Build Your Own Malware Analysis Pipeline Using ~~New~~ Open Source Tools

Jarosław Jedynak

Paweł Srokosz

CERT.PL >_

Annual FIRST Conference
Fukuoka, 8th June 2024

Agenda

- **mwdb.cert.pl**
  - What the heck is **MWDB?**
  - Tour de **mwdb.cert.pl**
  - Scripting and automation with **mwdblib**
- **karton and malduck**
  - Run a **self-hosted** mwdb-core and karton instances
  - Experiment with **karton-playground**
  - Automated unpacking with **malduck**
- **mquery 101**

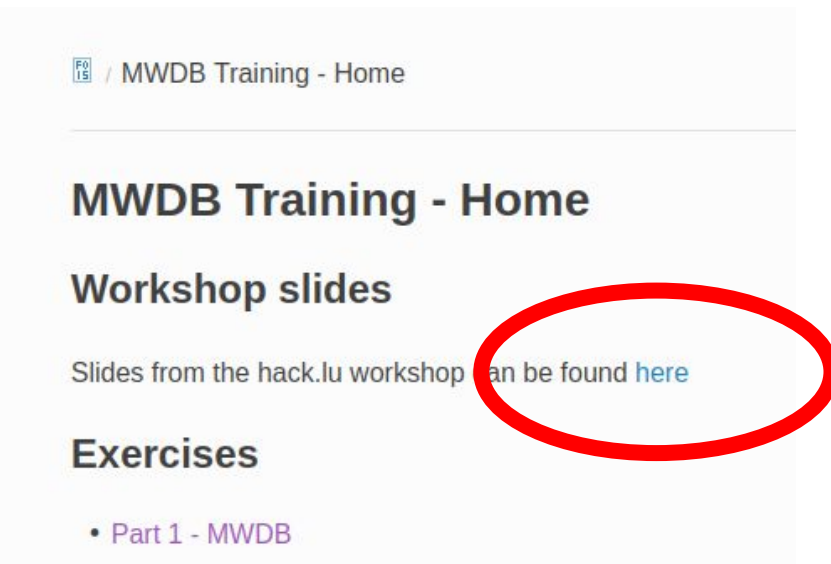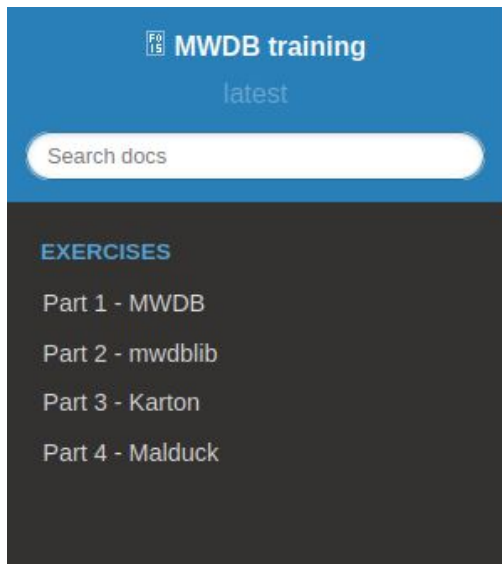CERT.PL >_

Prerequisites

A few things before we get started

Training materials (very important)

https://training-mwdb.readthedocs.io/

Training materials (very important)

https://training-mwdb.readthedocs.io/
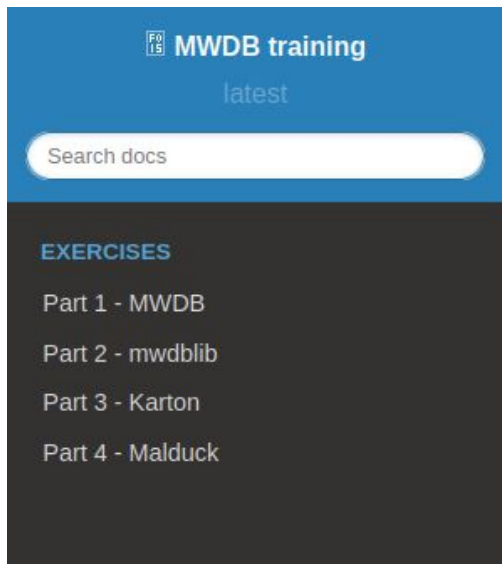
Prerequisites

Open a terminal and check if these tools are installed:

- `$ python3 -m pip`
- `$ git`
- `$ docker-compose`
  https://docs.docker.com/engine/install/ubuntu/
  https://docs.docker.com/compose/install/
- **Something like this may work:**
  `apt install python3 python3-pip git docker.io docker-compose`
  If you get Python errors from docker-compose check this link.
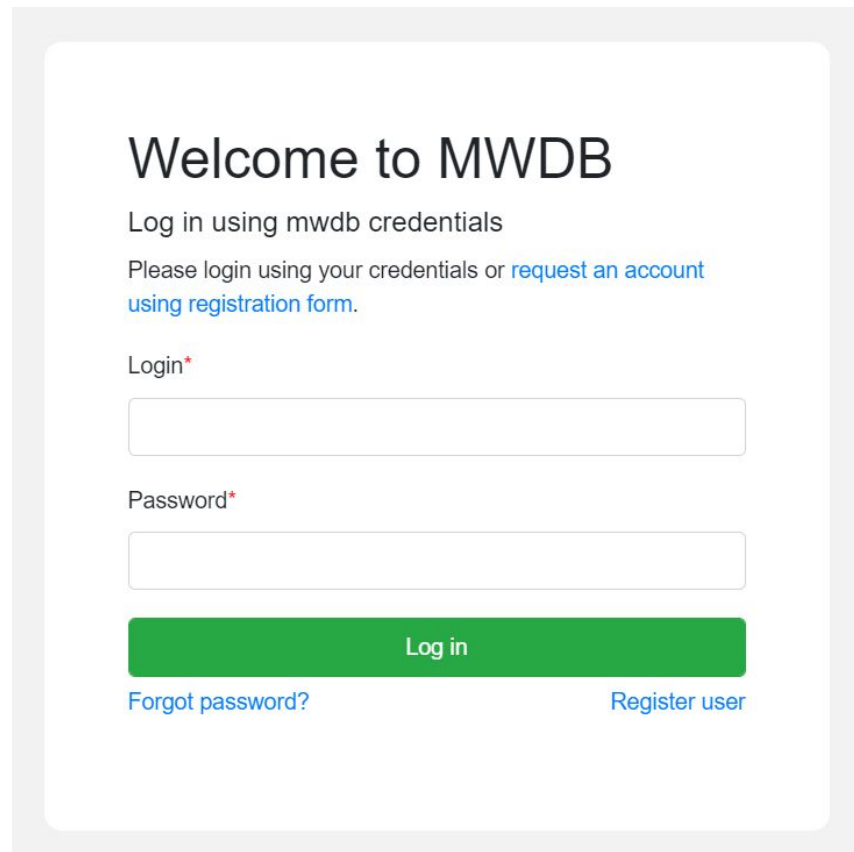
<ERT.PL >_

MWDB account

You can use temporary account made for a workshop.

Username: **first-training-2024**

Password: **first-training-2024**

The training account will be disabled after workshop, so consider registering right now to keep access.

Welcome to MWDB

Log in using mwdb credentials

Please login using your credentials or request an account using registration form.

Login*

Password*

Log in

Forgot password?   Register user

CERT.PL >_

MWDB account

Fill in the registration form at:

https://mwdb.cert.pl/register

Put "**FIRST 2024**" in additional info.

You will get confirmation email once the account is validated.

Register user

Provided data are needed for vetting process. Keep in mind that all submissions are reviewed manually, so the approval process can take a few days. Before filling this form, make sure you have read our Terms of use.

Login

Login must contain only letters, digits, '_' and '-' characters, max 32 characters allowed.

