iOS sysdiagnose analysis

Repurposing an Apple feature for forensics

CERT-EU

European Commission

How do you analyse the integrity of an iOS device WITHOUT jailbreaking it?

10 years CERT-EU

European Commission

sysdiagnose

Profiles and logs which developers use to provide bug-related information to Apple. They contain interesting information, reproducible test cases, and other useful data for investigating and diagnosing reported issues.

$whoarewe

Emilien Le Jamtel

FR

DevSecOps Head of Sector at CERT-EU

Aaron Kaplan

AT

Likes communities and genuinely cool ideas. Works with →

David Durvaux

BE

Situation Awareness Head of Sector at European Commission

Agenda

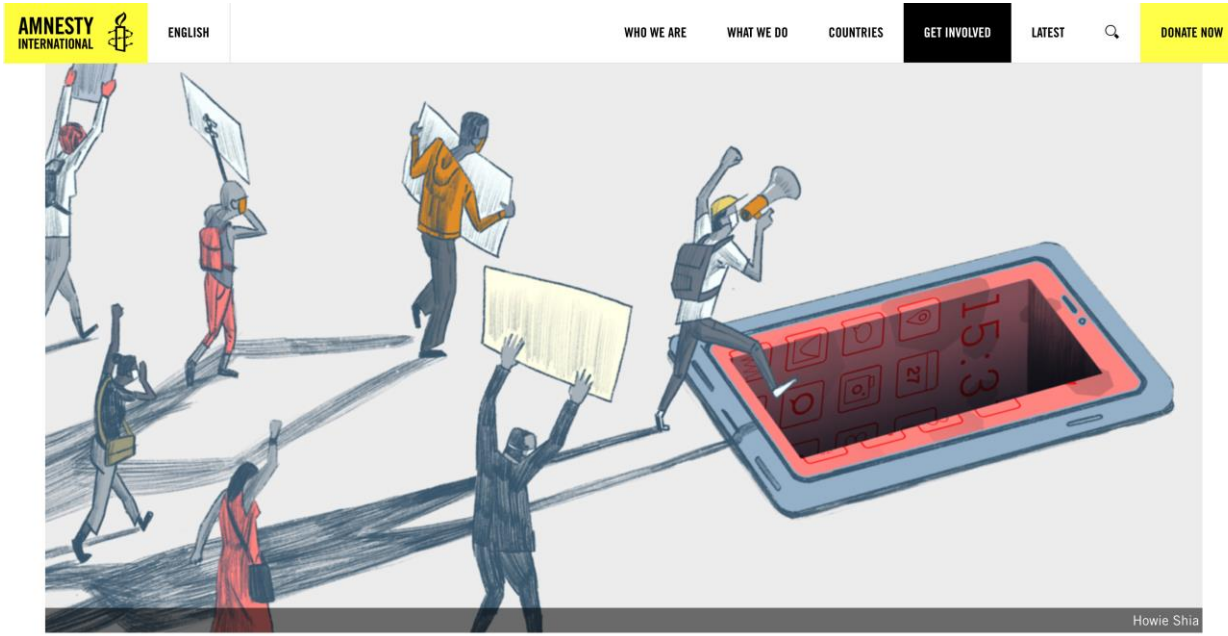
- Problem statement
- Introduction to Apple Sysdiagnose
- Our framework
- Demo
- Future

Problem statement

CERT-EU 10 years

European Commission

Why perform device analysis?



SHARE

< RESEARCH

July 18, 2021

Forensic Methodology Report: How to catch NSO Group's Pegasus

A copy of this report is available for download here.

Introduction

NSO Group claims that its Pegasus spyware is only used to "investigate terrorism and crime" and "leaves no traces whatsoever". This Forensic Methodology Report shows that neither of these statements are true. This report accompanies the release of the Pegasus Project, a collaborative investigation that involves more than 80 journalists from 17 media organizations in 10 countries coordinated by Forbidden Stories with technical support of Amnesty International's Security Lab.[1]

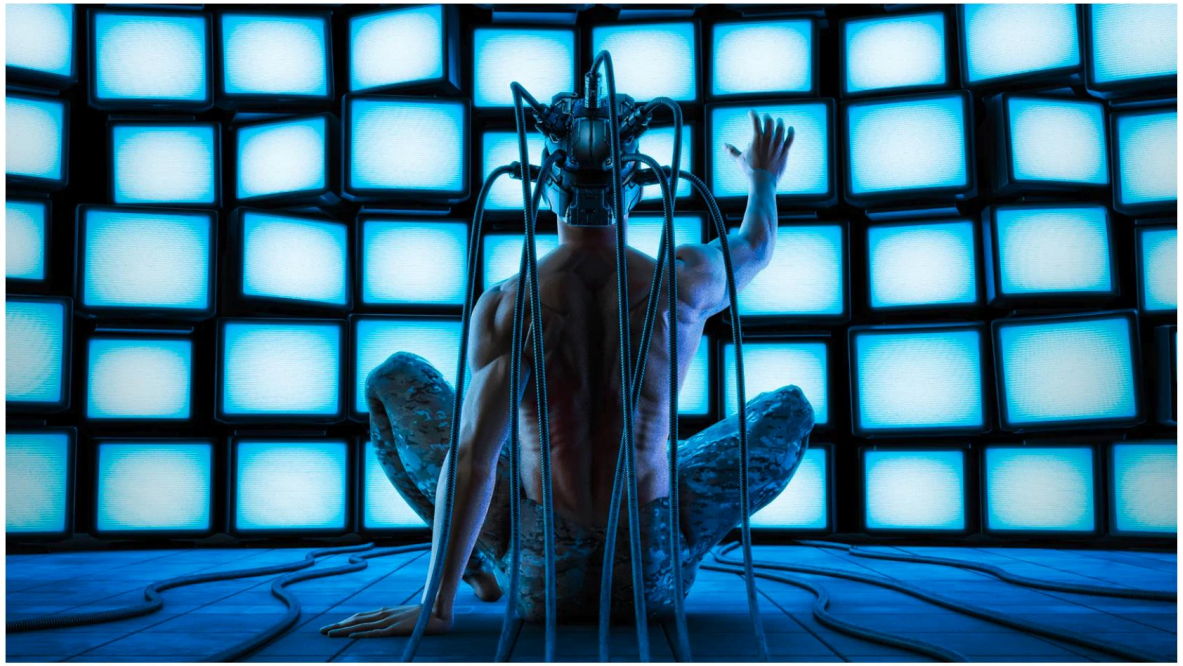
Recently added

- Myanmar: Urgent need to suspend aviation fuel as air strikes wreak havoc
- China: Heavy prison sentences for human rights activists 'disgraceful'
- Morocco: Journalist faces three years in jail for Facebook post
- Viet Nam: Independent journalist Nguyen Lan Thang facing up to 12

Google finds more Android, iOS zero-days used to install spyware

By Sergiu Gatlan

March 29, 2023 08:00 AM 0



Google's Threat Analysis Group (TAG) discovered several exploit chains using Android, iOS, and Chrome zero-day and n-day vulnerabilities to install commercial spyware and malicious apps on targets' devices.

