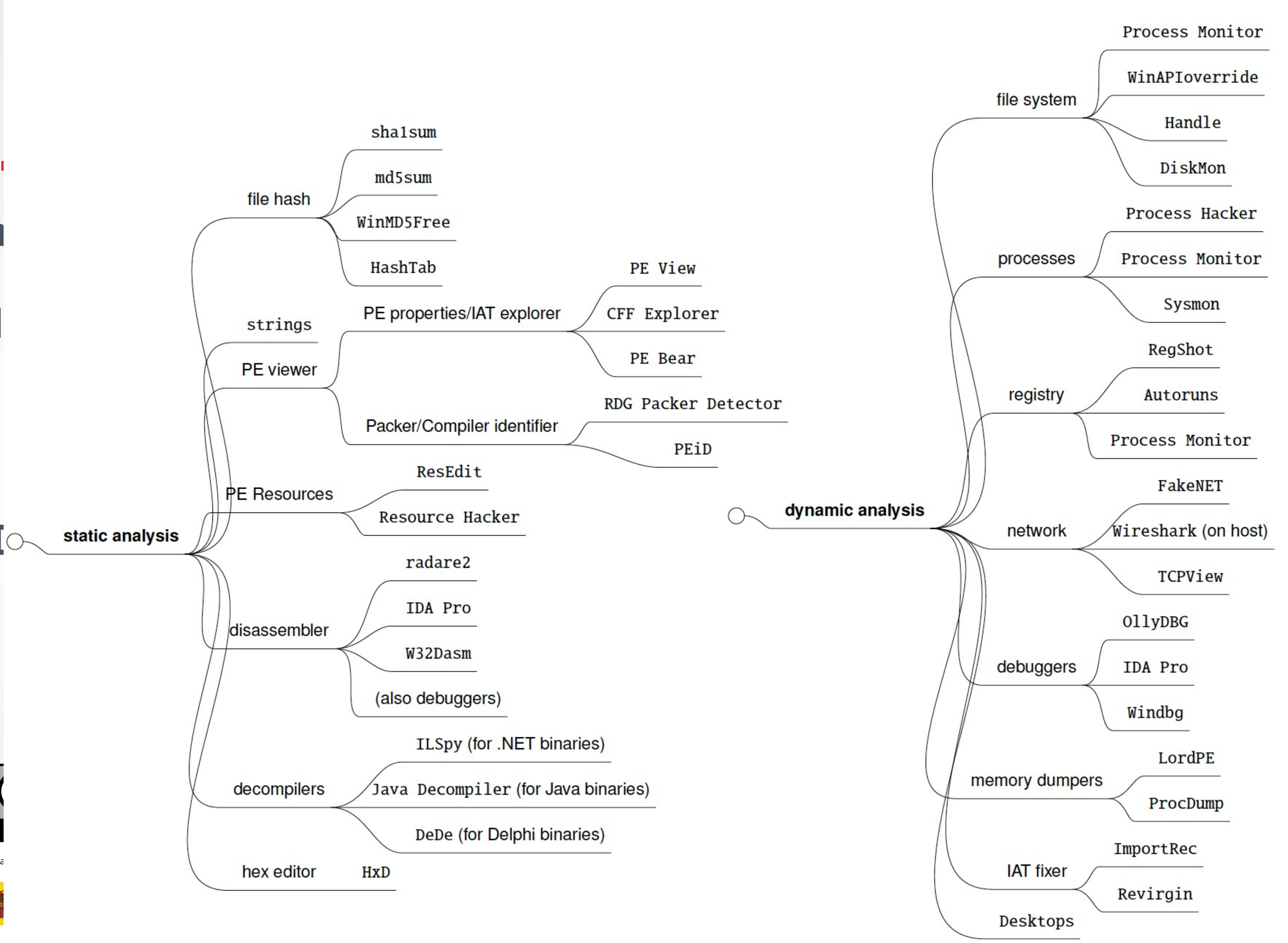
- **Dynamic program analysis** (also called *live code* or *hot analysis*)
  - **The program does run**
  - You should take a look at…
    - Interaction with the OS: at the filesystem, process, and Windows Registry levels
    - Interaction with the Internet: connections to domain names or IPs, network data transmitted
  - Helps find out their (malicious?) behaviour
  - **Disadvantage:**
    - Only one of the possible execution paths is explored
      - It may depend on the current execution conditions (environment variables, datime, etc.)