Business e-mail

Additional information:
Affiliation

Provide name of company or university

Job Title

Provide your job title

Job Responsibilities

Provide your job responsibilities and experience in the field of malware analysis

Other information

Botconf 2022

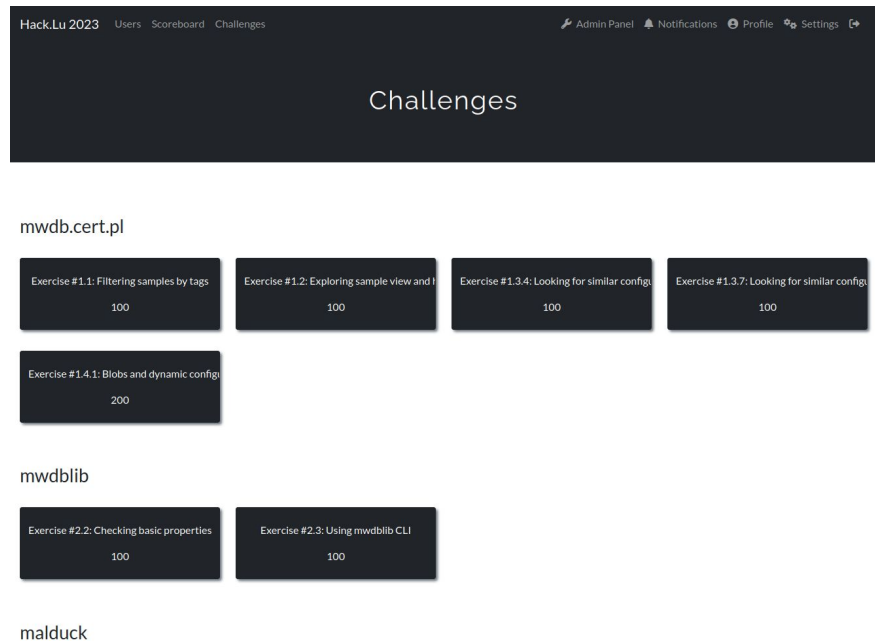Provide additional information e.g. Twitter handle, invitation info, blog URL etc.

CERT.PL >_

CTFd (optional but recommended)

Visit

http://submit.mwdb.net

Register an account
(**fake email OK**, it's not verified).
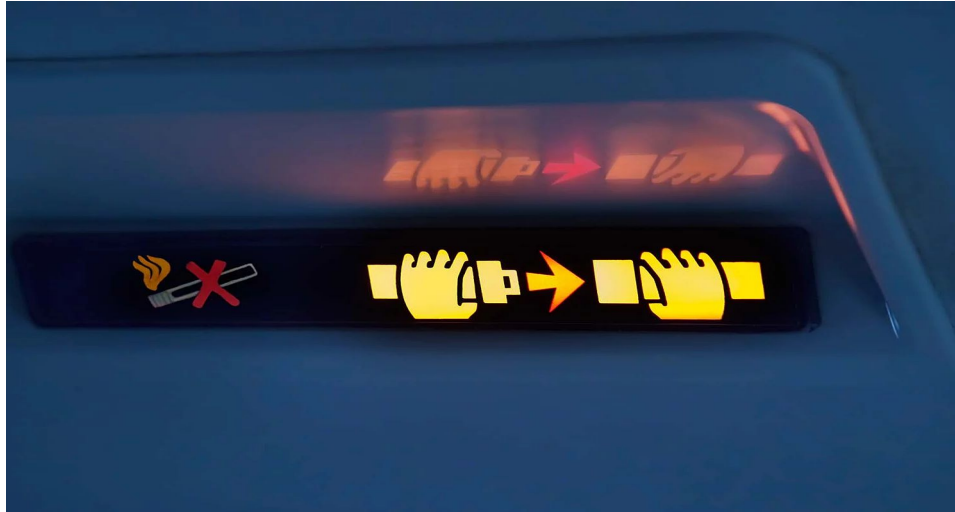
It'll be useful in a few minutes to submit your answers.

CERT.PL >_

mwdb.cert.pl

#1. Tour de **mwdb.cert.pl**

https://mwdb.cert.pl

CERT.PL

mwdb.cert.pl

Let's start!
Login to MWDB

Training materials: https://training-mwdb.readthedocs.io/

mwdb.cert.pl

Exercise #1.0: Getting familiar with the interface

Materials:
- https://training-mwdb.readthedocs.io/en/latest/part-1.html#exercise-1-0-getting-familiar-with-the-interface
- https://mwdb.readthedocs.io/en/latest/user-guide/1-Introduction-to-MWDB.html

mwdb.cert.pl

Exercise #1.0: Getting familiar with the interface

Link to sample details

Click on tags to filter (or filter out)

mwdb.cert.pl

Exercise #1.0: Getting familiar with the interface

mwdb.cert.pl

Exercise #1.0: Getting familiar with the interface

mwdb.cert.pl

Exercise #1.1: Filtering samples by tags

Introduction

CERT.PL >_

mwdb.cert.pl

Exercise #1.1: Filtering samples by tags

formbook — Simple tag, mostly used for marking artifacts that are associated with malware family

feed:sample — Tag describing the source of malware sample

ripped:formbook — Tag for the original sample, indicating recognized malware family

runnable:win32:exe — Tag describing the type of sample
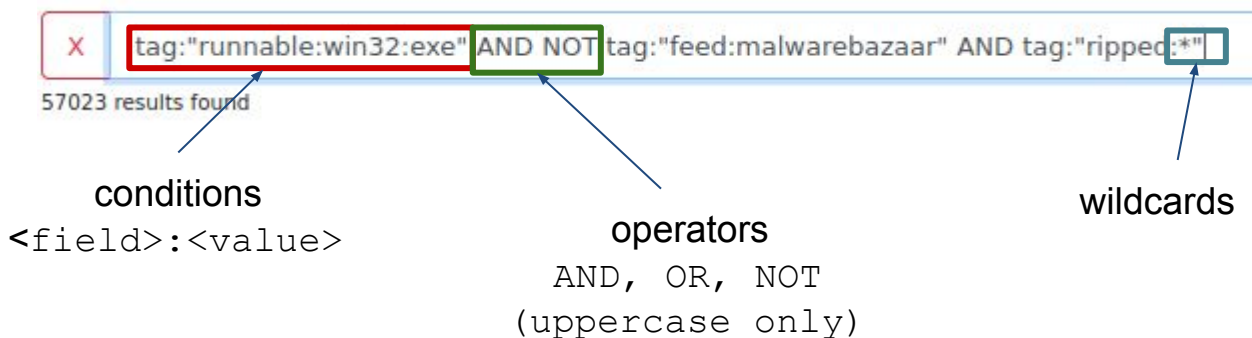
yara:win_formbook — Generic metadata tag with additional information that are useful for filtering

https://mwdb.readthedocs.io/en/latest/user-guide/5-Tagging-objects.html#built-in-tag-conventions

<CERT.PL >_

mwdb.cert.pl

Exercise #1.1: Filtering samples by tags
Lucene-based query syntax

```
X   tag:"runnable:win32:exe" AND NOT tag:"feed:malwarebazaar" AND tag:"ripped:*"
57023 results found
```

conditions
`<field>:<value>`

operators
`AND, OR, NOT`
`(uppercase only)`

wildcards

https://mwdb.readthedocs.io/en/latest/user-guide/7-Lucene-search.html

<CERT.PL >_

mwdb.cert.pl

Everything related with formbook

mwdb.cert.pl

Exercise #1.1: Filtering samples by tags
Ranges

size:[10000 TO 15000]

size:[10kB TO 15kB]

size:<=10kB

upload_time:<=2020-01-01

upload_time:"<=2020-01-01 16:00"

mwdb.cert.pl

Exercise #1.1: Filtering samples by tags

Goals: Get familiar with the interface, play around with the search query
- Include only `runnable:win32:exe` and `ripped:*` samples but exclude all coming from `feed:*`
- Click on tag with family name
- Add wildcards to family name to generalize to source of classification

Materials:
- https://training-mwdb.readthedocs.io/en/latest/part-1.html#exercise-1-1-filtering-samples-by-tags