The attackers targeted iOS and Android users with separate exploit chains as part of a first campaign spotted in November 2022.

Let's zoom into the Amnesty report

10. MOBILE DEVICES, SECURITY AND AUDITABILITY

Much of the targeting outlined in this report involves Pegasus attacks targeting iOS devices. It is important to note that this does not necessarily reflect the relative security of iOS devices compared to Android devices, or other operating systems and phone manufacturers.

In Amnesty International's experience there are significantly more forensic traces accessible to investigators on Apple iOS devices than on-stock Android devices, therefore our methodology is focused on the former. As a result, most recent cases of confirmed Pegasus infections have involved iPhones.

This and all previous investigations demonstrate how attacks against mobile devices are a significant threat to civil society globally. The difficulty to not only prevent, but posthumously detect attacks is the result of an unsustainable asymmetry between the capabilities readily available to attackers and the inadequate protections that individuals at risk enjoy.

While iOS devices provide at least some useful diagnostics, historical records are scarce and easily tampered with. Other devices provide little to no help conducting consensual forensics analysis. Although much can be done to improve the security posture of mobile devices and mitigate the risks of attacks such as those documented in this report, even more could be achieved by improving the ability for device technical experts to perform regular checks of the system's integrity.

Therefore, Amnesty International strongly encourages device vendors devices more auditable, without of course sacrificing any securi Platform developers and phone manufacturers should re tter understand the challenges faced by HRDs

- Relies on available artifacts
- Comes with a tool: Mobile Verification Toolkit (MVT)
- MVT for iOS:
  - Filesystem Dump: might have an impact on artifacts
  - iTunes Backup

Why this approach?

- Started before Pegasus
- Corporate **policies forbid us to access personal user data**
- Sysdiagnose is extensive, but relies on binaries from the device
- We want a generic approach (be IoC agnostic)

- **We consider jailbreaking as the last resort option**
  - How much can you trust a device after a jailbreak?
  - How much do you trust exploits provided by 3rd parties / a blackbox on corporate devices?

Forensically sound?

- Is **sysdiagnose forensically sound**?
  - Probably not …
- What about commercial tools?
  - They usually rely on exploits…
  - How much can you consider it to be forensically safe?
  - How much do you trust the device afterwards?

- Note: we don't take any position here!
  We are merely posing questions. It's up to you to decide - based on your needs!
  In our views, this is a complementary approach to commercial tools.

MVT vs this project

Mobile Verification Toolkit

- Supports Android & iOS
- Relies on backups for iOS
- Runs several modules to extract information
- Can ingest STIX2 IOCs to identify traces of compromise
- Has access to private user data

This project

- Only relies on Apple's sysdiagnose (gives an overview of devices' internals)
- Is IOC / detection rules agnostic
- Tries to mimic Volatility but for sysdiagnose of iOS devices
- Very easy to extend
- Consider it a **framework**

