An Overview Of Modern Windows Malware Analysis
A Researcher's Perspective

Simone Aonzo

Twitter: @packm4d
Website: https://simoneaonzo.it
Goals of this talk

- (Windows) malware analysis from a researcher's point of view
  - Emphasis on the state of the art
  - Oriented to large-scale analysis
- But... practical and modern
  - Everything I will mention is tested, working and maintained
  - Slides are verbose for quick reference
- Not self-referential
  - Not about our papers
  - About “how” we made them
Malware - Definition

Malware (malicious software) is any software intentionally designed to unknowingly interferes with security (CIA triad) and privacy of users and organizations

Reasons?

- Making money 😐
- Activism
- Data theft
- Vandalism/Prank
- Nation state-sponsored operations
- ...


Malware Analysis - (My) Definition

“Program analysis of a software that does not want to be analyzed”
Malware Types

➢ Infection
  ○ **Worm**: self-replicate/propagate
  ○ **Virus**: requires human interaction to spread
  ○ **Trojan**: benign appearance but hidden malicious features

➢ Features
  ○ **Adware**: displays unwanted or malicious advertising
  ○ **Bot**: performs a task given a remote command
  ○ **Exploit**: exploits a software vulnerability to gain authorized access
  ○ **HackTool**: exploiting, attack and scanning tools
  ○ **Ransomware**: encrypts device's data for ransom
  ○ **RootKit**: stealth and actively hiding software with elevated permissions
  ○ **Spyware**: software that invades the user’s privacy
  ○ …
Microsoft Windows is an amusement park for malware authors

- Native support for Android apps 😞
- No application sandbox (Process Injection)
- Support for old technologies (Classic Visual Basic)
- Scripting languages (Batch, Powershell, Javascript)
- Office Macro (VBA, Javascript)
- Portable Executable format (exe, dll)
  - Can “hide” a virtual machine (.NET, VB, Python)
  - Different structure w.r.t. language (C++, Go, Rust)
  - Same structure w.r.t. packer/protector (UPX, Themida) and virtual machine

“Survivalism: Systematic Analysis of Windows Malware Living-Off-The-Land” S&P 2021
Types Of Malware Analysis

- STATIC
- DYNAMIC
- MEMORY
- HYBRID
Static techniques reason about a program without executing it.

1. **Code** (original or lifted to an Intermediate Representation)
   - Data-flow analysis
     - Reason on the Control-Flow Graph (CFG)
     - Determine possible set of values calculated at various points in the program
   - Abstract interpretation
     - The program is interpreted (“executed”) over an abstract domain
   - Symbolic execution
     - Determine what inputs cause each part of a program to execute

2. **File structure**
   - Byte Patterns
   - Executable file format
Static (code) Analysis Tools [1/2]

There is no favorable wind for the sailor who does not know where to go” – Seneca

● What type of file are you analyzing?
○ In this talk: Portable Executable (PE) format

● What type of PE file are you analyzing?
   🤔
   ○ Native, .NET, VisualBasic, Python, AHK, …

● Use the correct tool to get the **actual** code

… it is a native PE file. Ok, which compiler/language?

○ C, C++, Go, Rust, …
Static (code) Analysis Tools [2/2]

- **IDA Pro** by Hex-Rays
  - Pros: since 1991, fast, best decompiler, awesome Windows support ⇒ industry standard
  - Cons: proprietary, very expensive (decompiler not included), low-level API

- **Ghidra** by National Security Agency (NSA)
  - Pros: oss, high-level API, collaborative projects, built-in program analysis
  - Cons: written in Java, immature debugger

- **Binary Ninja** by Vector 35
  - Pros: high-level API, multi-level IL ⇒ best for program analysis
  - Cons: proprietary, bugs with complicated analysis

- Last but not least
  - [https://github.com/radareorg/radare2](https://github.com/radareorg/radare2)  
  - [https://github.com/rizinorg/rizin](https://github.com/rizinorg/rizin)
  - [https://github.com/cea-sec/miasm](https://github.com/cea-sec/miasm)
  - [https://github.com/angr/angr](https://github.com/angr/angr)
Static (file) Analysis Tools

Multiplatform and suited for large-scale analysis

- **Yara** – [https://github.com/VirusTotal/yara](https://github.com/VirusTotal/yara)
  - Binary patterns signatures ⇒ fast
  - Signatures DB scattered around the internet – [https://github.com/InQuest/awesome-yara](https://github.com/InQuest/awesome-yara)

- **Detect It Easy** – [https://github.com/horsicq/Detect-It-Easy](https://github.com/horsicq/Detect-It-Easy)
  - Fine-grained signatures ⇒ slow
  - Signatures DB unique and well maintained

- **Capa** – [https://github.com/mandiant/capa](https://github.com/mandiant/capa)
  - Detects capabilities (what a program can do) in executable files

- **Manalyze** – [https://github.com/JusticeRage/Manalyze](https://github.com/JusticeRage/Manalyze) – [https://manalyzer.org/](https://manalyzer.org/)
  - Combines several tools + plugin architecture