<CERT.PL >_

mwdb.cert.pl

Exercise #1.2: Exploring sample view and hierarchy

Introduction

CERT.PL >_

mwdb.cert.pl

Exercise #1.2: Exploring sample view and hierarchy
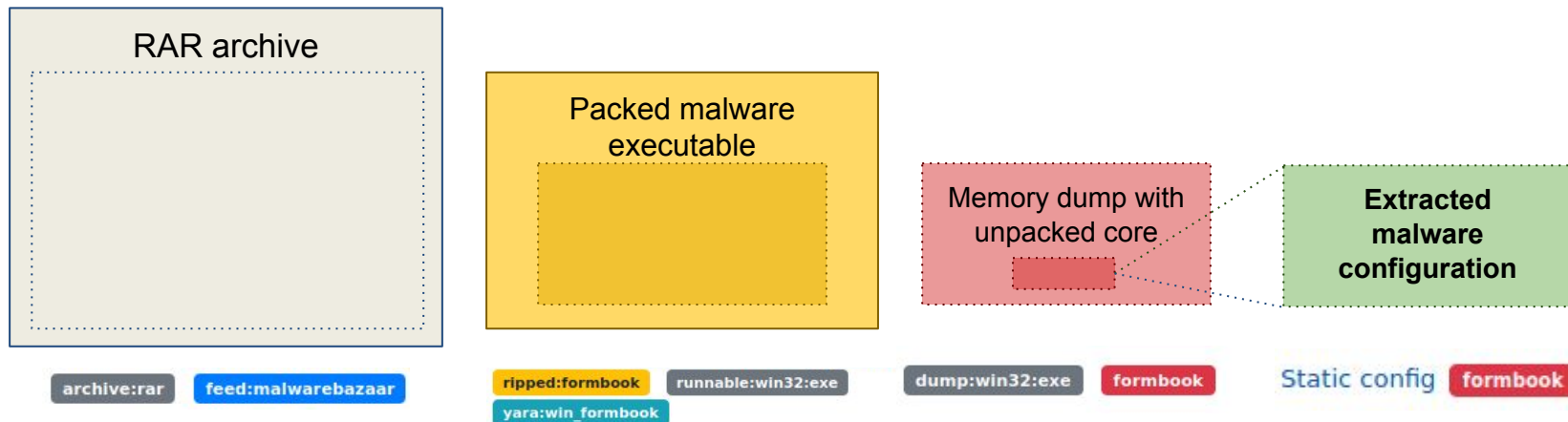
DRAKVUF Sandbox

Automated malware analysis system that is using DRAKVUF engine underneath (open source virtual machine introspection based agentless black-box binary analysis system by Tamas Lengyel et al.)

Uses various heuristics for choosing memory regions that may contain unpacked code.
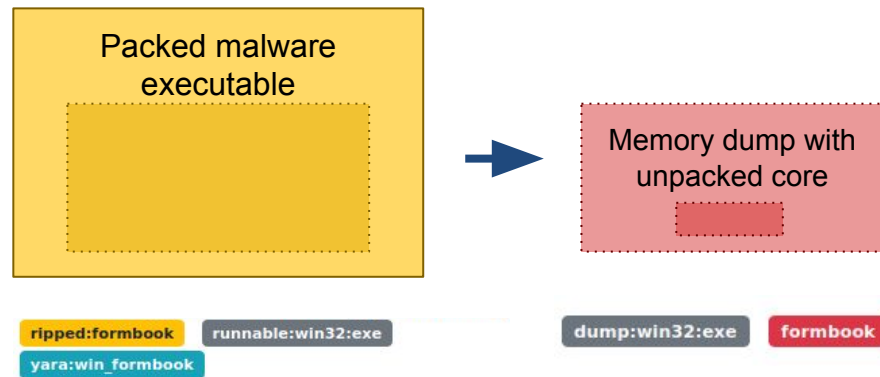
https://github.com/CERT-Polska/drakvuf-sandbox

https://github.com/tklengyel/drakvuf

mwdb.cert.pl

Exercise #1.2: Exploring sample view and hierarchy

mwdb.cert.pl
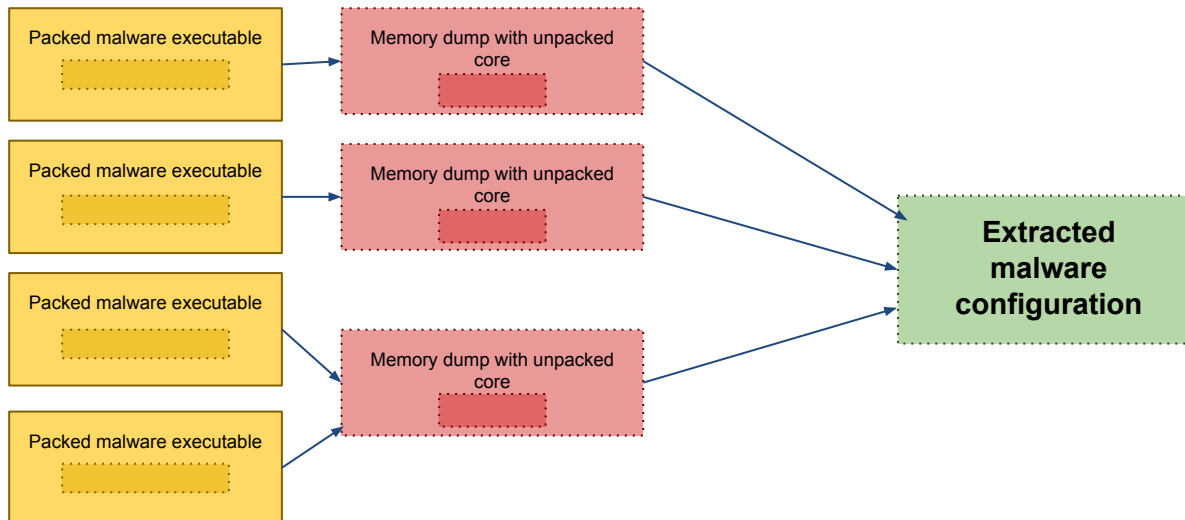
Exercise #1.2: Exploring sample view and hierarchy

Goals: Explore the sample view, understand the object hierarchy
- Navigate to `5762523a60685aafa8a681672403fd19`
- Follow the relationships and reach static configuration
- Go to Relations and check other parents of the configuration

Materials:
- https://training-mwdb.readthedocs.io/en/latest/part-1.html#exercise-1-2-exploring-sample-view-and-hierarchy

CERT.PL >_

mwdb.cert.pl

Exercise #1.3: Looking for similar configurations

Goals: Find configurations that are similar to the following Formbook config:

`f2e216695d4ce7233f5feb846bc81b8fffe9507988c7f5caaca680c0861e5e02`

- Click on URL to search for `www.discorddeno.land/suod/`
- Look for other configurations with path `/suod/`
- Exclude the configuration field and do full-text search on configuration
- Do the same for .land TLD. Do you see only configurations with .land TLD?

Materials:

- https://training-mwdb.readthedocs.io/en/latest/part-1.html#exercise-1-3-looking-for-similar-configurations

