Build Automated Malware Lab with CERT.PL Open Source Tools

Paweł Srokosz
Paweł Pawliński

CERT.PL

34th Annual FIRST Conference
27th June 2022, Dublin
Automated malware lab - why?
CERT.PL: who are we

- Established in 1996
- National CERT role formalized in the cybersecurity law in 2018
- Constituency: everything in Poland (*)
  (*): except government, military, critical infrastructure
- Part of NASK (research institute & .pl registry)
We are in threat intelligence business

- Monitoring threats to millions of users
- Malware incidents: 2nd most common (after phishing)
- We want to:
  - detect malware campaigns
  - warn potential victims
  - mitigate
    as early as possible
Evolution of our malware tooling

- Initially: tools developed case-by-case
- Early 2010s: rise of the banking trojans
- Mid 2010s: first automated malware analysis pipeline
- Late 2010s: live tracking of multiple botnets
- 2020s: era of open source analysis tools
Basic ingredients of malware analysis lab

- **Collect**: repository to collect and search samples, IoCs, etc. from various sources (internal and external)
- **Analyze**: framework to integrate analytical tools focused on specific threats
- **Share**: provide threat intelligence to constituents / peers / customers
Main components of our lab

<table>
<thead>
<tr>
<th>Repository</th>
<th>Stars</th>
<th>Forks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CERT-Polska/mwdb-core</td>
<td>139</td>
<td>36</td>
</tr>
<tr>
<td>CERT-Polska/karton</td>
<td>159</td>
<td>10</td>
</tr>
<tr>
<td>CERT-Polska/drakvuf-sandbox</td>
<td>420</td>
<td>65</td>
</tr>
<tr>
<td>CERT-Polska/mquery</td>
<td>273</td>
<td>52</td>
</tr>
<tr>
<td>CERT-Polska/malduck</td>
<td>124</td>
<td>9</td>
</tr>
</tbody>
</table>
Collect: MWDB Core
What is MWDB Core?

- Central component of our lab
- Repository for organizing and sharing malware intelligence
- Open-source
- Easy integration with other tools:
  - plugins
  - Karton
- Supported by CERT.PL and (small) community
MWDB Data model

- MWDB is made by analysts for analysts
- Not really a general purpose threat information sharing system
- Three basic object types:
  - Files
  - Configurations
  - Blobs
- Structured metadata for all objects
MWDB: Files

- The most basic object type
- Tags: file type, source, classification, …
- Attributes: source URL, Yara matches, AV detection, …
MWDB: Configurations

- Embedded in binary (static)
- Downloaded from C2 (dynamic)
- JSON
- Well-defined keys per malware family
- Structure determined by internal configuration format

**End-goal of a typical malware analysis task** (automated by us for families of interest)
Basic processing pipeline
MWDB: Blobs

- Unstructured
- Decrypted data, webinjects, commands, lists of peers, ...
- Stored for later processing or human inspection
- Full-text search
Pipeline for botnet monitoring

- Static malware configuration
- Dynamic malware configuration
- Unparsed static configuration parts
- Next stage malware
- Emulated bot
- Sandbox extractor
- Commands
- Webinfects
Real-life example: ISFB (Gozi) graph
Metadata: tags

Name: jew.mpsl
SHA256: 1d2e11bc0...ed53c5a78b3d
MD5: 19830e713...e61990b4dc42
Size: 94.21 kB
Type: ELF 32-bit LSB executable

feed:urihaus ○ miral ○
ripped:urihaus ○
runnable:linux ○
urihaus:elf ○
urihaus:miral ○

Sun, 11 Apr 2021
14:44:04 GMT
## Metadata: attributes

<table>
<thead>
<tr>
<th>File type</th>
<th>Zip archive data, at least v1.0 to extract, compression method=store</th>
</tr>
</thead>
<tbody>
<tr>
<td>md5</td>
<td>8e3280ebebc73e0863ac5b68854c5eb50</td>
</tr>
<tr>
<td>sha1</td>
<td>d645b41f3dfe38101177f449aa9b10ed53f399bb6b</td>
</tr>
<tr>
<td>sha256</td>
<td>d119aa91abdb0605e30b52602e2bb2ae0e40e5ae2255f7f1832f7531ae9c737</td>
</tr>
<tr>
<td>sha512</td>
<td>6b46c5fab22b87a7a8afdd7476c17b66d0cdef9547359e0409eb92bd9f8f5c02bca1ed52163474af421deba7e21f2904d715b4a8424eaa3c3c8a76e131508</td>
</tr>
<tr>
<td>crc32</td>
<td>5b882522d</td>
</tr>
<tr>
<td>ssdeep</td>
<td>2417KEG6sd655q2yUCv0Lmo0zELmyvTQB6QRQDUYSrITzAdIiCE6S8Fq2yjVylqTCGRwuYFItz4I</td>
</tr>
<tr>
<td>Upload time</td>
<td>Tue, 14 Jun 2022 18:54:20 GMT</td>
</tr>
</tbody>
</table>

### Attributes

- **From**
  - https://drive.google.com/uc?
    - export=download&id=16xAMiHiFlgYcKpmjZWb8RQuYXH8Ffx8y&confirm=t
  - https://drive.google.com/uc?
    - export=download&id=13HillaEZCE_51syJNe4aEP8XQ9mjnWyrl&confirm=t