2.

An

- I

- [



- 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

Orga

dad

2. Malware Analysis Methodology

Static Analysis

- **File structure analysis**
  - Examine the headers, sections, and metadata
- **Binary code examination**
  - Analyze the instructions, functions, and logic to understand its behavior
  - By disassembling or decompiling the code to obtain a human-readable representation for analysis
- **API calls and system functions**
  - Insights into the malware’s capabilities (accessing files, manipulating processes, establishing network communications)
- **String analysis**
  - Information about its functionality, communication protocols, or command structures
  - Indicators of malicious behavior, hardcoded URLs, encryption keys, or C&C server addresses

2. Malware Analysis Methodology

Static Analysis

- **Signature-based detection**
  - MD5/SHA1/SHA256 hashes
  - Approximate matching algorithms (ssdeep, SDHASH, TLSH)
  - Unique patterns from the code (e.g., YARA rules)

- **Limitations:**
  - Limited to known malware samples
  - Inability to detect polymormihc or encrypted malware

2. Malware Analysis Methodology

Use of Windows APIs

- **Static import**
  - Windows APIs invoked by the binary
  - They are present in the DataDirectory section, visible with any PE viewing tool
  - Function identified by string name or ordinal position (in EAT)
- **Dynamic import**
  - Windows API is resolved on execution
  - Different ways to dynamically import a function
    - Usually, LoadLibrary (loads a DLL) + GetProcAddress (gets the address of the function)
    - Can also be dynamically resolved by ordinal position (in EAT) instead of function name

2. Malware Analysis Methodology

Use of Windows APIs

- **Processes and IPCs (kernel32.dll)**
  - CreateProcessA, OpenProcess, CreateThread, CreatePipe, CreateNamedPipe, CreateMutex, OpenMutex, CreateToolhelp32Snapshot, CreateRemoteThread, ...
- **Files (kernel32.dll)**
  - CreateFile, WriteFile, ReadFile, CopyFile, MoveFile, OpenFile ...
- **Registry (advap32i.dll)**
  - RegOpenKey, RegEnumKey, RegEnumValue, RegDeleteKey, RegQueryInfoKey, …
- **Network (ws2_32.dll, wininet.dll, …) – Winsocks and others**
  - WSAStartup, WSASocket, socket, connect, accept, bind, recv, send, htons, …
  - urlmon.dll: URLDownloadToFile, …
  - wininet.dll: HttpOpenRequest, HttpSendRequestA, FtpOpenFileA, …

2. Malware Analysis Methodology

- **Dynamic analysis** (the program runs – typically in an isolated environment)
  - OS interaction: files
    - Creation? Access? Modification? Deletion?
  - OS interaction: Windows Registry
    - Creation? Access? Modification? Deletion?
  - OS interaction: processes
    - Creation? Access?
  - Interaction with the outside: network communications
    - IP addresses
    - Domain names

3. Hands-On: Malware Analysis

Cybersecurity Summer Bootcamp

LEON - 2023

3. Hands-On: Malware Analysis

LAB SESSION 1

- **Additional files for *Lab session 1***
  - https://webdiis.unizar.es/~ricardo/sbc-2023/laboratories/additional_files/lab1_malware_files.7z
- Follow the laboratory workbook provided on the workshop's website: https://webdiis.unizar.es/~ricardo/sbc-2023/laboratories/lab1_intro_malware_analysis.pdf

Organizers:

VICEPRESIDENCIA PRIMERA DEL GOBIERNO MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL
incibe_ SPANISH NATIONAL CYBERSECURITY INSTITUTE