<CERT.PL >_

mwdb.cert.pl

Exercise #1.4: Blobs and dynamic configurations

Introduction
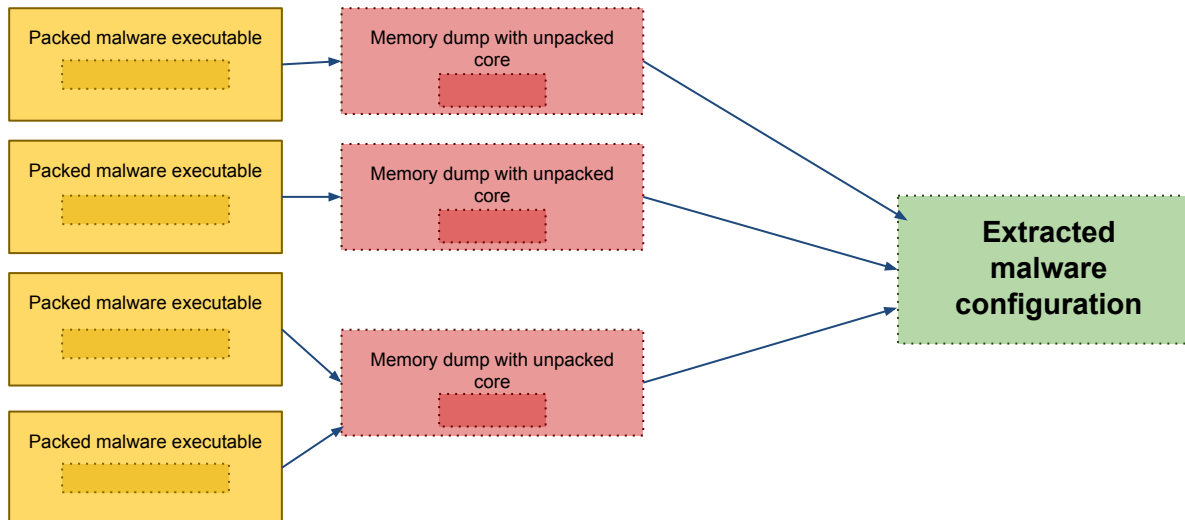
CERT.PL >_

mwdb.cert.pl

Exercise #1.4: Blobs and dynamic configurations

mwdb.cert.pl

Exercise #1.4: Blobs and dynamic configurations

mwdb.cert.pl
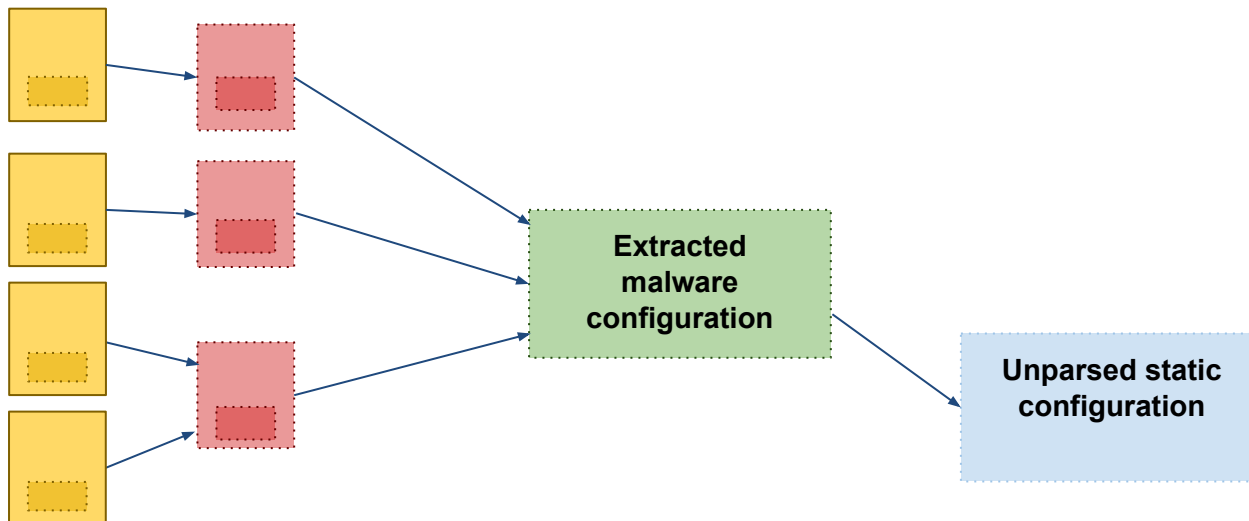
Exercise #1.4: Blobs and dynamic configurations

mtracker

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

mwdb.cert.pl
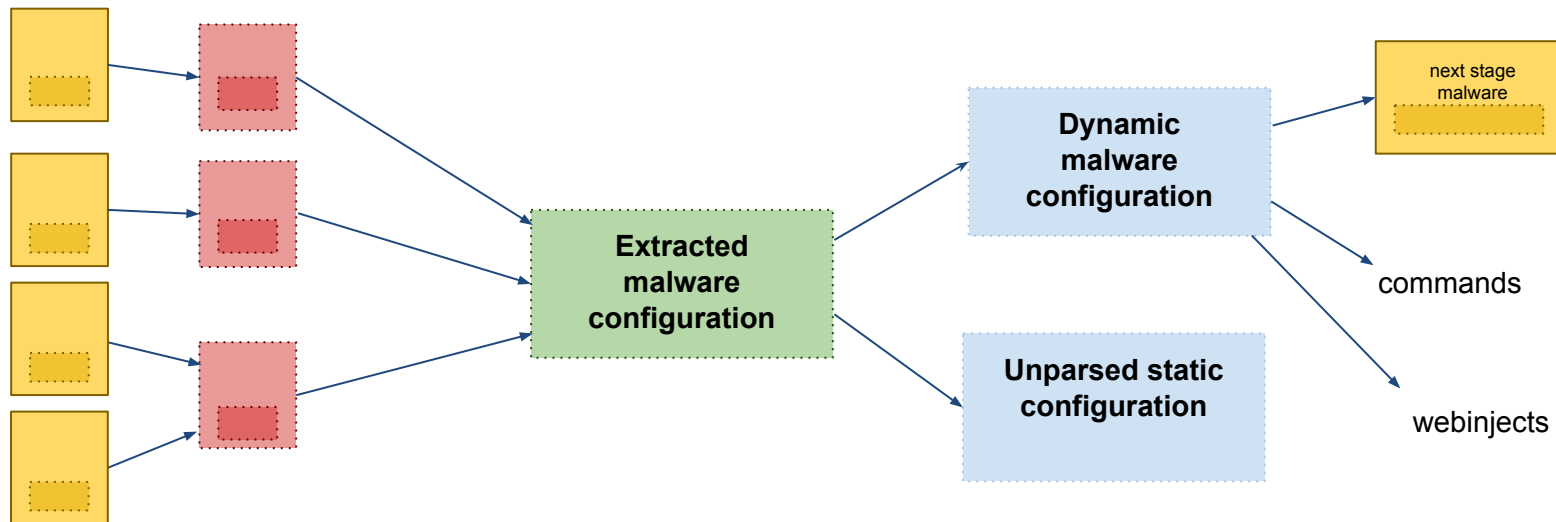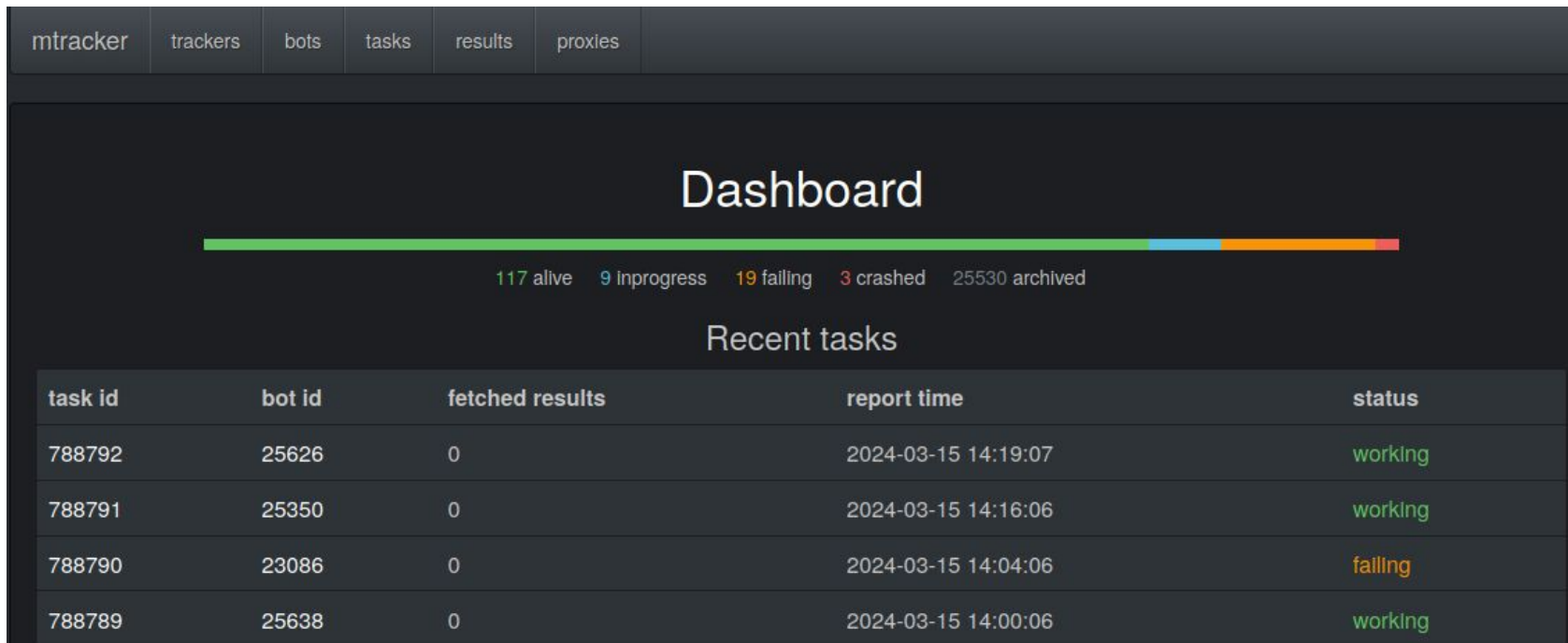
Exercise #1.4: Blobs and dynamic configurations

Goals: Familiarize yourself with the blob object type
- Take a look at AgentTesla and Remcos decrypted strings
- Find different configurations with `ongod4life.ddns.net:4344` and make a diff between related blobs
- Take a look at Hancitor dynamic configuration

Materials:
- https://training-mwdb.readthedocs.io/en/latest/part-1.html#exercise-1-4-blobs-and-dynamic-configurations

mwdb.cert.pl

Exercise #1.5: Let's upload something!

Introduction

CERT.PL >_

mwdb.cert.pl

Exercise #1.5: Let's upload something!

mwdb.cert.pl

Exercise #1.5: Let's upload something!

mwdb.cert.pl

Exercise #1.5: Let's upload something!

mwdb.cert.pl

Exercise #1.5: Let's upload something

Goals: Learn how object sharing and access inheritance work.
- Download malware sample from
  https://github.com/CERT-Polska/training-mwdb/raw/main/ex5malware.zip
- Upload to MWDB and check Shares tab
- Go to the child sample. What shares tab shows?