- **Python Modules**
  - **Pefile** – PE files parsing – [https://github.com/erocarrera/pefile](https://github.com/erocarrera/pefile)
  - **Signify** – Verifies PE Authenticode-signed binaries – [https://github.com/ralphje/signify](https://github.com/ralphje/signify)
  - **LIEF** – Parsing and editing of several executable file formats – [https://github.com/lief-project/LIEF](https://github.com/lief-project/LIEF)
Types Of Malware Analysis

- STATIC
- DYNAMIC
- MEMORY
- HYBRID
Dynamic Analysis

Executing a sample inside an isolated and instrumented environment to analyse its behavior

Also known as: **Sandbox**

- **Runtime Environment**
  - Virtual Machines (VM) – virtualized or emulated hw
  - Bare metal

- **Analysis Component**
  - In-guest
    - User-space (debugger or Dynamic Binary Instrumentation tool)
    - Kernel-space (module or driver)
  - Out-of-guest
    - Hypervisor or Emulator “APIs”
Dynamic Analysis Tools

Requirements: instruction granularity + suitable for large-scale

1. **Intel Pin** - DBI
   - [https://www.intel.com/software/pintool](https://www.intel.com/software/pintool)
   - **Pros**: well documented, stable, full control
   - **Cons**: just x86-64, closed source

2. **PANDA** - emulator (QEMU) based
   - [https://github.com/panda-re/panda](https://github.com/panda-re/panda)
   - **Pros**: multiarch, oss, record & replay executions, taint engine
   - **Cons**: just monitoring, records need disk space

3. **Triton** - DBA
   - [https://github.com/JonathanSalwan/Triton](https://github.com/JonathanSalwan/Triton)
   - **Pros**: multiarch, oss, different inputs (Pin, QEMU, ...), symbolic|taint engine
   - **Cons**: bugs
Large-Scale Dynamic Analysis

Two approaches

1. Single machine, multiple emulators
   - Best control over the instances
     - But you have to write all the management APIs
   - If the machine gets stuck...

2. Multiple machines, single runtime environment
   - Type-1 hypervisor (ESXi, KVM, ...) and management (vCenter, Proxmox, ...)
   - Off-the-shelf virtualization management APIs
     - Not meant for being stressed
Large-Scale Dynamic Analysis – Tips

● Prepare a Windows machine
  ○ Minimum: Windows 7 x32 with 2 GB of RAM
  ○ Make it look “used”: install programs, surf the internet, populate with documents, ...
  ○ Install SSH for remote management and take a snapshot at the end

● Buy RAM 💸 and abuse RAM Disks

● Try to use the original filename of the sample
  ○ How? Check VirusTotal report

● State-Of-The-Art: Run the sample for at least 2 minutes
  ○ But consider the overhead introduced!

● Simulate common internet services
  ○ https://www.inetsim.org/

● Mitigate evasive techniques...

“Spotless sandboxes: Evading malware analysis systems using wearand-tear artifacts”
S&P 2017

“Does Every Second Count?
Time-based Evolution of Malware Behavior in Sandboxes”
NDSS 2021
Evasive Techniques

Malware does not want to be analyzed ⇒ evasive techniques

Taxonomy

- Anti Debug
- Anti Dump
- Anti Instrumentation
- Code Injections
- Resource Profiling
- VM Checks
- Timing Attacks (time stalling & runtime measurements)

Resources

- Public evasive techniques: https://github.com/LordNoteworthy/al-khaser
- Detection and Mitigation: https://github.com/Maff1t/JuanLesPIN-Public

"On the dissection of evasive malware" IEEE Forensics and Security 2020

“Longitudinal Study of the Prevalence of Malware Evasive Techniques” arXiv 2021
Follow the white rabbit 🐰

Malware often jumps between different technologies, for example Ursnif

Phase 1: Infection
- Stage 1: System Exploit
- Stage 2: Binary (Dropper) Loading

Phase 2: Callback
- Stage 3: Callback
- Stage 2: Data Exfiltration

“phone home”
Large-Scale Dynamic Analysis – Architecture

- VM1
- VM2
- VMx
- Orchestrator
- PostgreSQL
- NAS
- Why not...? 😐
  - MongoDB
  - ELK stack
  - Apache Spark
  - ...
Types Of Malware Analysis

- **STATIC**
- **DYNAMIC**
- **MEMORY**
- **HYBRID**
Memory (Analysis|Forensics)

Analyzing the computer's RAM for forensic artifacts

- Large research area ⇒ focus on malware
- Malware often write components only in memory (e.g., unpacking)
  - ... but can only exists exclusively in RAM (AKA, fileless malware)

Two phases

- Memory Acquisition
  - Suspend VM (.vmem)
  - https://github.com/Velocidex/WinPmem
    - ⇒ Minidump crash report

