An Overview Of Modern Windows Malware Analysis

Where We Are and Where We Are Going

WoRMA 2023
2023-07-07, Delft, Netherlands 🇳🇱

Simone Aonzo

EURECOM
Sophia Antipolis

$ whoami

- ATM: assistant professor @ Eurecom 🇫🇷
  - Mobile|Android System Security
  - Malware Analysis (Android and Windows)
  - Humans in the Cybersecurity Loop (i.e., Phishing and User Study)
  - Network Security
- Alma mater @ University of Genoa 🇮🇹
  - Master and Ph.D. in Computer Science and Systems Engineering
- Work experience
  - Android Pentester and Malware Analyst
- Contacts
  - Website: https://simoneaonzo.it
  - No namesakes ⇒ Google *name surname*

This talk!

- Windows malware analysis from a researcher's point of view
  - Emphasis on the state of the art
  - Oriented to large-scale analysis
- “Data-oriented”
  - What/how we analyze determines our results
- I present some results we obtained
  - With emphasis about “how” we made them

Agenda 📋

Where We Are

1. Malware, the tools and how/what to analyse
2. Creating an analysis pipeline

3. Humans vs. Machines

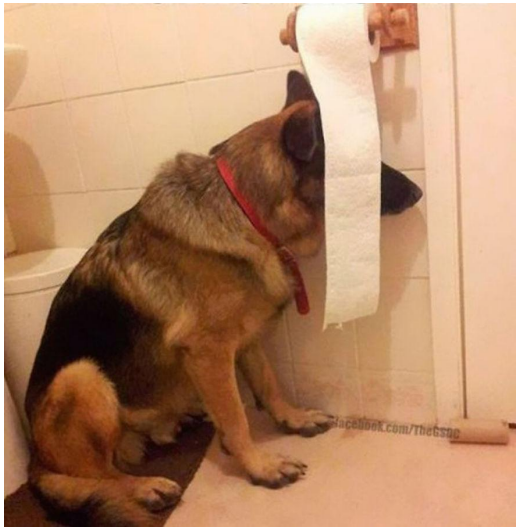
Where We Are Going

Agenda 📋

1. **Malware, the tools and how/what to analyse**
2. Creating an analysis pipeline
3. Humans vs. Machines

Malware Analysis - (My) Definition

*"**Program analysis** of a software that does not want to be analyzed"*



Malware Types

- *Infection*
  - **Worm**: self-replicate/propagate
  - **Virus**: infect other programs to include a possibly evolved copy of itself
  - **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
  - ...

Windows Malware

Microsoft Windows is an amusement park 🎢 for malware authors ☣️

- Native support for Android apps
- No applicati “Survivalism: Systematic Analysis of Windows Malware Living-Off-The-Land” S&P 2021
- Support for old technologies (Classic Visual Basic: Final release 6.0 / 1998; 25 years ago)
- Scripting languages (Batch, Powershell, Javascript)
- Office Macro (VBA, Javascript)
- Portable Executable (PE) format [. exe]
    - Can “hide” a virtual machine (.NET, VB, Python)
    - Different structure w.r.t. language/compiler (C++, Go, Rust)
    - Same structure w.r.t. packer/protector (UPX, Themida)

Types Of Malware Analysis

STATIC

DYNAMIC

MEMORY

HYBRID

“SOK: (State of) The Art of War: Offensive Techniques in Binary Analysis” S&P 2016

Static Analysis

1. **Code** (original or lifted to a Intermediate Representation)
   - Data-Flow Analysis
     - Tracks the possible values of variables or expressions at each program point
     - Reason on the Control-Flow Graph (CFG)
   - Abstract Interpretation
     - Systematically explores all executions by a series of over-approximations
     - Uses abstract domains and operators to model the semantics
   - Symbolic Execution
     - It uses symbolic exprs to represent the values of variables and path conditions
     - Relies on constraint solvers to check the feasibility of each path
2. **File structure**
   - Byte Patterns
   - Executable File Format

Static Analysis… what?