Materials:
- https://training-mwdb.readthedocs.io/en/latest/part-1.html#exercise-1-5-let-s-upload-something

<CERT.PL >_

Agenda

- **mwdb.cert.pl**
  - What the heck is **MW**
  - Tour de **mwdb.cert.p**
  - Scripting and automati   **mwdblib**
- **karton and malduck**
  - Run a **self-hosted** mwdb-core and karton instances
  - Experiment with **karton-playground**
  - Distributed collaboration with mwdb **remotes**
  - Advanced programming techniques with **malduck**

YOU ARE HERE

CERT.PL>_

mwdb.cert.pl

Coffee break

Prepare for later:

**$ git clone **
https://github.com/CERT-Pols
ka/karton-playground.git

$ cd karton-playground

$ docker-compose pull
(or "docker compose pull")
(it will take a while)

CERT.PL>_
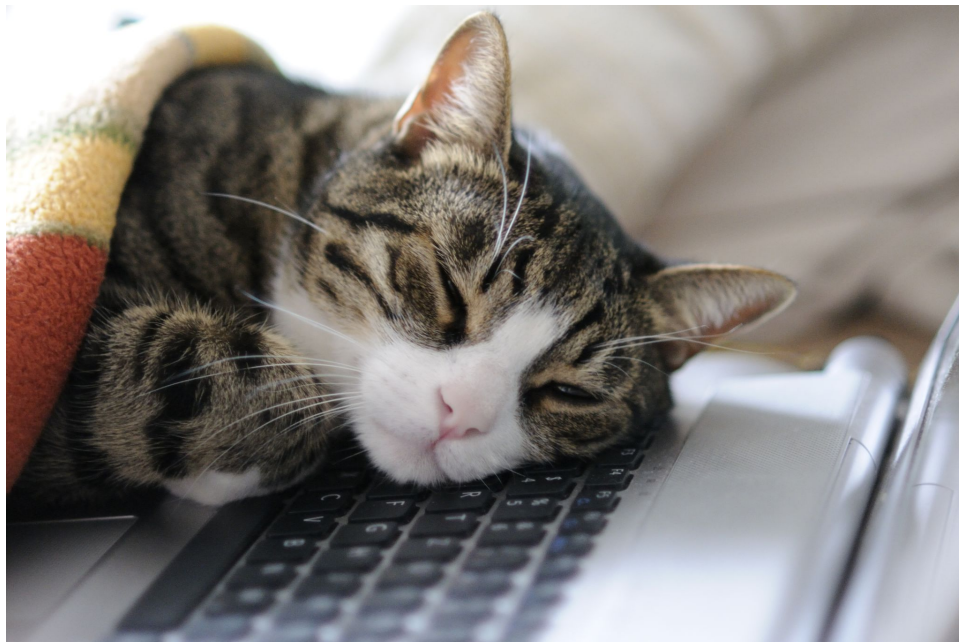
mwdblib installation
Setup environment

```
# Create virtualenv and activate
$ python3 -m venv venv
$ . venv/bin/activate
# On Debian/Ubuntu you might need to install python3-venv
```

```
# On older distributions - upgrade pip:
(venv)$ pip install -U pip
# Install mwdblib with CLI extras
(venv) $ pip install mwdblib[cli]
# ... and ipython for convenience
(venv) $ pip install ipython
```

If you don't know what is virtualenv, read more:
[Installing packages using pip and virtual environments — Python Packaging User Guide](#)

mwdb.cert.pl

Exercise #2.1: Get recent files

Goals:
- Login using mwdblib and use `recent_files` method
- Get information about 10 most recent files

Materials:
- https://training-mwdb.readthedocs.io/en/latest/part-2.html#exercise-2-1-get-information-about-10-recent-files-using-mwdblib
- https://mwdblib.readthedocs.io/

CERT.PL >_

mwdb.cert.pl

Exercise #2.2: MWDBObject properties

Goals: Get information about `780e8fb254e0b8c299f834f61dc80809`
- Check file's name, tags and children
- Get the first 16 bytes of the file
- Get the configuration linked to this file
- Check names of the other files that are parents of that configuration

Materials:
- https://training-mwdb.readthedocs.io/en/latest/part-2.html#exercise-2-2-check-properties-of-780e8fb254e0b8c299f834f61dc80809
- https://mwdblib.readthedocs.io/

Task: Use `mwdb.search_configs("family:valak")` to get a list of all URLs referenced by the valak family (config field `urls`). How many URLs are there in total?

mwdb.cert.pl

Exercise #2.3: Using mwdblib CLI

Goals: Learn to use mwdblib CLI component
- Download 10 files that were tagged as ripped:lokibot using mwdblib CLI

Materials:
- https://training-mwdb.readthedocs.io/en/latest/part-2.html#exercise-2-3-using-mwdblib-cli

Task: Use mwdb get to get information about hash `c6f50cb47d61092240bc9e7fd6631451ddb617011ab038b42a674585668dc54a`. What is the malware family of this sample (you can use the tags to get this information)?

<CERT.PL >_

mwdb.cert.pl

Exercise #2.4: Joining CLI with other tools

Goals: Get 10 most recent Mutexes from nanocore configs
- `mwdb fetch` can also fetch configurations in JSON format
- You can select things from JSONs using jq tool

Materials:
- https://training-mwdb.readthedocs.io/en/latest/part-2.html#exercise-2-4-joining-cli-with-other-tools

<CERT.PL >_

MWDB + Yara = bff

- MWDB (with plugins) also has support for searching with Yara rules

- This feat is achieved with mquery integration

- Mquery is a whole another open-source project that you can use to manage your corpus.

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

- There is an unofficial public instance of **mquery** that you can use to find some samples, reachable via https://mquery.net.

Learn karton with the karton-playground

Run a self-hosted mwdb-core and karton instance

mwdb-core

mwdb-core
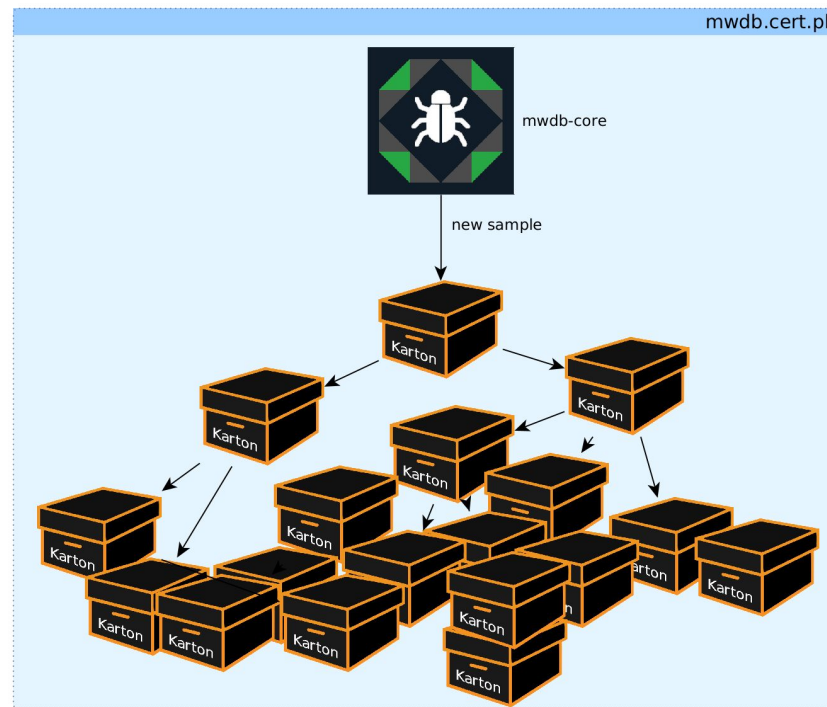
mwdb-core

mwdb-core + karton = a usable service

MWDB is only a frontend.

To make it possible to create an environment similar to ours, we've decided to open-source the "engine" of our pipeline too.