Introduction to Apple sysdiagnose

CERT-EU 10 years

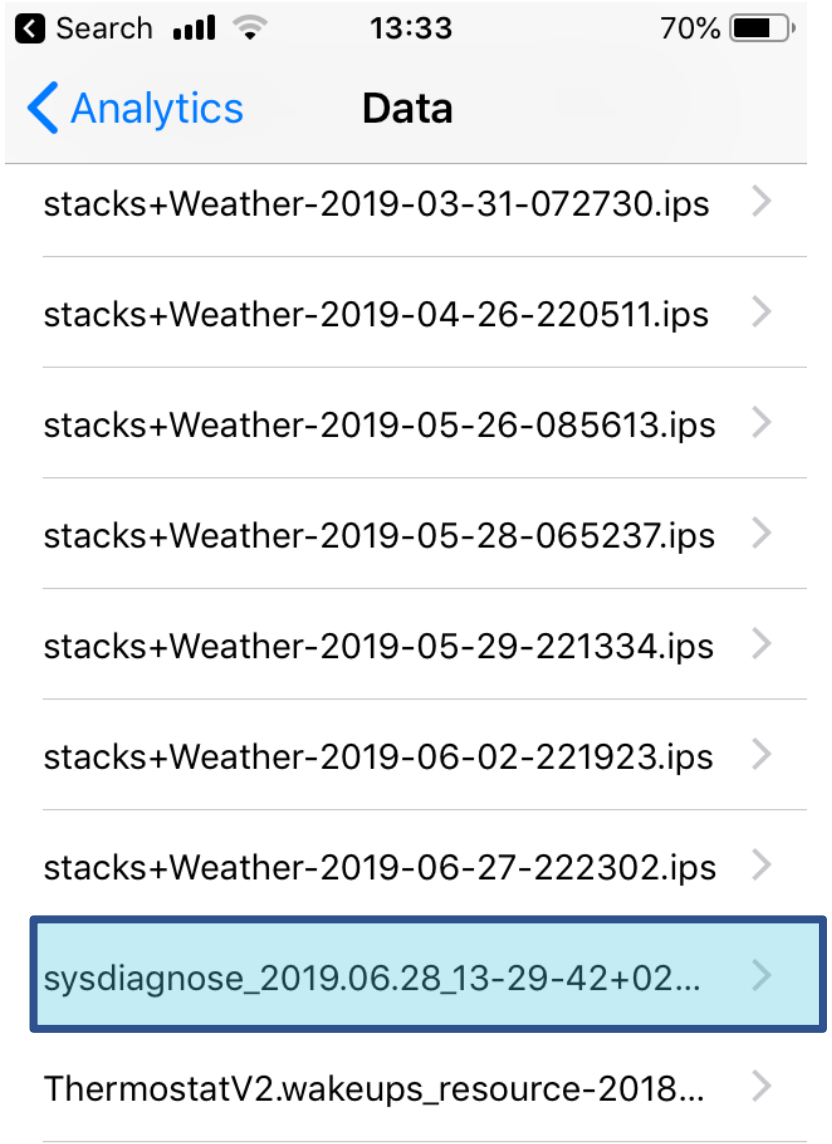
European Commission

man sysdiagnose (on macOS)

```
DESCRIPTION:
  sysdiagnose gathers system diagnostic information helpful in investigating system performance issues.
  A great deal of information is harvested, spanning system state and configuration. The data is stored /var/tmp directory.
  sysdiagnose needs to be run as root. To cancel an in-flight sysdiagnose triggered via command line interface, press Ctrl-\.
  sysdiagnose is automatically triggered when the following key chord is pressed: Control-Option-Command-Shift-Period.
WHAT sysdiagnose COLLECTS:
  - A spindump of the system
  - Several seconds of fs_usage ouput
  - Several seconds of top output
  - Data about kernel zones
  - Status of loaded kernel extensions
  - Resident memory usage of user processes
  - Recent system logs
  - A System Profiler report
  - Recent crash reports
  - Disk usage information
  - I/O Kit registry information
  - Network status
  - If a specific process is supplied as an argument, will collect:
      - A list of malloc-allocated buffers in the process's heap
      - Data about unreferenced malloc buffers in the process's memory
      - Data about the virtual memory regions allocated in the process
```

Generating sysdiagnose logs

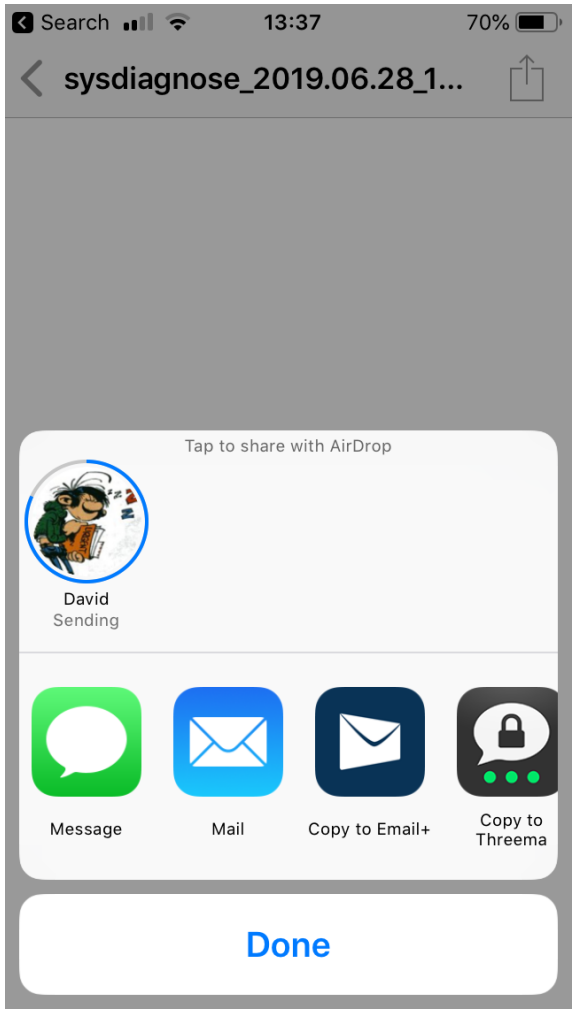
- Simultaneously **press** and **release** both volume buttons + the Side (or Top) button for 250 milliseconds.
  - Holding too long (>1s) will lock the device instead.
- Wait 10 mins
- Go to *Settings.app > Privacy > Analytics & Improvements > Analytics Data*
- Locate the sysdiagnose file *sysdiagnose_{date}_{time}.tgz*



See: https://download.developer.apple.com/iOS/iOS_Logs/sysdiagnose_Logging_Instructions.pdf

Retrieving sysdiagnose logs

- Standard Apple mechanisms:
  - *AirDrop*
  - Save to “Files” (can be iCloud)
  - …
- iTunes Sync (now: via Finder)
- libimobiledevice: *idevicecrashreport* command
- 3rd party tools
  - Magnet Forensics
  - Cellebrite
  - Elcomsoft iOS forensic toolkit
  - …



Sysdiagnose content

- Results of commands run on the device to create a status overview: running processes, mounted partitions…
- Copy of key preferences files (plist)
- Network configuration & history
- Information on hardware health
- Log files
- Device diagnostic
- Usage overview
- …

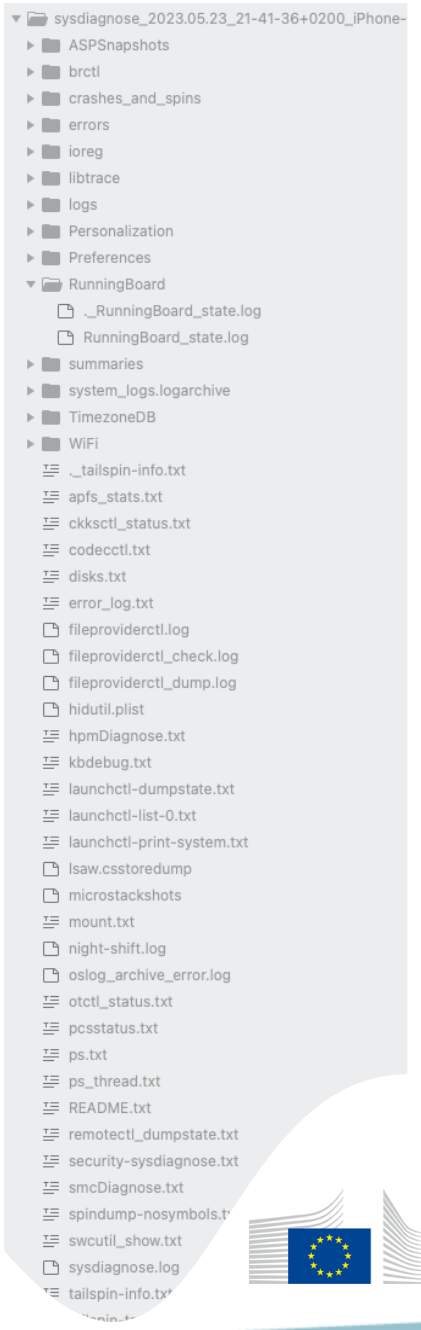
Contents of sysdiagnose dumps

- Results are stored in many different formats
  - ASCII text
  - CSV text
  - GZIP files
  - SQLite
  - Unicode
  - Plist (text and binary)