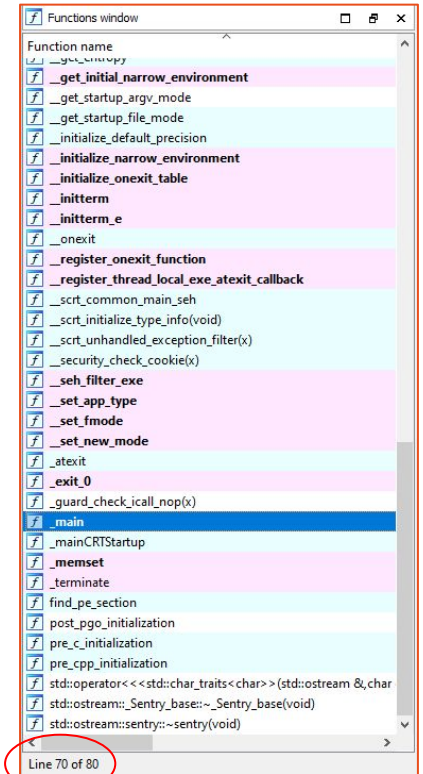
“*There is no favorable wind for the sailor who does not know where to go*” – Seneca

- What type of file are you analyzing?
  - In this talk: Portable Executable (PE) format
- What is the target “architecture”? 🤔
  - Native → Target: CPU
    - However, different code structure w.r.t. language/compiler (C++, Go, Rust)
  - Non-Native → Target: virtual machine/interpreter
    - E.g., .NET, Classic Visual Basic , Python, AutoHotkey, …
    - The exe is just a wrapper around a more complex runtime environment
  - Use the correct tool to get the **actual** code

Native PE – Compiler/Language

The different code structure w.r.t. compiler/language, e.g.

C++



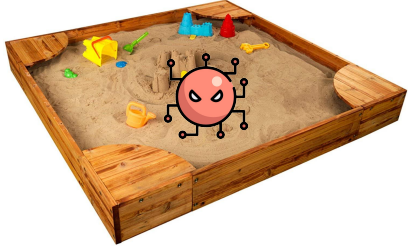
Rust

Non-Native PE – Internal/External VM

Non-Native PE files embeds the “bytecode” and need a “VM” to run it

1. **External** VM
   - Assumed that it is already installed on the system, e.g.:
     - `mscoree.dll` – .NET
     - `msvbvm(50|60).dll` – Classic Visual Basic
2. **Internal** VM
   - Embedded in the executable
   - ⇒ Large file size
   - Most common in malware: AutoHotKey~AutoIt and PyInstaller
   - ⚠️ .NET can also be embedded in a stand-alone file

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 Binary Analysis Tools

Requirements: instruction granularity + suitable for large-scale

1. **Intel Pin** - DBI
   - https://www.intel.com/software/pintool
   - Pros: well documented, stable, full control
   - Cons: just x86-64, closed source, learning curve
2. **PANDA** - Emulator (QEMU) based
   - 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
   - Pros: multiarch, oss, different inputs (Pin, QEMU, ...), symbolic|taint engine
   - Cons: bugs

Large-Scale Dynamic Analysis

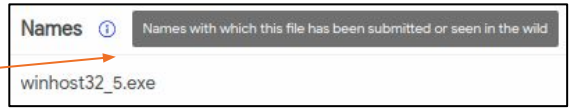
Two approaches

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 wi
  - 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 introd
- 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