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

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

Karton Playground

- [Karton Playground](#) - a project dedicated for karton learners
- An easy way to set up the environment and get to work
- Not suitable for production
- https://github.com/CERT-Polska/karton-playground

image credit: wikipedia

CERT.PL >_

Karton Playground

```
git clone https://github.com/CERT-Polska/karton-playground.git
cd karton-playground
sudo docker-compose up   # this may take a while
```

- This may take a while. But when it's done, you will have a working instance on your local machine

CERT.PL >_

Karton Playground

While you wait...

mwdb-core

new samples
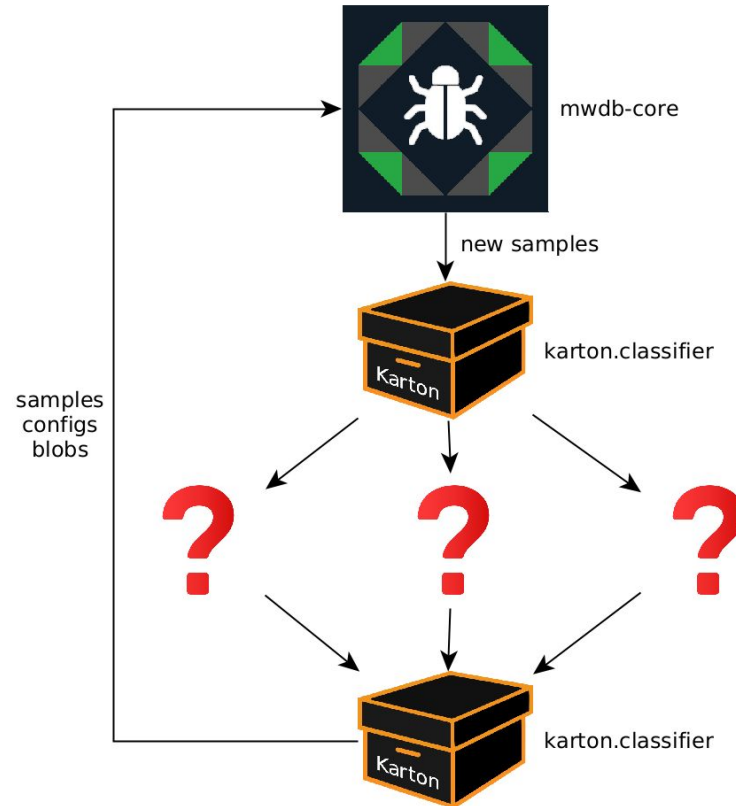
karton.classifier

samples
configs
blobs

karton.classifier

Karton pipeline

Karton pipeline

karton.extractor

`2.x.x`

`kind:archive` `stage:recognized` `type:sample`
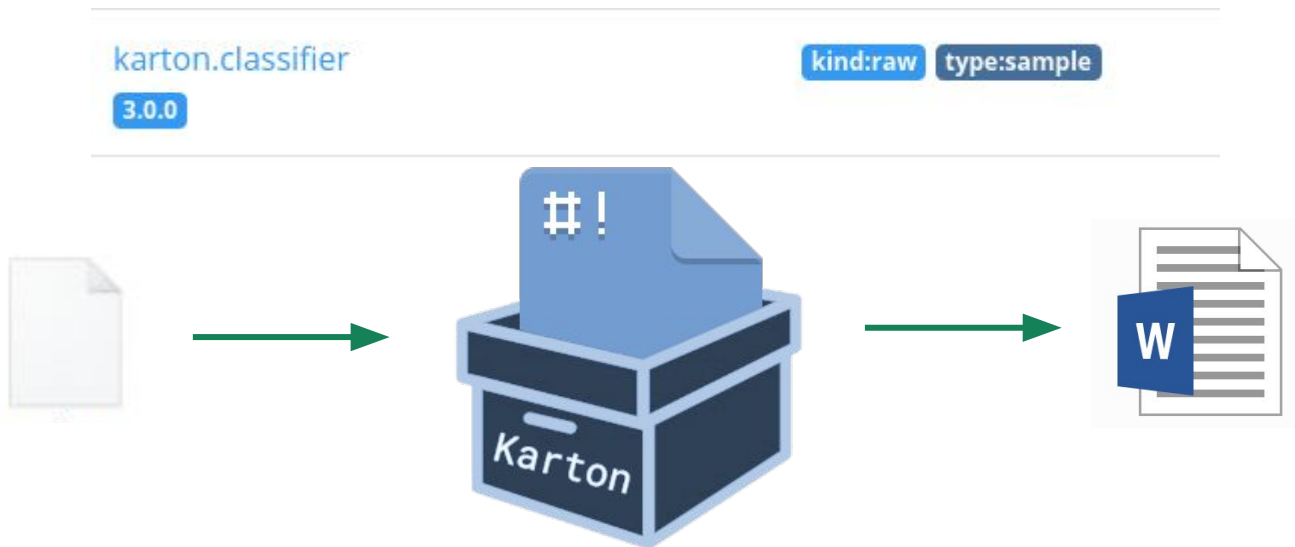
Karton pipeline

Karton pipeline
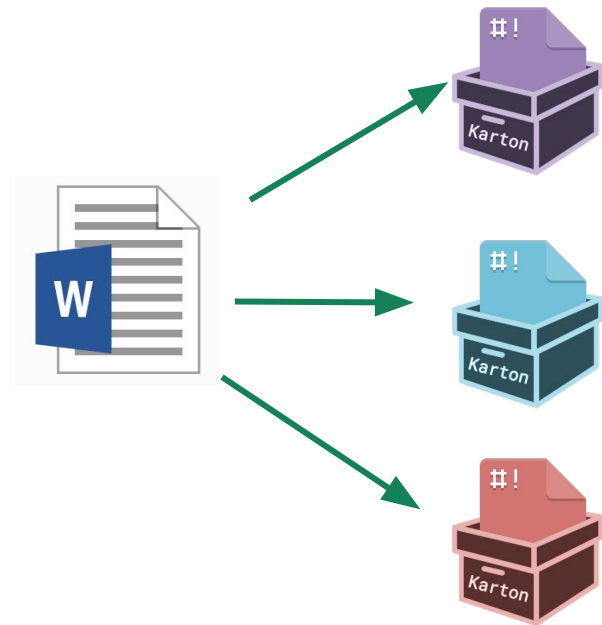
karton.cuckoo1
2.x.x
`platform:win32` `quality:high` `stage:recognized` `type:sample`

karton.drakrun-prod
2.x.x
`platform:win32` `stage:recognized` `type:sample`
`platform:win64` `stage:recognized` `type:sample`

karton.macro-unpacker
3.1.0
`extension:xlsx` `kind:document` `stage:recognized` `type:sample`
`extension:xlsm` `kind:document` `stage:recognized` `type:sample`
`extension:xls` `kind:document` `stage:recognized` `type:sample`
`extension:doc` `kind:document` `stage:recognized` `type:sample`

karton.emodoc
2.x.x
`extension:doc` `kind:document` `stage:recognized` `type:sample`
`extension:docx` `kind:document` `stage:recognized` `type:sample`

Karton pipeline

karton.drakrun-prod
2.x.x

`platform:win32` `stage:recognized` `type:sample`
`platform:win64` `stage:recognized` `type:sample`

karton.macro-unpacker
3.1.0

`extension:xlsx` `kind:document` `stage:recognized` `type:sample`
`extension:xlsm` `kind:document` `stage:recognized` `type:sample`
`extension:xls` `kind:document` `stage:recognized` `type:sample`
`extension:doc` `kind:document` `stage:recognized` `type:sample`

DMP DMP DMP

http://

Karton pipeline

karton.roach-ripper

2.x.x

kind:runnable platform:win32 stage:recognized type:sample
kind:runnable platform:win64 stage:recognized type:sample
kind:runnable platform:linux stage:recognized type:sample
kind:drakrun-prod type:analysis

DMP

#!

Karton

Family	zloader
Config type	static
+ binary_id	bot7
+ rc4key	NvuVIV3kbg7
+ rsa_key	-----BEGIN PUBLIC KEY----- MIG
+ type	zloader
+ urls	[{ "url": "https://militanttr

Karton pipeline in the real world

Karton Playground: click around

Navigate to http://127.0.0.1:8080. Login using `admin:admin`.

Karton Playground: click around

Navigate to http://127.0.0.1:8080. Login using admin:admin.

There is no malware yet… But that's about to change!