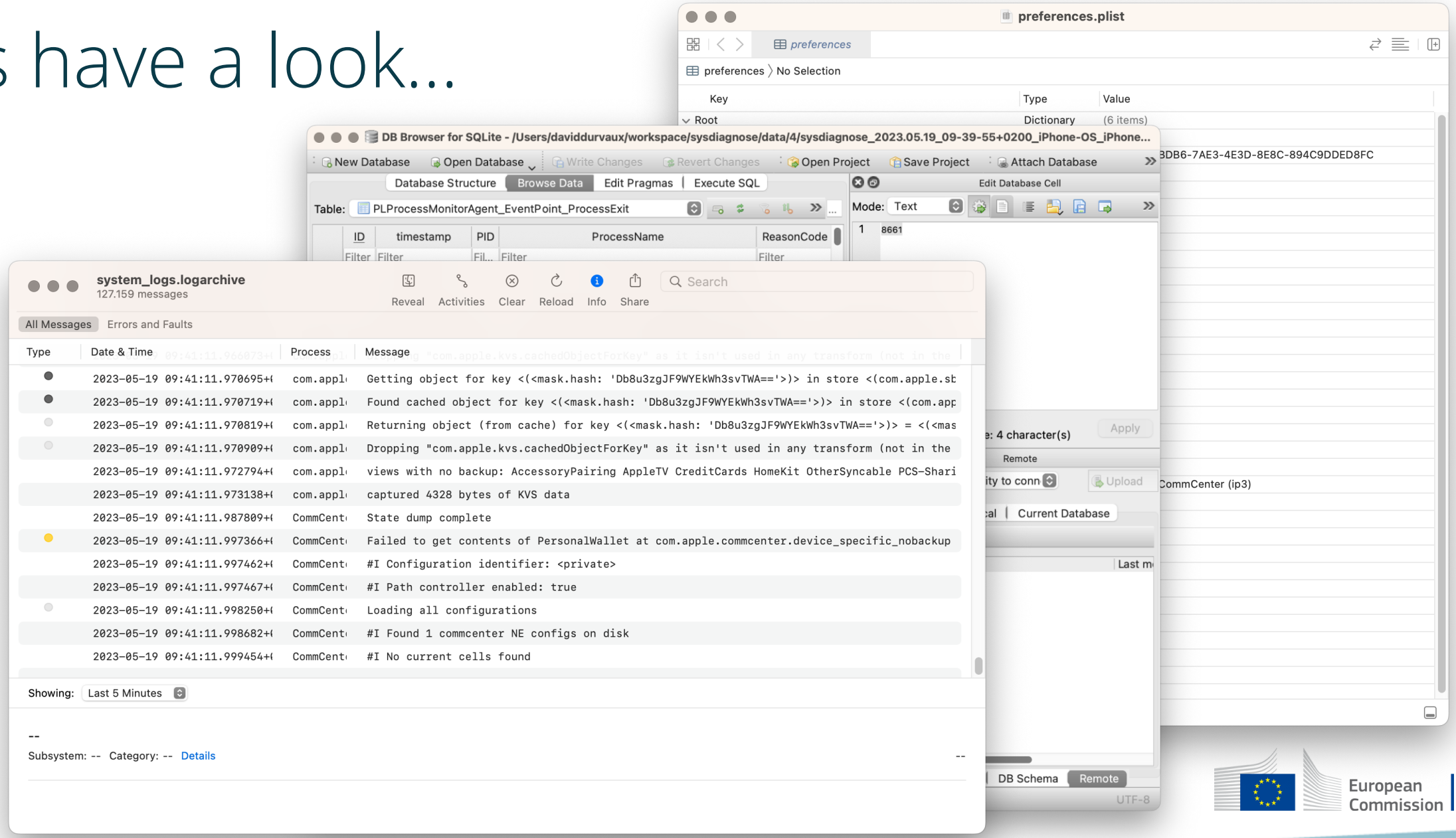
- Timestamps aren't uniform
  - Mac Epoc
  - Unix Epoc
  - …

Sysdiagnose structure

- Results of commands
- ./logs : device logs including Power Logs
- ./Preferences: device preferences
- ./summaries: extract from Power Logs
- ./system_logs.logarchive: system logs
- ./WiFi: Network and Bluetooth informations
- many other info :)



Let's have a look...



Challenges with manual analysis

- Most files can be analysed with standard tools to read CSV, Plist, SQLite files…
- But information is spread across many different files
  - Analysts need to know all and not forget one
- The structure is relatively self-explanatory
  - Analysts need to be familiar with iOS artifacts
- Manual analysis is a tedious process

Our framework, Architecture

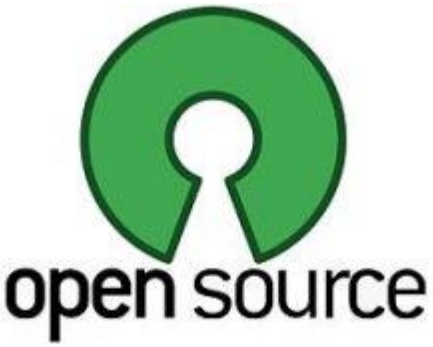
CERT-EU 10 years

European Commission

Our framework

- FOSS-licenced, under the European Union Public License (EUPL)
  https://github.com/EC-DIGIT-CSIRC/sysdiagnose