40-92% 🤔 of malware use at least one evasive technique

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

“Longitudinal Study of the Prevalence of Malware Evasive Techniques” arXiv 2021

Taxonomy

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

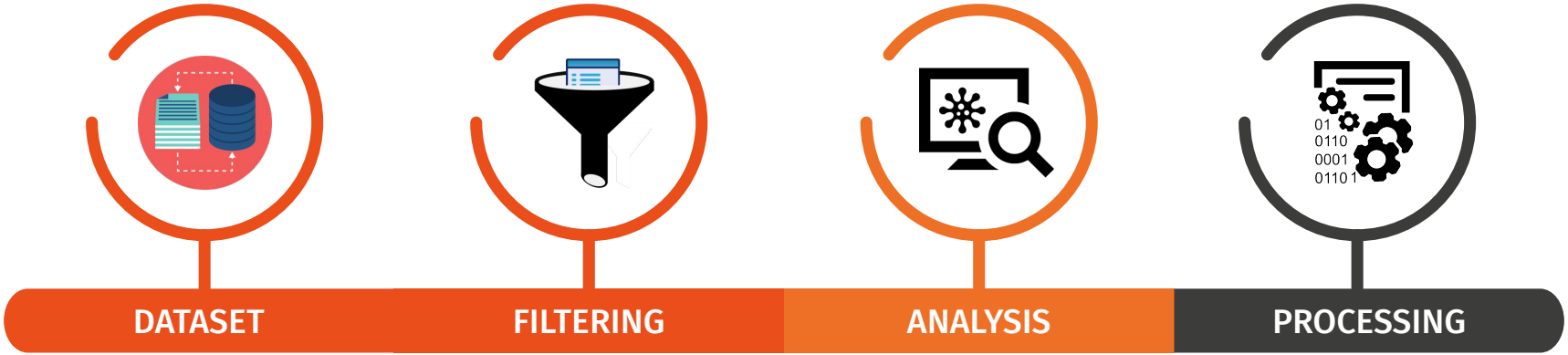
Resources

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

Agenda 📋

1. Malware, the tools and how/what to analyse
2. **Creating an analysis pipeline**
3. Humans vs. Machines

Pipeline



Datasets

- 🔝 https://www.virustotal.com/ 🔝
  - Insanely expensive 💸
- https://www.virussign.com/
  - “Cheap” live feed
- https://virusshare.com/
  - Torrents (must be cleaned 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, installer…
   - https://github.com/packmad/Siggregator
2. Family
   - CARO naming convention 😓
   - VirusTotal report ➡ AVClass2 ➡ family
   - https://github.com/malicialab/avclass

Acronis (Static ML)	Suspicious	Ad-Aware	Generic.TeslaCryptC.B878C4A3
AhnLab-V3	Trojan/Win32.Poseidon.R230029	Alibaba	TrojanDownloader:Win32/Zdowbot.4588...
ALYac	Generic.TeslaCryptC.B878C4A3	Antiy-AVL	Trojan/Generic.ASMalwS.1A2A8BC
Avast	Win32:Malware-gen	AVG	Win32:Malware-gen
Avira (no cloud)	TR/Crypt.XPACK.Gen3	BitDefender	Generic.TeslaCryptC.B878C4A3
BitDefenderTheta	AI:Packer.605213541F	Bkav Pro	W32.AIDetect.malware2
CAT-QuickHeal	Trojan.GenericRI.S21298173	ClamAV	Win.Malware.Teslacryptc-7652404-0
Comodo	Malware@#2ud0albu08v3v	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
Cybereason	Malicious.15d681	Cylance	Unsafe
Cynet	Malicious (score: 100)	DrWeb	Trojan.Chanitor.28
Elastic	Malicious (high Confidence)	Emsisoft	Generic.TeslaCryptC.B878C4A3 (B)