<CERT.PL >_

Karton Playground: click around

Check out the karton dashboard at http://127.0.0.1:8030/ too:

Karton Playground: click around

Optional: check out the minio interface http://127.0.0.1:8090/ (mwdb:mwdbmwdb)
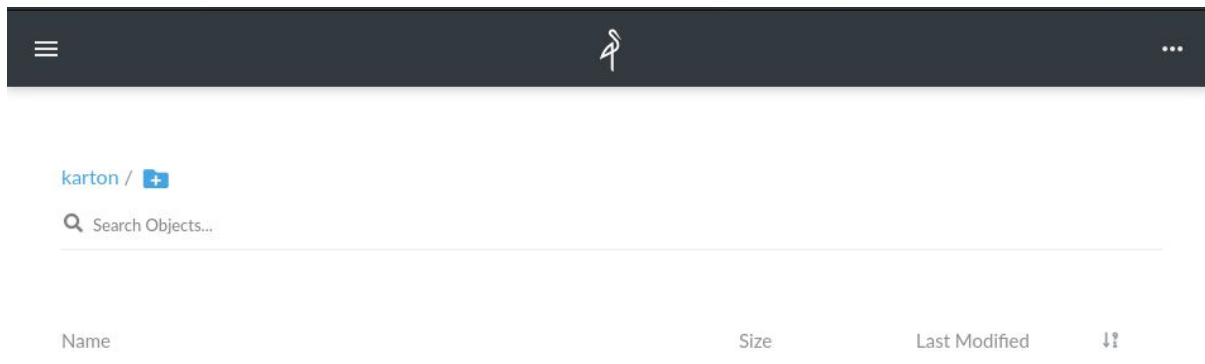
Karton Playground exercise #3.1

Integrate an existing karton service into your pipeline: karton-autoit-ripper

https://github.com/CERT-Polska/karton-autoit-ripper

```
$ python3 -m venv venv
$ source ./venv/bin/activate
$ pip install karton-autoit-ripper

$ # playground-specific: copy local config to cwd
$ cp config/karton.ini karton.ini
$ karton-autoit-ripper
[2021-04-11 17:19:57,867][INFO] Service karton.autoit-ripper started
```

Use MWDB to analyze the this sample:
https://github.com/CERT-Polska/training-mwdb/blob/main/autoit-malware.bin

<CERT.PL >_

Karton Playground exercise

Download a sample, and verify its hash:

```
$ wget https://github.com/CERT-Polska/training-mwdb/blob/main/autoit-malware.bin
$ sha256sum autoit-malware.bin
a4816d4fecd6d2806d5b105c3aab55f4a1eb5deb3b126f317093a4dc4aab88a1 autoit-malware.bin
```

Finally, upload it to your local mwdb (http://127.0.0.1:8080, admin:admin)

```
$ karton-autoit-ripper
[2021-04-11 17:19:57,867][INFO] Service karton.autoit-ripper started
/home/msm/Projects/karton-playground/venv/lib/python3.8/site-packages/karton/core/logger.py:57: UserWarning: There is no active
  warnings.warn("There is no active log consumer to receive logged messages.")
[2021-04-11 17:19:57,871][INFO] Binding on: {'type': 'sample', 'stage': 'recognized', 'kind': 'runnable', 'platform': 'win32'}
[2021-04-11 17:19:57,871][INFO] Binding on: {'type': 'sample', 'stage': 'recognized', 'kind': 'runnable', 'platform': 'win64'}
[2021-04-11 17:20:10,645][INFO] Received new task - cbe177c0-a824-47be-a1c9-fb0aa4898f75
[2021-04-11 17:20:10,661][INFO] Found a possible autoit v3.26+ binary
[2021-04-11 17:20:14,149][INFO] Found embedded data, reporting!
[2021-04-11 17:20:14,150][INFO] Sending a task with script.au3
[2021-04-11 17:20:14,261][INFO] Looking for a binary embedded in the script
[2021-04-11 17:20:14,305][INFO] Task done - cbe177c0-a824-47be-a1c9-fb0aa4898f75
```


CERT.PL >_

Karton Playground exercise

Volia!

Karton Playground exercise

- But using existing services is just half the fun
- For a real Karton experience, write your own service
- Download a template:
  https://github.com/CERT-Polska/training-mwdb/blob/main/karton-template.py

```python
class MyFirstKarton(Karton):
    identity = "karton.first"
    filters = [{"type": "sample", "stage": "recognized"}]

    def process(self, task: Task) -> None:
        sample_resource = task.get_resource("sample")  # Get the incoming sample
        self.log.info(f"Hi {sample_resource.name}, let me analyse you!")  # Log with self.log

        with sample_resource.download_temporary_file() as sample_file:  # Download to a temporary file
            result = do_your_processing(sample_file.name)  # And process it

        self.send_task(Task(
            {"type": "sample", "stage": "analyzed"},
            payload={"parent": sample_resource, "sample": Resource("result-name", result)},
        ))  # Upload the result as a sample:

if __name__ == "__main__":
    MyFirstKarton().loop()  # Here comes the main loop
```

Karton Playground exercise

- Karton's "identity":   identity = "**karton.first**"
- Python namespace: import   **karton.first**
- Pypi package: pip install **karton-first**

```python
class MyFirstKarton(Karton):
    identity = "karton.first"
    filters = [{"type": "sample", "stage": "recognized"}]

    def process(self, task: Task) -> None:
        sample_resource = task.get_resource("sample")  # Get the incoming sample
        self.log.info(f"Hi {sample_resource.name}, let me analyse you!")  # Log with self.log

        with sample_resource.download_temporary_file() as sample_file:  # Download to a temporary file
            result = do_your_processing(sample_file.name)  # And process it

        self.send_task(Task(
            {"type": "sample", "stage": "analyzed"},
            payload={"parent": sample_resource, "sample": Resource("result-name", result)},
        ))  # Upload the result as a sample:

if __name__ == "__main__":
    MyFirstKarton().loop()  # Here comes the main loop
```

Karton Playground exercise

- What are **these**?

```python
class MyFirstKarton(Karton):
    identity = "karton.first"
    filters = [{"type": "sample", "stage": "recognized"}]

    def process(self, task: Task) -> None:
        sample_resource = task.get_resource("sample")  # Get the incoming sample
        self.log.info(f"Hi {sample_resource.name}, let me analyse you!")  # Log with self.log

        with sample_resource.download_temporary_file() as sample_file:  # Download to a temporary file
            result = do_your_processing(sample_file.name)  # And process it

        self.send_task(Task(
            {"type": "sample", "stage": "analyzed"},
            payload={"parent": sample_resource, "sample": Resource("result-name", result)},
        ))  # Upload the result as a sample:

if __name__ == "__main__":
    MyFirstKarton().loop()  # Here comes the main loop
```

Karton Playground exercise

- Karton tasks are routed in the system based on their headers
- Consumer declares what kind of tasks is it interested in
- Producer indicates the kind of produced task

```python
class MyFirstKarton(Karton):
    identity = "karton.first"
    filters = [{"type": "sample", "stage": "recognized"}]

    def process(self, task: Task) -> None:
        sample_resource = task.get_resource("sample")  # Get the incoming sample
        self.log.info(f"Hi {sample_resource.name}, let me analyse you!")  # Log with self.log

        with sample_resource.download_temporary_file() as sample_file:  # Download to a temporary file
            result = do_your_processing(sample_file.name)  # And process it

        self.send_task(Task(
            {"type": "sample", "stage": "analyzed"},
            payload={"parent": sample_resource, "sample": Resource("result-name", result)},
        ))  # Upload the result as a sample:

if __name__ == "__main__":
    MyFirstKarton().loop()  # Here comes the main loop
```

<ERT.PL >_

Karton Playground exercise

- Karton = Consumer + Producer

```python
class Karton(Consumer, Producer):
    """
    This glues together Consumer and Producer - which is the most common use case
    """
```

```python
class MyFirstKarton(Karton):
    identity = "karton.first"
    filters = [{"type": "sample", "stage": "recognized"}]

    def process(self, task: Task) -> None:
        sample_resource = task.get_resource("sample")  # Get the incoming sample
        self.log.info(f"Hi {sample_resource.name}, let me analyse you!")  # Log with self.log

        with sample_resource.download_temporary_file() as sample_file:  # Download to a temporary file
            result = do_your_processing(sample_file.name)  # And process it

        self.send_task(Task(
            {"type": "sample", "stage": "analyzed"},
            payload={"parent": sample_resource, "sample": Resource("result-name", result)},
        ))  # Upload the result as a sample:

if __name__ == "__main__":
    MyFirstKarton().loop()  # Here comes the main Loop
```

<CERT.PL>_

Karton Playground exercise

- Resource - bigger files, hosted on minio (or other S3 compatible storage server)

```python
class MyFirstKarton(Karton):
    identity = "karton.first"
    filters = [{"type": "sample", "stage": "recognized"}]

    def process(self, task: Task) -> None:
        sample_resource = task.get_resource("sample")  # Get the incoming sample
        self.log.info(f"Hi {sample_resource.name}, let me analyse you!")  # Log with self.log

        with sample_resource.download_temporary_file() as sample_file:  # Download to a temporary file
            result = do_your_processing(sample_file.name)  # And process it

        self.send_task(Task(
            {"type": "sample", "stage": "analyzed"},
            payload={"parent": sample_resource, "sample": Resource("result-name", result)},
        ))  # Upload the result as a sample:

if __name__ == "__main__":
    MyFirstKarton().loop()  # Here comes the main loop
```