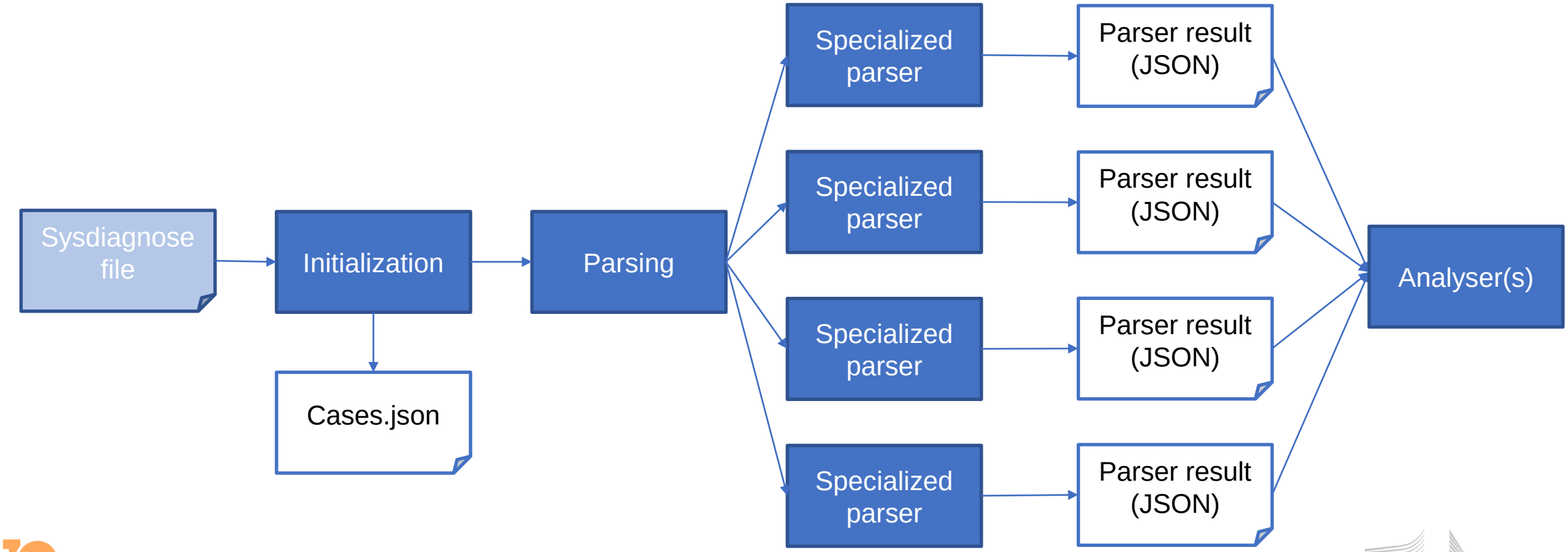
- Feel free to
  - **Use** it
  - **Extend** it
  - **Propose** changes
  - …



Our philosophy

- Keep every code block as simple as possible (KISS)
- Every analyser and parser can run independently from the others
  - Can be used as standalone tools
  - Some offer goodies
- A parser takes one artifact from the sysdiagnose directory and produces JSON output
  - A parser is not trying to provide any analysis at this stage
- An analyser relies on the JSON provided by parsers to create a relevant output
  - Analysers are independent of the sysdiagnose dump structure
  - Typical analysers: export timestamps and build a timeline

Processing workflow



Current Developments– iOS16

Type of module	Count
Parsers	**25**
Analysers	**4**
• Timeliner (→ goes to timesketch)	
• Wifi Geolocation KML	
• Wifi Geolocation GPX	
• Application UUID	

Usage

1. initialyze.py
   - **Extracts** the archive and produces a per case JSON
   - Feeds the information into **`cases.json`**
2. parsing.py
   - **Calls** selected (or **all**) parsers
   - Parsers' results are stored in **./parsed_data/*** JSON files
3. analyse.py
   - **Calls** selected (or **all**) analysers
   - Uses the results of parsers to produce an analysis output

Demo

Extracts the sysdiagnose archive locally, updates cases.json and db.json

Parses artifacts and produces a JSON view for later use

Ingests JSON and produces an analysis report

```
❯ python3 initialyze.py file ../sysdiagnose-data/sysdiagnose-files/iOS16/sysdiag
nose_2023.05.15--ROGUE-WIFI.tar.gz
89b21be917af40c6df6670283a501bc33978f21056a54fbfbf80d83112e6e42d
```

```
❯ python3 parsing.py parse sysdiagnose-wifisecurity 3
```

```
❯ python3 analyse.py analyse sysdiagnose-timeliner 3
```

How to make your own parser?

Parsers required variables:

```
# ----- definition for parsing.py script -----#

parser_description = "Parsing WiFi Security logs"
parser_input = "wifisecurity"
parser_call = "get_wifi_security_log"
```

→ Free text description

→ File to parse as defined into case JSON

→ Function to call and that returns a JSON

- **`parser_input`** corresponds to an entry into `./data/<case id>.json`
  - If required file not defined there, can be added into parsing.py
- Function defined in **`parser_call`** is expected to:
  - Be given a path to a file to parse as 1st argument
  - iOS version as an int (optional 2nd argument)
  - Returns a valid JSON object

How to make your own analyser?

Analysers required variables:

```
# ----- definition for analyse.py script -----#
# -----          DO NET DELETE             ----#

analyser_description = "Generate a Timesketch compatible timeline"
analyser_call = "generate_timeline"
analyser_format = "jsonl"
```