- **Archive password**
  - E58346

- **Incident ID**
  - 1700028
Analyze: Karton
Pareto rule

- 20% efforts, 80% effect
  writing an actual script to process a malware feed

- 80% efforts, 20% effect
  polling for data, queueing, integration with other scripts, logging, proper error handling, maintenance…
Pareto rule

- 20% efforts, 80% effect

writing an actual script to process a malware feed

- 80% efforts, 20% effect

(handle all of the common things with some common approach)
Karton design

- Queue-based data processing pipelines
- Data-driven routing of tasks
- Lightweight
- Based on Redis (KV store) and S3-compatible object stores
- Built for microservices:
  - each processing module is focused on one task
  - “Plug and Play”, researcher should be able to easily add a new service
- Management interface

Inspiration: Assembly Line by Canadian Centre for Cyber Security
Example: consumers of Office documents
queue karton.yaramatcher

Description
Scan samples and analysis results and tag malware samples using matched yara rules.

Filters
- kind:runnable
- stage:recognized
- type:sample
- kind:dump
- stage:recognized
- type:sample
- kind:cuckoo1
- type:analysis
- kind:drakrun
- type:analysis
- kind:joesandbox
- type:analysis

Karton-core library version
4.3.0

Service version
1.1.1

Queue persistence
yes

Spawned tasks
0

Crashed tasks
1

Replicas online
1

Crashed tasks

<table>
<thead>
<tr>
<th>task</th>
<th>headers</th>
<th>exception</th>
<th>actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>cf5e6599-e4be-417e-9aaf-</td>
<td>low</td>
<td>kind:drakrun</td>
<td>minio.error.S3Error: S3 operation failed; code: IncompleteBody, message: You did not provide the number of bytes specified by the Content-Length</td>
</tr>
</tbody>
</table>
Providing threat intelligence

- Making our know-how & data available for defenders
- Access to our MWDB instance
  - samples
  - configurations
  - output of our private analyzers
- Free service: https://mwdb.cert.pl/
- Open registration + manual vetting
Statistics

- 1000+ accounts
- Extractors for 133 families (*)
  
  (*) not all work with current variants
- 2.4M+ samples
- 67k+ configurations
- 700/day avg new samples
Working with the community

MalwareBazaar now integrates results from @CERT_Polska_en Malware Database (MWDB) 🥳 🎉 🎊

Sample report:
[bazaar.abuse.ch/sample/2629fbf...](bazaar.abuse.ch/sample/2629fbf...)

1:57 PM · Jun 30, 2020 · Twitter Web App

Replied to @nazywam

Up next: German banking (lots of https://*bank*.de)

sample: bazaar.abuse.ch/sample/01d5f1b...

c2: ylnfkeznzg7o4xfj[.]onion/kpanel/connect.php

mwdb: [mwdb.cert.pl/blob/d730eefc...](mwdb.cert.pl/blob/d730eefc...)

12:59 PM · Feb 12, 2021 · Twitter Web App
Plugin showcase: malware similarity
Finding similar samples

- Objectives:
  - classify malware family
  - discover clusters
- Can be used to detect new variants
- No reversing & development of analysis modules necessary
- Better understanding of the development of threats
- Common use case: support attribution
Using Windows API for classification

Drakvuf Sandbox: dynamic analysis and gathering memory dumps

ApiScout: finding informative Windows API calls

ApiVectors

Dump classification: nearest cluster

Detected families (labels)

Sample classification: aggregation of labels

Final malware family

Tool by Daniel Plohmann
http://byte-atlas.blogspot.com/2017/04/apiscout.html
## Classification results

<table>
<thead>
<tr>
<th>File details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details</td>
</tr>
</tbody>
</table>

### Summary
- **lokibot**: 100.00%
- **trickbot**: 9.58%
- **remcos**: 6.39%
- **azorult**: 2.45%

#### 400000_a0a37f39a93379fa
- **Families**: lokibot
- **Similarity**: 100.00%
- **Packed avivector**: A130GA6CA10MA4gAAQAAIAUgAAQA5

#### 76450000_76b2201913d40b4e
- **Families**: azorult
- **Similarity**: 1.04%
- **Packed avivector**: A20IA21QA31A5CA17QA11CA33C8CBA7gA9EA4EA28
Upcoming integration: msource

- Finding similar code in malware binaries
- Function-level comparison
- Flexible backend: currently multiple disassemblers
- Internal web interface for analysts and administrators
- PoC plugin for MWDB in 2021, improved version coming soon
msource: behind the scenes
How to get started
MWDB Core: official docs

https://mwdb.readthedocs.io/
Online training materials

https://training-mwdb.readthedocs.io/
mwdblib: automation library for MWDB

https://github.com/CERT-Polska/mwdblib
malduck: supports malware analysis

- Open-source configuration extractor engine, written in Python
- Collection of common algorithms and utilities for extracting data from binaries

[github.com/CERT-Polska/malduck]

[README.md]

<Malduck>

Installation ⚙ Docs ▶

CERT.PL
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 830892.

The contents of this presentation are the sole responsibility of CERT.PL / NASK and do not necessarily reflect the opinion of the European Union.

Contact:

pawel.srokosz@cert.pl
pawel.pawlinski@cert.pl
info@cert.pl

https://github.com/CERT-Polska/