TeslaCrypt

"AVclass2: Massive Malware Tag Extraction from AV Labels" ACSAC 2020

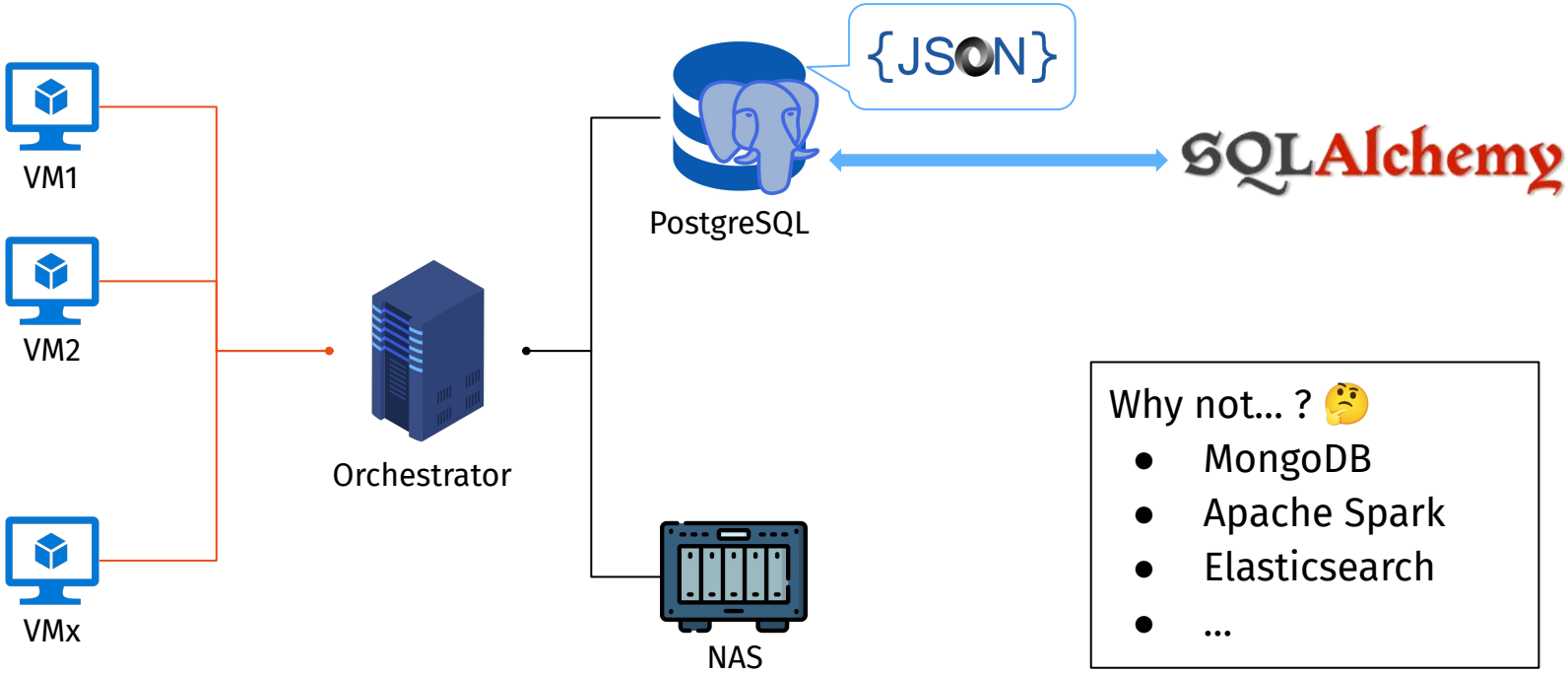
AVclass Family Filtering 🚨

AVclass output strips information

- Sometimes family name == campaign
- Within the same family you have different
  - Stages ⇒ Types
    - E.g., trojan/dropper and virus/ransomware
  - Versions
    - E.g., updated crypto algorithm in virus/ransomware
  - Variants
    - E.g., trojan/dropper detected ⇒ new obfuscation
  - Technologies
    - E.g., dropper created with pyinstaller, ransomware in rust

... let's think about it when we build a dataset

Large-Scale Dynamic Analysis + Processing



Why not… ? 🤔

- MongoDB
- Apache Spark
- Elasticsearch
- …

Real-World Numbers

"Decoding the Secrets of Machine Learning in Malware Classification: A Deep Dive into Datasets, Feature Extraction, and Model Performance" CCS 2023

Dataset	Samples	Families
Malware Balanced (M_B)	67,000	670
Benign (B)	16,611	-
Malware Unbalanced (M_U)	18,000	1,500
Malware Generic (M_G)	16,500	-
All	118,111	-

118,111 samples dynamically analyzed (with IntelPin) in less than a month

- Properly filtered by removing non-natives, installers, and DLLs
- 21 Proxmox VMs: 2 (dedicated) CPUs and 2GB RAM each
  - Non-persistent storage (in a RAMdisk)

Agenda 📋

1. Malware, the tools and how/what to analyse
2. Creating an analysis pipeline
3. **Humans vs. Machines**

"Humans vs. Machines in Malware Classification" Usenix-Security 2023

Machine Learning 🤖

ML offers an easy-to-deploy and scalable solution

- Vast amount of research on ML-based malware classification
- ML works great in applications like speech/text/image recognition
  - Pronunciations/Characters/Objects remain relatively constant over time
- Models cannot go beyond the training data
  - Attackers aware of this limitation will always be one step ahead
  - ⇒ Malware constantly changes to evade detection
  - *Which features really influences the accuracy of classification?*

🐒 Humans vs. Machines 🤖

What info do humans and machines use to decide if a sample is benign or malicious?