→ Free text description

→ Function to call to generate content

→ Output format

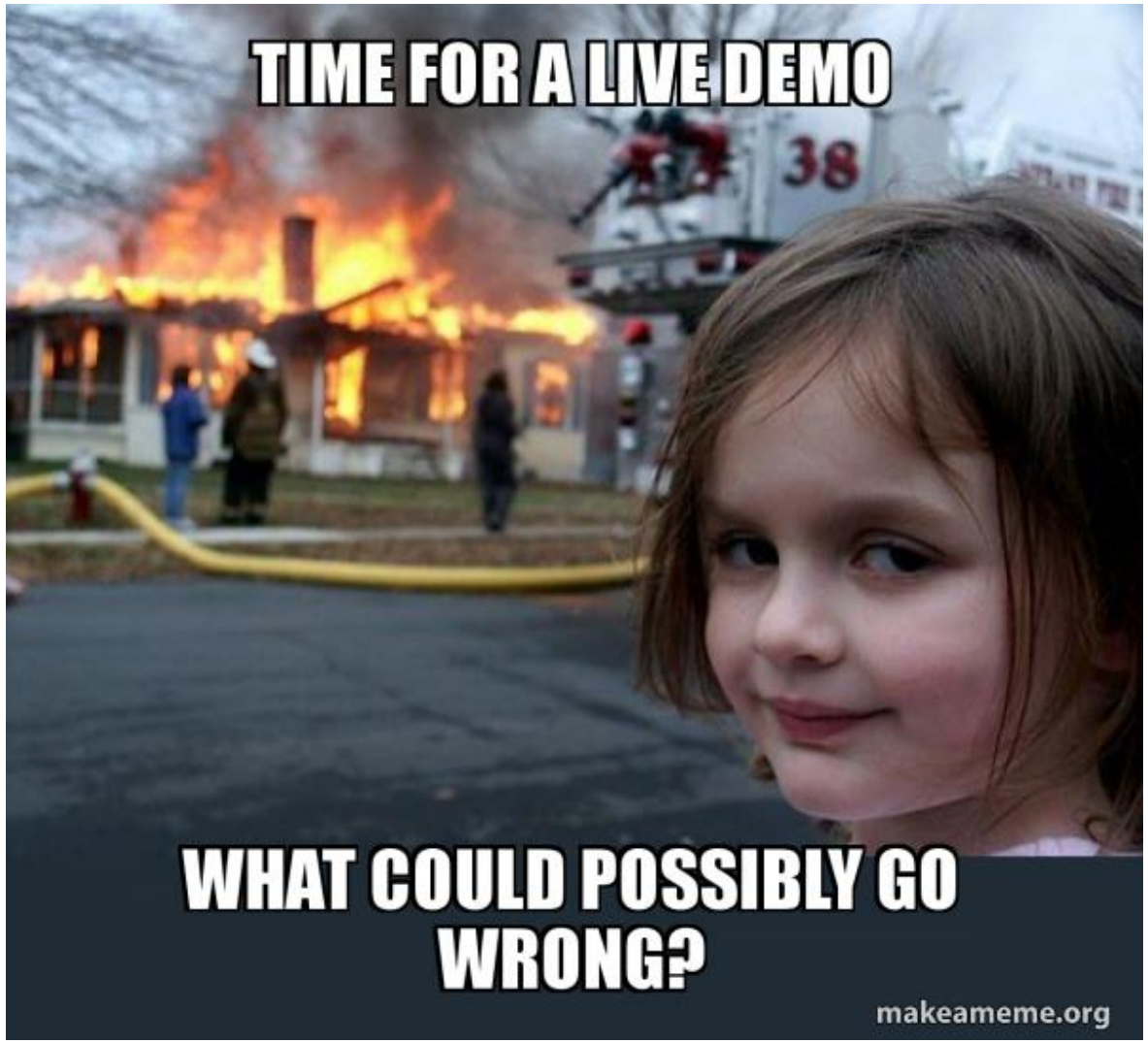
- The function defined in the var `**`analyser_call`**` expects:
  - a path directory with JSON generated by the parsers (argument #1)
  - a path to a file to save result (argument #2)
- Outputs format depends on analyser goals

Demo

10 years CERT-EU

European Commission

Demo

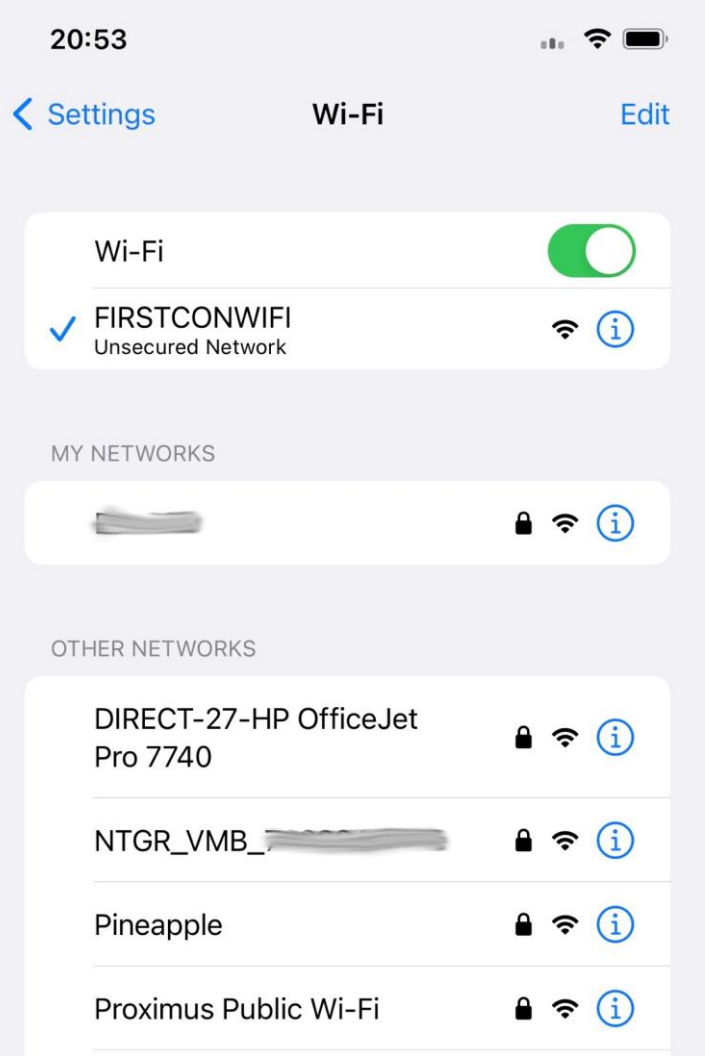

TIME FOR A LIVE DEMO
38
WHAT COULD POSSIBLY GO WRONG?
makeameme.org

Demo: rogue Wi-Fi (easy)