- Analysis of memory dump
  - https://github.com/volatilityfoundation/volatility3/
    - procdump plugin
  - https://github.com/skelsec/minidump
Pipeline

DATASET

FILTERING

ANALYSIS

PROCESSING
Datasets

- [Top] https://www.virustotal.com/
  - Insanely expensive 💰
- https://www.virussign.com/
  - “Cheap” live feed
- https://virusshare.com/
  - Torrents (must be tidy up)
- https://urlhaus.abuse.ch/
  - Malicious URLs
- https://bazaar.abuse.ch/
  - Advanced APIs
- https://www.vx-underground.org/
  - APT samples, organized in families, and source codes
- https://malshare.com/
  - Daily digest, researchers often upload famous samples
Filtering

1. File structure
   - Compiler, packer, protector, ...
   - https://github.com/packmad/Siggregator

2. Family distribution is crucial
   - CARO naming convention 😞
   - VirusTotal report ➔ AVClass2 ➔ family
   - https://github.com/malicialab/avclass

“AVclass2: Massive Malware Tag Extraction from AV Labels” ACSAC 2020
Vendors and Engines

<table>
<thead>
<tr>
<th>Vendors</th>
<th>Website</th>
<th>Country</th>
<th>Third Party Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avast</td>
<td><a href="https://www.avast.com">https://www.avast.com</a></td>
<td>Czech Republic</td>
<td></td>
</tr>
<tr>
<td>AVG</td>
<td><a href="https://www.avg.com">https://www.avg.com</a></td>
<td>Czech Republic</td>
<td>Avast</td>
</tr>
<tr>
<td>Avira</td>
<td><a href="https://www.avira.com">https://www.avira.com</a></td>
<td>Germany</td>
<td></td>
</tr>
<tr>
<td>Bitdefender</td>
<td><a href="https://www.bitdefender.com">https://www.bitdefender.com</a></td>
<td>Romania</td>
<td></td>
</tr>
<tr>
<td>Check Point (ZoneAlarm)</td>
<td><a href="https://www.zonealarm.com">https://www.zonealarm.com</a></td>
<td>Israel</td>
<td>Kaspersky</td>
</tr>
<tr>
<td>Dr. Web</td>
<td><a href="https://www.drweb.com">https://www.drweb.com</a></td>
<td>Russia</td>
<td></td>
</tr>
<tr>
<td>Emsisoft</td>
<td><a href="https://www.emsisoft.com">https://www.emsisoft.com</a></td>
<td>New Zealand</td>
<td>Bitdefender</td>
</tr>
<tr>
<td>ESET</td>
<td><a href="https://www.eset.com">https://www.eset.com</a></td>
<td>Slovakia</td>
<td></td>
</tr>
<tr>
<td>F-Secure</td>
<td><a href="https://www.f-secure.com">https://www.f-secure.com</a></td>
<td>Finland</td>
<td>Avira</td>
</tr>
<tr>
<td>G Data</td>
<td><a href="https://www.gdata.de">https://www.gdata.de</a></td>
<td>Germany</td>
<td>Bitdefender</td>
</tr>
<tr>
<td>K7 Computing</td>
<td><a href="https://www.k7computing.com">https://www.k7computing.com</a></td>
<td>India</td>
<td></td>
</tr>
<tr>
<td>Kaspersky</td>
<td><a href="https://www.kaspersky.com">https://www.kaspersky.com</a></td>
<td>Russia</td>
<td></td>
</tr>
<tr>
<td>Malwarebytes</td>
<td><a href="https://www.malwarebytes.org">https://www.malwarebytes.org</a></td>
<td>USA</td>
<td></td>
</tr>
<tr>
<td>McAfee</td>
<td><a href="https://www.mcafee.com">https://www.mcafee.com</a></td>
<td>USA</td>
<td></td>
</tr>
</tbody>
</table>

FYI engines are a software...

- Reverse engineering
- Exploit

Consumer: https://www.av-comparatives.org/list-of-consumer-av-vendors-pc/
Enterprise: https://www.av-comparatives.org/list-of-enterprise-av-vendors-pc/
Research In Malware Analysis

1. Machine Learning
2. Adversarial Machine Learning
3. (De)obfuscation
4. Measurements
5. Big-data Algorithms
6. Dynamic Analysis Improvements
7. Operations
8. Memory Forensic
9. Humans
Books

- [2014] Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation
- [2014] The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and MAC Memory
- [2018] Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware
- [2018] Malware Data Science: Attack Detection and Attribution
- [2019] Rootkits and Bootkits: Reversing Modern Malware and Next Generation Threats
- [2020] Learn Computer Forensics: A beginner's guide to searching, analyzing, and securing digital evidence
- [2021] Malware Analysis Techniques: Tricks for the triage of adversarial software

Learning — Reading

- [2019] Sandworm: A New Era of Cyberwar and the Hunt for the Kremlin's Most Dangerous Hackers
- [2021] This Is How They Tell Me the World Ends: The Cyberweapons Arms Race
The End

Thanks for your attention