- 110 humans 🐒
  - 38 Experts
    - Renowned cybersecurity companies + Academic researchers
  - 72 Novices – attended at least a course malware analysis
    - Students + Beginner CTF players
- State-of-the-art Machine Learning algorithms 🤖
  - Random Forests (RF) 👴
  - Convolution Neural Network (CNN) 👶

Experiment setup (for the humans 🐒)

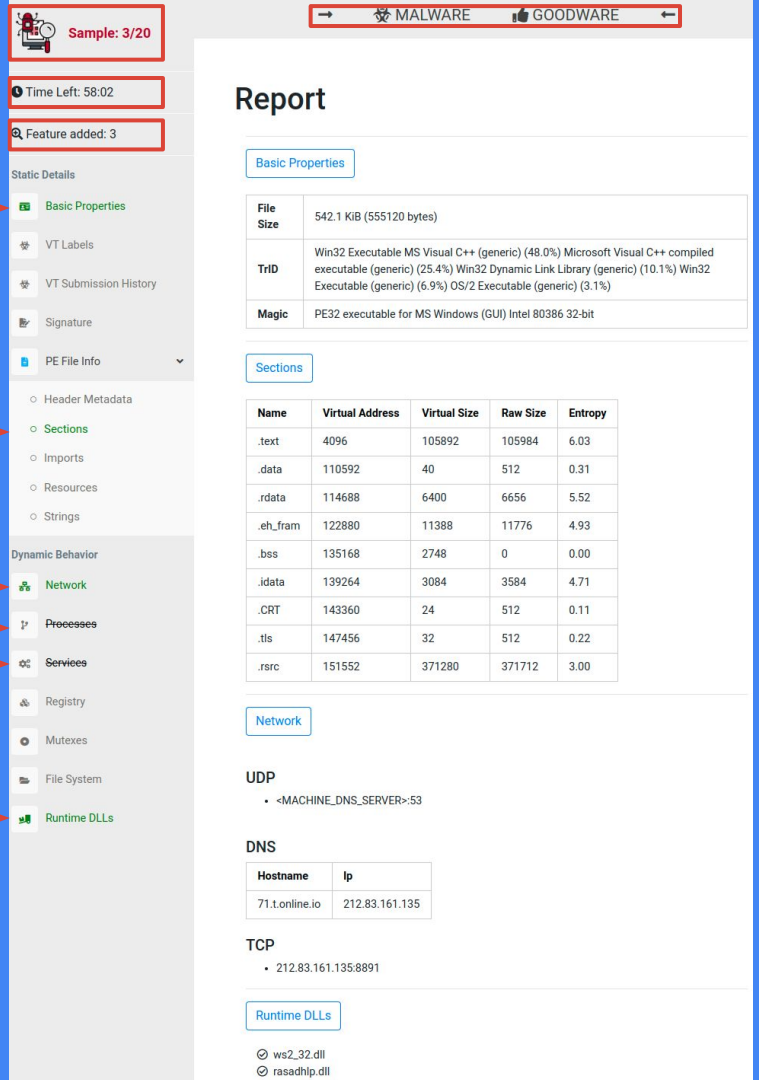
We designed an web-based game: “*Detect Me If You Can!*” [DMIYC]

- Participants have to classify 20 suspicious files based on sandox report
  - Static and dynamic
- Design elements
  - Points: numerically represent a player’s outcome
  - Leaderboard: rank players according to their relative success
- Players must correctly classify (goodware/malware) the higher number of samples
  - Using **as few features as possible**
  - ⇒ Players have to “buy” each feature

Scoring Mechanism

Players start with a blank report

- Adds new features to the report by choosing them from a pre-defined catalog of 15 features
- Until she has gained enough information to make a **confident** binary classification
- 20 samples → 20 rounds
- 20 **potential** points for each round
  - When she buys a new feature → potential_points -= 1
    - “Empty feature” → potential_points -= 0
- If the sample is correctly classified → the player gets the remaining potential points
  - Otherwise zero 😞
- Final score = sum of all points obtained in each round * number of correct answers
  - ⇒ Highest possible score in DMIYC is 19*20*20 = 7600



Report

Basic Properties

File Size	542.1 KiB (555120 bytes)
TrID	Win32 Executable MS Visual C++ (generic) (48.0%) Microsoft Visual C++ compiled executable (generic) (25.4%) Win32 Dynamic Link Library (generic) (10.1%) Win32 Executable (generic) (6.9%) OS/2 Executable (generic) (3.1%)
Magic	PE32 executable for MS Windows (GUI) Intel 80386 32-bit