wifi_status.txt

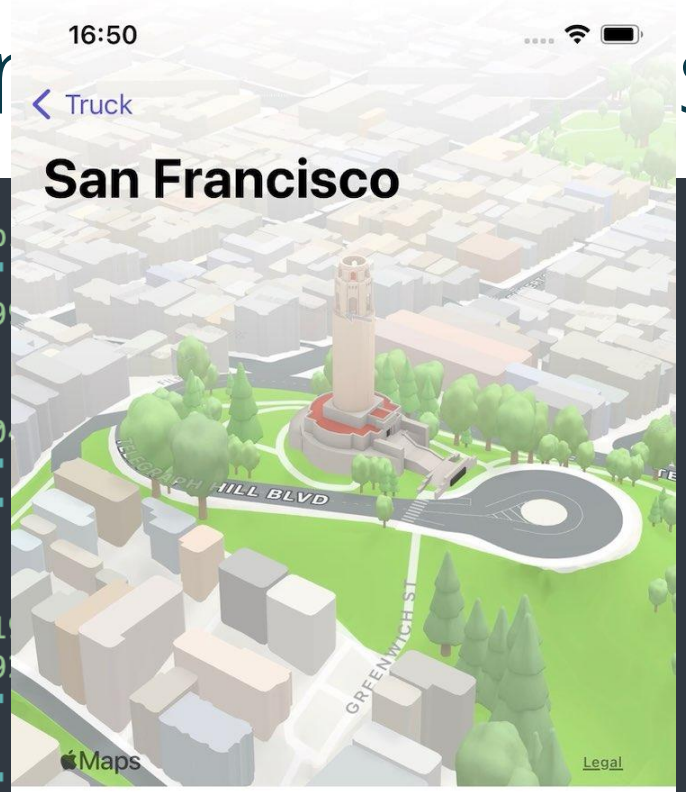
```
# --- Wi-Fi Status

    MAC Address          : b2:74:30:d0:88:9c (hw=6c:e8:5c:43:e0:2f)
    Interface Name       : en0
    Power                : On [On]
    Op Mode              : STA
    SSID                 : FIRSTCONWIFI
    BSSID                : 00:13:37:a9:e7:eb
    RSSI                 : -42 dBm
    Noise                : -94 dBm
    Tx Rate              : 72.0 Mbps
    Security             : None
    PHY Mode             : 11n
    MCS Index            : 7
    Guard Interval       : 800
    NSS                  : 1
    Channel              : 2g11 (20 MHz, Active)
    Country Code         : BE
```



Der st on


```
"536": {
    "USER": "mob
    "UID": "501"
    "PRSNA": "19
    "PID": 536,
    "PPID": 1,
    "F": "400400
    "CPU": "0.0"
    "MEM": "0.5"
    "PRI": "4",
    "NI": "0",
    "VSZ": "4081
    "RSS": "1859
    "WCHAN": "-"
    "TT": "??",
    "STAT": "Ss"
    "STARTED": "
    "TIME": "0:0
    "COMMAND": "
```

Recommended
Coit Tower
Parking Spot
22°C Popular Trending

Cloud cover percentage is currently 15% in San Francisco

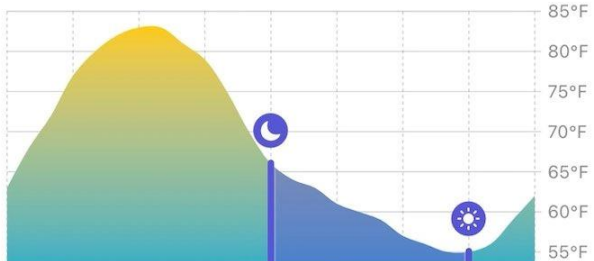
Popular donuts this season include Custard, Super Lemon, and Rainbow

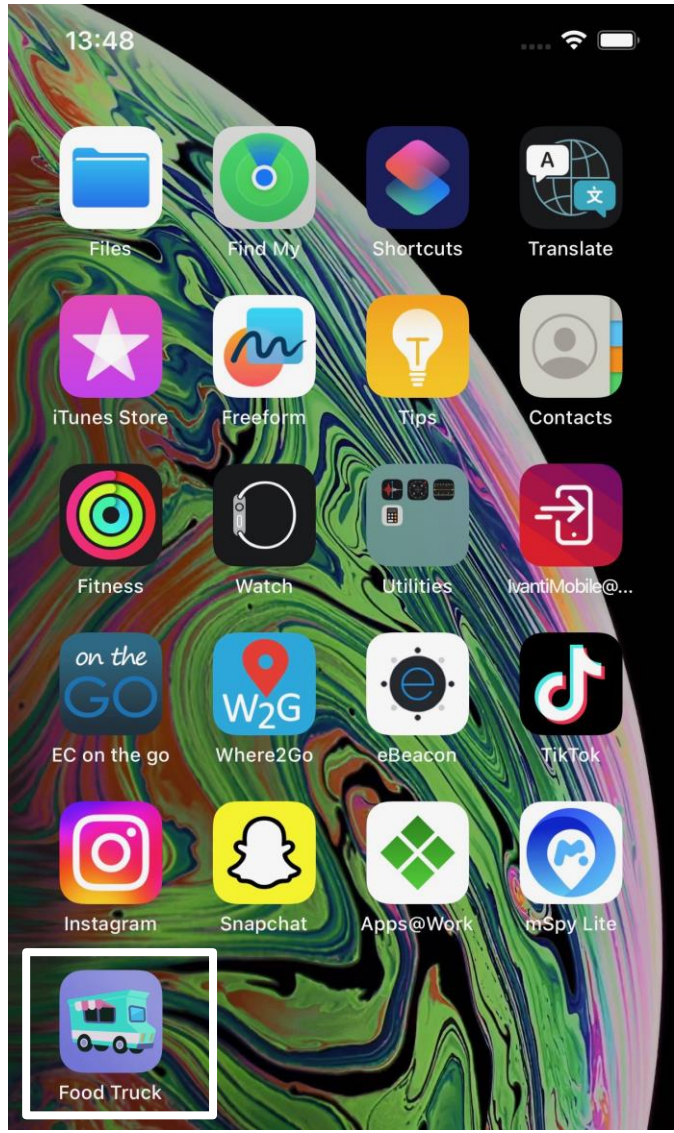
Recommendation to stock up on cold ingredients and popular toppings to be