Karton Playground exercise #3.2

- Download a template:
  https://github.com/CERT-Polska/training-mwdb/blob/main/karton-template.py
- Your task: edit the template, and:
  - Run the `strings` utility on every incoming sample
  - Save the result in a variable (use `subprocess.check_output`)
  - Upload the result to mwdb (already handled in the template)
- Start your first karton service!

```
$ python3
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
>>> import subprocess
>>> s = subprocess.check_output(["strings", "/bin/ls"])
>>> print(s.decode())
/lib64/ld-linux-x86-64.so.2
.j<c~
MB#F-
Libselinux.so.1
...
```

Karton Playground exercise: solution

CERT.PL >_

Karton Playground exercise: solution

CERT.PL >_

Coffee break

Malduck 🦆


```
00000000  26 00 00 00  73 00 6b 00   79 00 72 00 6f 00 63 00   &...s.k.y.r.o.c.
00000010  6b 00 65 00 74 00 2e 00   6f 00 6f 00 67 00 75 00   k.e.t...o.o.g.u.
00000020  79 00 2e 00 63 00 6f 00   6d 00 c7 0b 00 00 00 00   y...c.o.m.......
00000030  00 00 00 00 00 00 00 00   00 00 00 00 d0 07 00 00   ................
00000040  14 00 00 00 54 00 57 00   4e 00 4d 00 55 00 35 00   ....T.W.N.M.U.5.
00000050  4f 00 34 00 54 00 39 00                             O.4.T.9.
```

```
struct config:
- domain_length: 0x26
- domain: b's\x00k\x00y\x00r\x00o\x00c\x00k\x00e\x00t\x00.\x00o\x00o\x00g\x00u
- port: 0xbc7
```
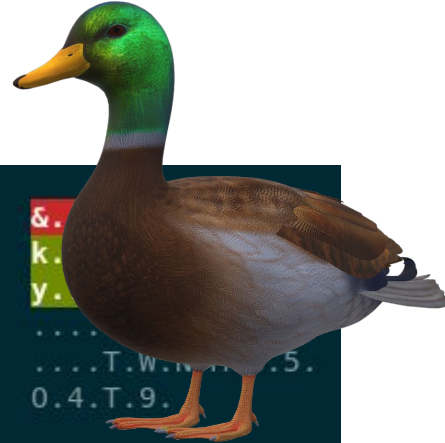
Malduck 🦆

We will **skip** this part (check out slides from hacklu or botconf if you're interested)

"Malduck is your ducky companion in malware analysis journeys"

Basically three things for the price of one:
- A set of neat low-level utilities (crypto, compression, uints, disasm, etc)
- Memory model objects (work on memory dumps, PE/ELF, raw files and IDA dumps using the same code)
- Extraction engine (modular extraction framework for config extraction from files/dumps) - used by karton.config-extractor

Malduck 🦆 : sharing is caring

We're still thinking about a sharing model that will work for malduck modules.

- ● We recently TLP:WHITE open-sourced some module examples: https://github.com/CERT-Polska/malduck-modules
  check them out if you want to start with malduck/config-extractor
- ● We plan to create a TLP:AMBER sharing group someday
  thoughts?
- ● And if you create a malduck module, you can always share it with us and we'll use it in mwdb.cert.pl for everyone's benefit.
  but no pressure

CERT.PL >_

mquery 101

- Another Open-Source project https://github.com/CERT-Polska/mquery

- Yara query accelerator - search through large datasets using Yara

- Let's just do a very quick self-hosted exercise (if we have time), and then test a query on the hosted instance.

- But first, Yara 101

yara 101

- YARA is a simple language for matching patterns in binary files
- Industry standard in IT security
- Syntax reference: https://yara.readthedocs.io/en/stable/writingrules.html

```
rule MyRuleName
{
    meta:
        description = "Example rule"
        author = "author name"
    strings:
        $text_string = "SELECT * FROM Win32_ComputerSystem"
        $binary_data = { 4d 5a ff ff }
    condition:
        $text_string or $binary_data
}
```

yara 101

- "meta" section - arbitrary key-value pairs

```
rule MyRuleName
{
    meta:
        description = "Example rule"
        author = "author name"
    strings:
        $text_string = "SELECT * FROM Win32_ComputerSystem"
        $binary_data = { 4d 5a ff ff }
    condition:
        $text_string or $binary_data
}
```

<CERT.PL>_

yara 101

- "strings" section - things to look for in the binary
- "Text" strings
    - optional modifiers like "ascii", "wide", "nocase"
- Hex strings for binary data
    - May use wildcards, like { 11 ?? ?? ?? 33 } or {11 [3] 33}.
- Regular expressions like /abc[a-z]+/ also supported, use with caution.

```
rule MyRuleName
{
    meta:
        description = "Example rule"
        author = "author_name"
    strings:
        $text_string = "SELECT * FROM Win32_ComputerSystem"
        $binary_data = { 4d 5a ff ff }
    condition:
        $text_string or $binary_data
}
```

<CERT.PL>_

yara 101

- "condition" section - how to combine strings to determine match
  - $string_name
  - $string1 or $string2
  - $string1 and $string2
  - all of ($string*)
  - all of them

```
rule MyRuleName
{
    meta:
        description = "Example rule"
        author = "author name"
    strings:
        $text_string = "SELECT * FROM Win32_ComputerSystem"
        $binary_data = { 4d 5a ff ff }
    condition:
        $text_string or $binary_data
}
```

<CERT.PL>_

self-hosting mquery

- git clone https://github.com/CERT-Polska/mquery.git
- cd mquery
- docker-compose up

- cp ~/Downloads/training-mwdb/ ./samples/ -r
- sudo docker-compose exec ursadb ursacli
- index "/mnt/samples";

- Check the database status: http://localhost/status
- Execute a rule that finds the unique sample with a "NtQueryInformationProcess" string
  - Go to the main page: http://localhost/
  - Write "rule"<tab>
  - Change string to "$name = "NtQueryInformationProcess""
  - Press "query"

Getting a bit more realistic

- Write a Yara rule that finds samples with the following characteristics:
    - It contains an ASCII string "StartCryptDesctop"
    - It contains an unicode string "RSAKeyValue" (use the "wide" modifier)

- Visit https://mquery.net (this is not an official instance of mquery)
- Run your query. It should finish quickly, and match exactly one file.

<CERT.PL >_

Wraping up

- mwdb (malware database)
  - to store, organise, query and share your malware collection
- Karton - used to orchestrate the elements of analytic pipeline
  - to analyse your malware
- Malduck - a helper library used by analysts and config extractors alike
  - to extract configuration from your malware
- mquery - Yara query accelerator
  - to search through your malware collection

Wraping up

- mwdb (malware database)
  - to store, organise, query and share your malware collection
- Karton - used to orchestrate the elements of analytic pipeline
  - to analyse your malware
- Malduck - a helper library used by analysts and config extractors alike
  - to extract configuration from your malware
- mquery - Yara query accelerator
  - to search through your malware collection
- And there's more
  - Mtracker - to download payloads and configs from C2 servers
  - Drakvuf - alternative malware sandbox that uses VMI and EPT for stealth
  - Outside of pure malware world - Artemis, Mailgoose, n6

Wraping up

- mwdb (malware database)
  - to store, organise, query and share your malware collection
- Karton - used to orchestrate the elements of analytic pipeline
  - to analyse your malware
- Malduck - a helper library used by analysts and config extractors alike
  - to extract configuration from your malware
- mquery - Yara query accelerator
  - to search through your malware collection
- And there's more
  - Mtracker - to download payloads and configs from C2 servers
  - Drakvuf - alternative malware sandbox that uses VMI and EPT for stealth
  - Outside of pure malware world - Artemis, Mailgoose, n6
  - Maybe more to come?
  - But that's a story for another day.

Q & A

https://**github.com/**CERT-Polska/
https://**mwdb.readthedocs.io/**
https://**karton-core.readthedocs.io/**en/latest/
https://**malduck.readthedocs.io/**
https://**mwdb.cert.pl/**
https://**cert.pl/**en/

pawel.srokosz@cert.pl
jaroslaw.jedynak@cert.pl
info@cert.pl

CERT.PL >_

Co-financed by the Connecting Europe Facility of the European Union