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy
.text	4096	105892	105984	6.03
.data	110592	40	512	0.31
.rdata	114688	6400	6656	5.52
.eh_fram	122880	11388	11776	4.93
.bss	135168	2748	0	0.00
.idata	139264	3084	3584	4.71
.CRT	143360	24	512	0.11
.tls	147456	32	512	0.22
.rsrc	151552	371280	371712	3.00

Network

UDP

- <MACHINE_DNS_SERVER>:53

DNS

Hostname	Ip
71.t.online.io	212.83.161.135

TCP

- 212.83.161.135:8891

Runtime DLLs

- ws2_32.dll
- rasadhlp.dll

Samples of the game

Sample	M|G	Malware Family	Description
1	M	hematite	file infector
2	M	kryptik	trojan
3	M	onlineio	adware
4	G	-	*Dell Backup & Recovery*
5	G	-	*TeamViewer*
6	M	sysn	dropper
7	G	-	*Google Chrome installer*
8	M	nanolocker	ransomware
9	M	doomjuice	worm
10	M	zbot	spyware
11	G	-	*Fallout 4 component*
12	G	-	*custom Autohotkey*
13	M	nitol	backdoor
14	G	-	*DOSBox*
15	M	zbot	packed spyware
16	M	nanocore	RAT
17	G	-	*WinDirStat*
18	G	-	*Java Update Checker*
19	G	-	*Media Player Classic*
20	M	zdowbot	keylogger downloader

Samples of the game + VirusTotal T/F P/N

True Positive → 2

True Negative → 4

False Negative → 15

False Positive → 17

Sample	M|G	Malware Family	Description
1	M	hematite	file infector
2	M	kryptik	trojan
3	M	onlineio	adware
4	G	-	*Dell Backup & Recovery*
5	G	-	*TeamViewer*
6	M	sysn	dropper
7	G	-	*Google Chrome installer*
8	M	nanolocker	ransomware
9	M	doomjuice	worm
10	M	zbot	spyware
11	G	-	*Fallout 4 component*
12	G	-	*custom Autohotkey*
13	M	nitol	backdoor
14	G	-	*DOSBox*
15	M	zbot	packed spyware
16	M	nanocore	RAT
17	G	-	*WinDirStat*
18	G	-	*Java Update Checker*
19	G	-	*Media Player Classic*
20	M	zdowbot	keylogger downloader

VirusTotal Impact

Sample	M|G	Malware Family	Description
1	M	hematite	file infector
True Positive →2	M	kryptik	trojan
3	M	onlineio	adware
True Negative →4	G	-	*Dell Backup & Recovery*

Sample №	2	4	17	15
Type	TP	TN	FP	FN
Class	Malicious	Benign	Benign	Malicious
VT Matches	10	0	5	0
Experts	29/31 (93%)	27/29 (93%)	23/28 (82%)	28/28 (100%)
Novices	59/64 (92%)	57/63 (90%)	20/61 (32%)	49/61 (80%)

False Negative →15	M	zbot	packed spyware
16	M	nanocore	RAT
False Positive →17	G	-	*WinDirStat*
18	G	-	*Java Update Checker*
19	G	-	*Media Player Classic*
20	M	zdowbot	keylogger downloader