Partners:

OAS More rights for more people
Canada
AYUNTAMIENTO DE LEÓN
LEÓN CONGRESOS
universidad de león

4. Incident Response Integration

Cybersecurity Summer Bootcamp

LEON - 2023

4. Incident Response Integration

Collection of Memory Evidence: Memory Acquisition

- **Various acquisition techniques**
  - 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
- **Software tools for complete memory dump**
  - WinPmem: https://github.com/Velocidex/WinPmem
    - Apache license
    - Support for Windows XP up to Windows 10, for 32 and 64 bits
    - Example: winpmem_mini_x64.exe physmem.raw
  - Linux Memory Extractor (LiME): https://github.com/504ensicsLabs/LiME
    - GNU/GPLv2 license
    - Support for Linux and Android
    - Extraction via local port connection
  - FTK Imager: https://accessdata.com/product-download/ftk-imager-version-4-2-1
    - Commercial tool
    - Support for Windows

4. Incident Response Integration

Collection of Memory Evidence: Memory Acquisition

- **Acquisition in virtual machines**
  - VirtualBox
    - vboxmanage debugvm "Win7" dumpvmcore --filename test.elf
  - VMWare
    1. Create a snapshot of the virtual machine execution (.vmss and .vmem files are generated)
    2. vmss2core tool: https://flings.vmware.com/vmss2core??src=vmw_so_vex_mraff_549
- **Other tools for extracting processes or modules**
  - ProcDump: https://docs.microsoft.com/en-us/sysinternals/downloads/procdump
    - procdump -ma 4572
    - Single dump (fichero .dmp)
  - Windows Memory Extractor: https://github.com/reverseame/windows-memory-extractor
    - GNU/GPLv3 license
    - WindowsMemoryExtractor_x64.exe --pid 1234
    - Create sectional dump of process memory

4. Incident Response Integration

Memory Dump Analysis: *Volatility*

- **De facto standard** to analyze memory dumps
- FOSS (GNU/GPLv2 license)
- Published in 2007 in BH USA, called *Volatoools*
- Support for Windows, Linux and MacOS, in 32 and 64 bits
- Very extensive API for your own implementations
- Version 2.6 vs. Version 3
  - Python2 vs Python3
  - Version 3 is already stable! https://github.com/volatilityfoundation/volatility3

4. Incident Response Integration

First Steps with *Volatility*

- Virtual machine provided: Debian 10.10
  - Volatility 2.6 and Volatility 3.0 already installed
  - User/password: alumno / alumno
- **Help**:
  - python2/python3 vol.py –h
- **Memory dump to analyze** :
  - python2 vol.py --f mem.dmp --profile Win7SP1x86
  - The profile is only necessary in version 2.6. It indicates where are the internal structures of the SO
- *How to know the profile to use?* → imageinfo / windows.info plugins (Volatility2 / Volatility3)
  - python2 vol.py --f mem.dmp imageinfo
  - python3 vol.py --f mem.dmp windows.info
- **Plugins are always indicated at the end of the command**

4. Incident Response Integration

Detection of Indicators of Compromise with *Volatility*

- **Processes and DLLs**
  - pslist, pstree (psscan for possible rootkits)
  - dlllist, dlldump
  - handles
  - enumfuncs (list of imported and exported functions, by process/dll)
- **Process memory**
  - memmap, memdump
  - procdump
  - Vadinfo, vadwalk, vadtree, vaddump
  - evtlogs
  - iehistory
- **Network**
  - connections, connscan
  - sockets, sockscan
  - netscan (network artifacts in Win7)

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

4. Incident Response Integration

Detection of Indicators of Compromise with *Volatility*

- Kernel memory and other (internal) objects
  - modules, modscan, moddump
  - driverscan
  - filescan
- Register
  - hivescan, hivelist, hivedump
  - printkey
  - lsadump
  - userassist, shellbags, shimcache
  - dumpregistry
- Filesystem
  - mbrparser, mftparser
- Hibernation file analysis or other dumps