New Orders

Order#1244 5

Forecast



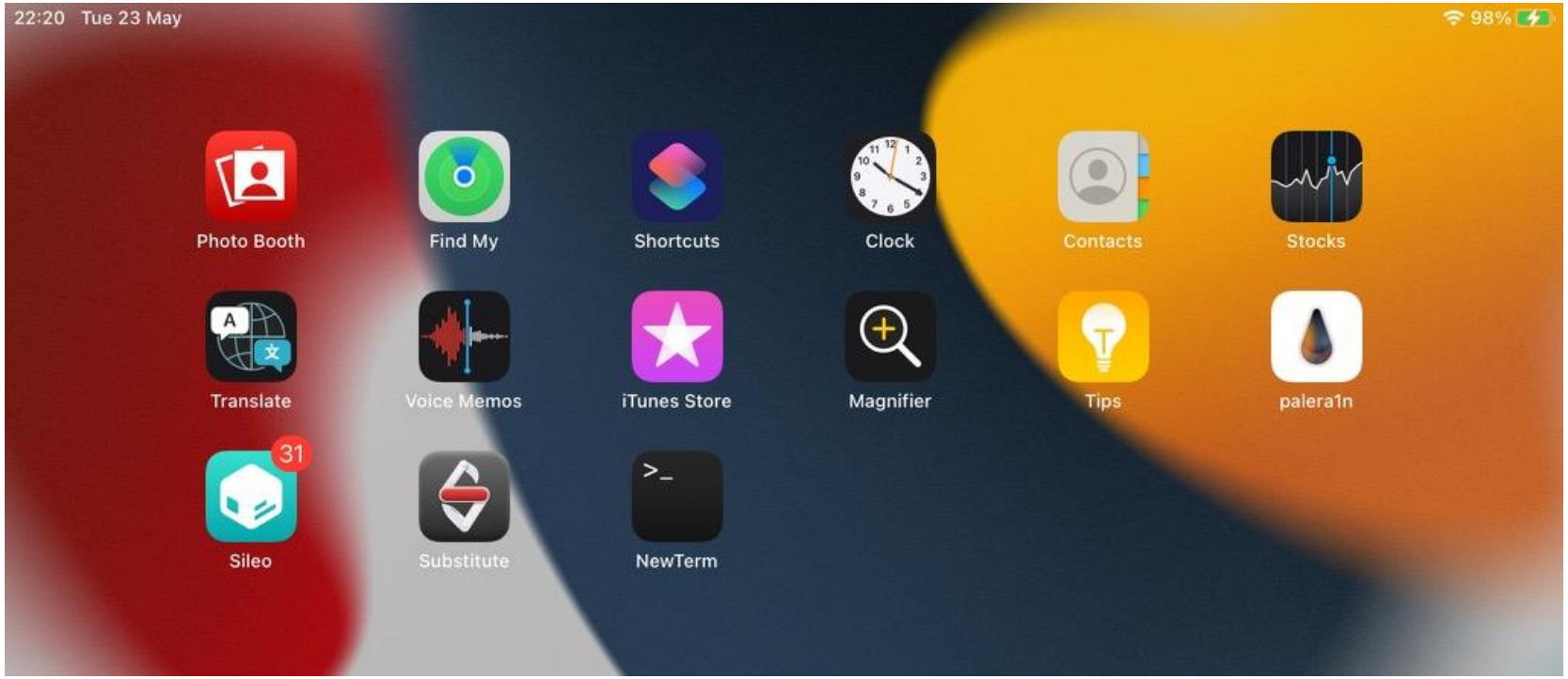


CERT-EU

#FIRST

RST CONFERENCE | EMPOWERING COMMUNITIES

Demo: strange processes




```
root             0  1000  1461  1460  4004106   0.0  0.1  31  0 407927312   1424 -      s000  Ss    9:35PM   0:00.02 login -fp mobile
mobile         501  1000  1462  1461  4004006   0.0  0.3  31  0 407932896   6352 -      s000  S     9:35PM   0:00.20 -zsh
root             0  1000  1497  1462  4004106   0.0  0.2  31  0 407932176   3216 -      s000  S     9:37PM   0:00.05 sudo su
root             0  1000  1498  1497  4004006   0.0  0.1  31  0 407919136   1328 -      s000  S     9:37PM   0:00.02 su
root             0  1000  1499  1498  4004006   0.0  0.1  31  0 407928976   2720 -      s000  S     9:37PM   0:00.06 zsh
root             0  1000  1504  1499  4004006   0.0  0.2  31  0 407931328   4576 -      s000  S+    9:40PM   0:00.05 ssh -R localhost:7070:localhost:22 david@
```

Challenges

- Artifacts can change a lot with each (major) release of iOS
  - Formats change
  - Log file contents may be completely different
  - Can disappear « randomly »
  - …
- Log formats are not properly documented by Apple
- Many differents formats and data encoding types
- Need to differentiate relevant vs non relevant data

Limitations

- Sysdiagnose is for **diagnostic** purpose
  - Doesn't contain user data
- For example, the Kaspersky Operation Triangulation detection tool relies on artifacts that are **only in a full device backup**
  - Check modification to SMS attachment database and its properties
  - Check preferences that are not copied into a sysdiagnose file
  - Detection via sysdiagnose is unknown
- You need to be aware of the difference between a sysdiagnose and a full backup (via respective pros and cons)

Future

10 years CERT-EU

European Commission

Future

We plan to…

- **Extend** the coverage and support future iOS versions
- Add support for more Apple devices (watchOS, tvOS, macOS…)
- Bring more **analysers** to support analysts
- Validate the **effectiveness** of this approach on as many use-cases as possible (and share the results back with this community)

- **Remember**: this is an **open source framework**. It's here for you, your use-cases… feel free to adapt, rip, copy & mix

Validating effectiveness

- We are searching for **sysdiagnose of compromised devices**
  - Any version of iOS
  - Can be shared under [TLP:RED]

Goals?

- Confirm effectiveness of this framework with more samples
- Identify gaps / issues / improvements
- Make this tool more useful for our community

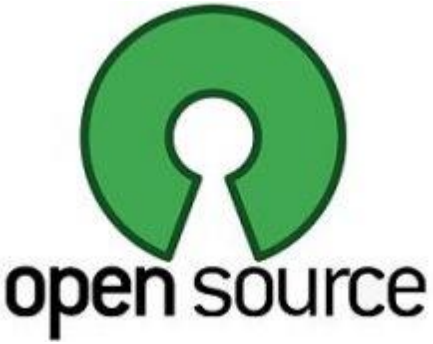
References & acknowledgments

- **Sarah Edwards** for the discussion that triggered this
- **Mattia Epifani**, **Heather Mahalik** and **Cheeky4n6monkey** for the **iOS_sysdiagnose_forensic_scripts** (GitHub)
- Johan Berggren (TimeSketch, google)
- Amnesty International Pegasus Report

One last word…

- Again, this is a free, open source project https://github.com/EC-DIGIT-CSIRC/sysdiagnose
  - Using the European Union Public License (EUPL)

- Feel then free to
  - **Use** it
  - **Extend** it
  - **Propose** changes
  - …



Thank you