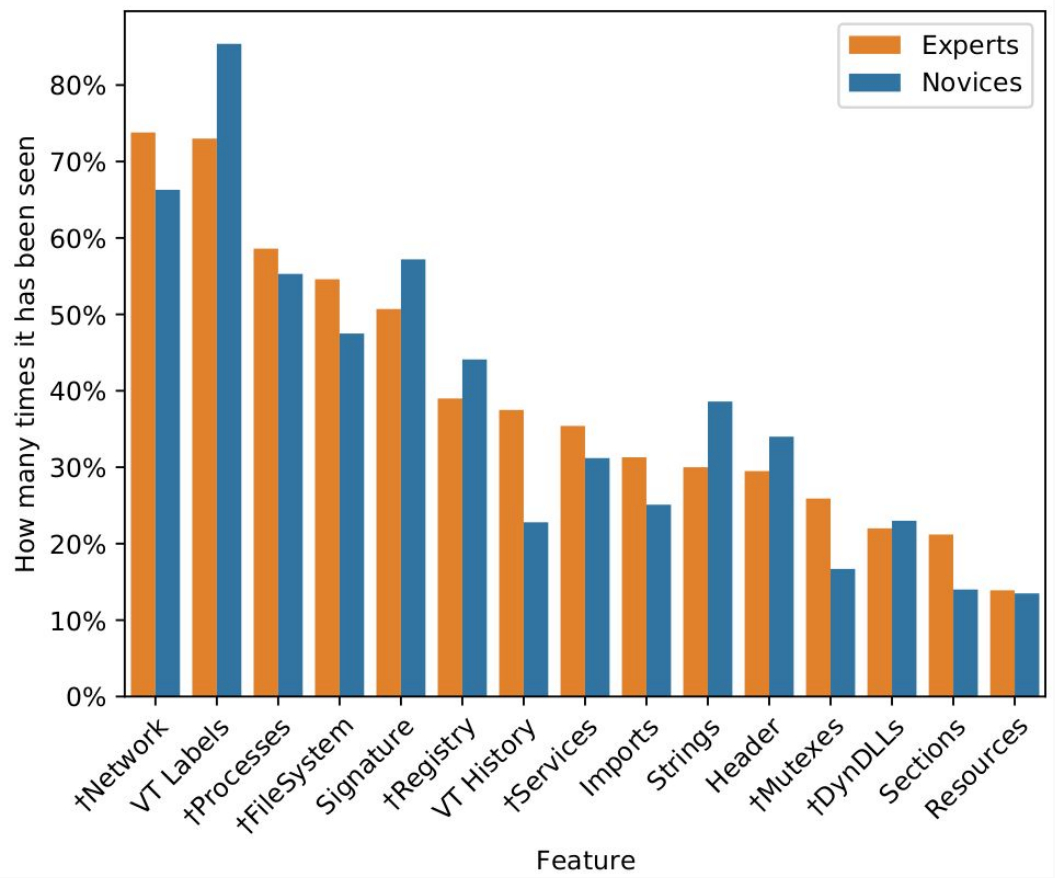
View of the results

Metric	Experts Novices	Min	Max	Avg	Std	Median
Time	**E**	7:48	56:48	29:04	08:53	26:51
	N	8:14	59:58	44:31	10:05	46:32
Score	**E**	2310	5339	4103	742	4329
	N	1072	6042	3072	1054	2991
Right Answers	**E**	13	19	16.1	1.4	16
	N	8	19	13.7	2.4	14
Total Used Features	**E**	42	165	82.0	35.1	70
	N	37	146	81.7	27.5	68.5
Unique Used Features	**E**	7	16	13.4	2.6	14
	N	7	16	14.1	2.1	15

Statistically-significant differences between E|N? Welch's t-test

1. Time needed to complete the game
2. Final score
3. Number of correct answers
4. … features? 🤔

Feature Ranking



Most used top 5 features

	All	Correct	Misclassified
Experts	†Network	†Network	†Network
	VT labels	VT labels	VT labels
	†Processes	†Processes	†Processes
	†FileSystem	†FileSystem	†FileSystem
	Signature	Signature	Signature
Novices	VT labels	VT labels	VT labels
	†Network	†Network	†Network
	Signature	Signature	†Processes
	†Processes	†Processes	Signature
	†FileSystem	†FileSystem	†FileSystem

🤖 Machine Learning Players – Dataset

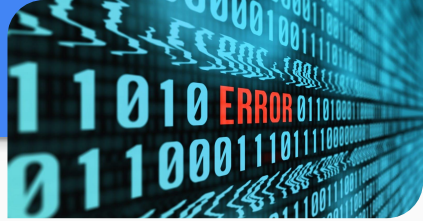
- Benchmark Dataset: 21,944 reports from VirusTotal
  - 50% (10,972) malware
    - [2018, 2020]
    - Detection >= 21 antivirus engines
    - No malware families were over-represented (AVClass2)
      - Most frequent family had 125/10,972 occurrences (1.1%)
  - 50% (10,972) goodware
    - Clean Windows 10 machine
    - Installed all community-maintained Chocolatey software
    - Extracted all the executable files present on the hard disk
    - Filtered by detection < 3 (e.g., hacking/scanning tools)