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

4. Incident Response Integration

Methodology for Malware Analysis

1. **Protect the memory dump**
   - Store it in read-only filesystems
   - Set special permissions to prevent accidental changes (e.g., `chattr + i`)
2. **Preliminary memory dump analysis**
   - Analyze it with different AVs and check results
3. **Data carving, file hashing, and file identification**
   - Extract content and analyze the extracted data
   - Use of several UNIX commands, pipelining them

4. Incident Response Integration

Methodology for Malware Analysis

4. Process-based Volatility plugin memory analysis

- Identify the underlying machine (`windows.info`)
- Processes (windows.pslist, windows.psscan). See differences in output
  - Another good plugin is psxview, but it is only available for Volatility2 (at the moment)
- Commands typed into a command shell (`windows.cmdline`)
- Network connections (`windows.netscan`, `windows.netstat`)
  - Analyze the IP addresses (WHOIS, DNS reputation, etc.)
  - Relationship between processes and open sockets (check the ports)
- File handles in memory (`windows.filescan`)
- Windows-thread mutexes (`windows.mutantscan`)
- Other handles (`windows.handles`)
- Drivers (`windows.driverscan`, `windows.driverirp`)
- Modules (`windows.modscan`)
- Services (`windows.svcscan`)

4. Incident Response Integration

Methodology for Malware Analysis

4. **Process-based Volatility plugin memory analysis**
   - Linked modules per process (`windows.ldrmodules` in Volatility2)
   - DLLs loaded (windows.dlllist)
   - Thread analysis (`threads` and `thdrscan`, only Volatility2)
5. **Detection and extraction of suspicious drivers, processes, and other elements of interest**
   - Create appropriate directories for storing outputs
   - For each output, analyze it with AVs and calculate hashes
   - Plugins:
     - `windows.malfind`
     - With option –dump: `windows.pslist`, `windows.dlllist`, `windows.modules`, `windows.memmap`
     - `windows.lsadump`
     - `windows.dumpfiles`
   - **Analyze extracted files using the malware analysis methodology** explained before. *Enjoy*! ☺

4. Incident Response Integration

Methodology for Malware Analysis

6. Windows Registry memory analysis
   - Check Registry hives available in the memory dump:
     - `windows.registry.hivelist`, `windows.registry.hivescan`
   - Get Registry keys: `windows.registry.printkey` (more details with --recurse)
   - Check UserAssist: `windows.registry.userassist` (useful for persistence)
7. Optional analysis
   - Relationship between device drivers and their required Windows services:
     - `windows.devicetree`

4. Incident Response Integration

Best Practices

- Establish cross-functional collaboration between IR and malware analysis teams
- Define IR and malware analysis workflows
- Conduct regular training and skill development
- Implement automated malware analysis tools
- Establish IR and malware analysis metrics to measure the effectiveness of the integration
- Share threat intelligence between IR and malware analysis teams
- Conduct post-incident analysis and lessons learned
- Emphasize continuous improvement

5. Hands-On: Malware Analysis Integrated into Incident Response

Cybersecurity Summer Bootcamp

LEON - 2023

5. Hands-On: Malware Analysis Integrated into Incident Response

LAB SESSION 2

- **Additional files for *Lab session 2***
    - https://webdiis.unizar.es/~ricardo/sbc-2023/laboratories/additional_files/wannacry.elf.tar.gz
- Follow the laboratory workbook provided on the workshop's website: https://webdiis.unizar.es/~ricardo/sbc-2023/laboratories/lab2_malware_analysis_incident_response.pdf

Cybersecurity Summer Bootcamp

LEÓN - 2023

3 al 13 julio de 2023

León, España

#CyberSBC2023

incibe.es/en/events/summer-bootcamp

Organized by:

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA PRIMERA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

incibe_
INSTITUTO NACIONAL DE CIBERSEGURIDAD

OAS More rights for more people

With the collaboration of:

Canada

AYUNTAMIENTO DE LEÓN

LEÓN CONGRESOS

universidad de león