🤖 Machine Learning Players – Validation

Validation of the classification accuracy using Machine Learning Players

- Balanced dataset containing VirusTotal reports
  - 10,972 goodwares and 10,972 malware samples.
- Training: 80% of the goodwares and malwares are selected randomly
- Testing: remaining 20% of the samples
- 5-fold cross-validation to derive averaged AUC-ROC scores
- Both Machine Learning players reach high classification accuracy
  - 0.9962 for RF and 0.9950 for CNN

Humans 🐒 vs. 🤖 Machines



- Results (reminder, machines had the "*all feature advantage*")
  - Human Experts: 16/20 (avg == median)
  - RF: 17/20 – CNN: 16/20
- Machines errors
  - Both samples **3** [M] and **17** [G]
    - Sample **3** connects to a malicious domain
    - Human experts who correctly classified it looked at the "Network"
  - RF: sample **12** [G]
  - CNN: samples **4** [G] and **15** [M]
- The misclassified game files by the ML players and the human subjects are different

Humans 🐒 vs. 🤖 Machines – Feature Ranking

- We adopt SHAP as a model-agnostic model explanation tool
- Not the recursive feature elimination using out-of-bag error (OOB) evaluation of RF
  - Inclines to overestimate the importance of high-cardinality categorical variables

#	RF	CNN	Expert Humans
1	Resources	Resources	†Network
2	†Services	Sections	†Processes
3	Header Metadata	†Network	†FileSystem
4	†Network	†Runtime DLLs	Signature
5	Signature	Header Metadata	†Registry
6	†Runtime DLLs	Signature	†Services
7	Strings	†Services	Imports
8	Sections	†FileSystem	Strings
9	Imports	Strings	Header Metadata
10	†Mutexes	†Registry	†Mutexes
11	†Registry	†Mutexes	†Runtime DLLs
12	†FileSystem	Imports	Sections
13	†Processes	†Processes	Resources

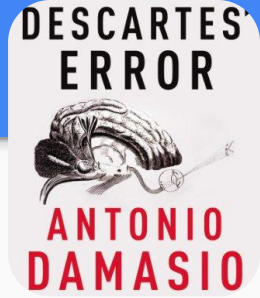
Takeaways (1/2)

- Experts and Novices base their decisions on the same set of features
- Humans and Machines agree on the importance of two features
  - *“Network traffic”* and a *valid “signature”*
- Machines rank top “*resources”*, Humans last – always take a look at it analysts!
- During goodware classification
  - Experts used more features
  - Novices make the majority of mistakes
  - ⇒ We must teach that one must check for the absence of any malicious signs!

Takeaways (2/2)

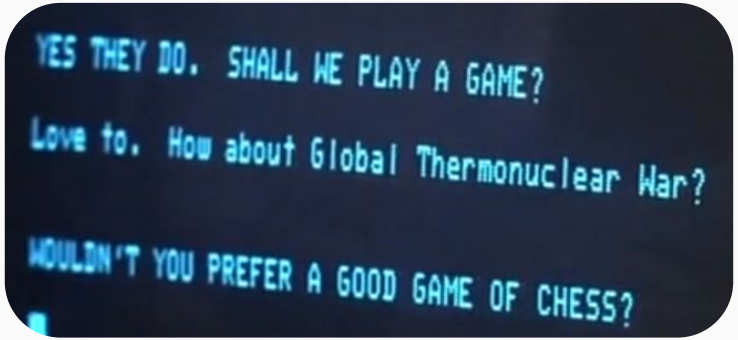
- Experts classify samples by using less than 1/3 of the available features
  - With a preference for dynamic behaviour
- The problem of missing dynamic features
  - Missing observations weaken the trustworthiness of the ML-based decision
- Impact on the human-computer interaction; machines must show to humans
  - OSINT data to humans (e.g., IP and domains info)
  - What are the most significant features that helped classify the sample
    - The analyst can focus on the others and bridge the cognitive gap

Final Remark



The patient Elliot of Antonio Damasio

- A (successfully cured) brain tumor wounded the frontal lobe tissue in his brain
- Fully recovered, BUT: loss of his job, divorced, bankruptcy, etc.
- Several doctors declared that his mental faculties were intact ⇒ denied assistance
- Damasio tested him with lots of emotionally charged images: NO RESPONSE 💡
- When **emotion** was impaired, so was **decision-making**



– The End –
Thanks for your attention

Q&A