

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

Jarosław Jedynak

Paweł Srokosz



Annual FIRST Conference
Fukuoka, 8th June 2024

Agenda

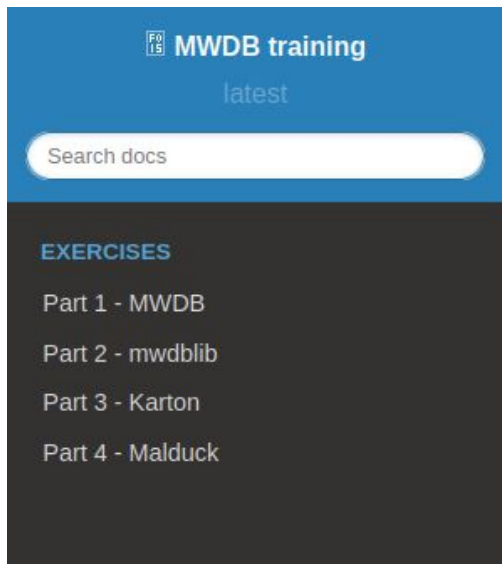
- **mwdb.cert.pl**
 - What the heck is **MWDB**?
 - Tour de **mwdb.cert.pl**
 - Scripting and automation with **mwdblib**
- **karton and malduck**
 - Run a **self-hosted** mwdb-core and karton instances
 - Experiment with **karton-playground**
 - Automated unpacking with **malduck**
- **mquery 101**

Prerequisites

A few things before we get started

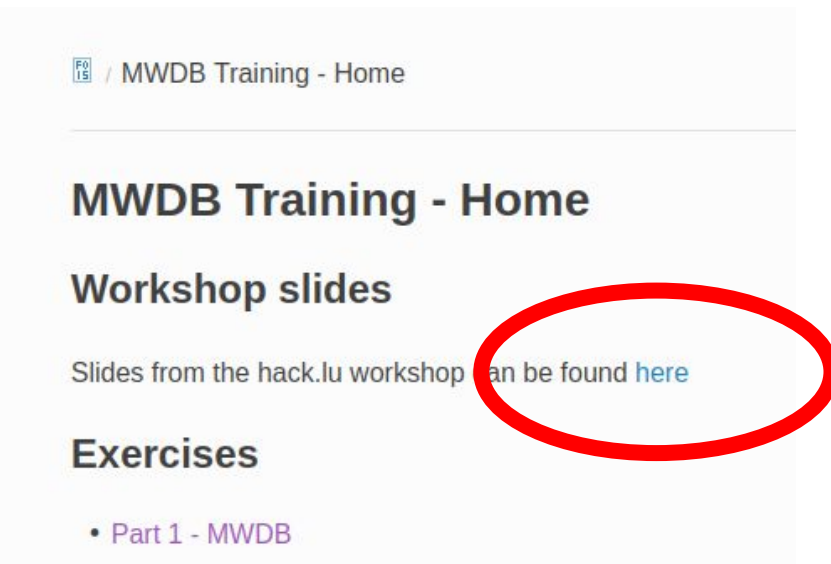
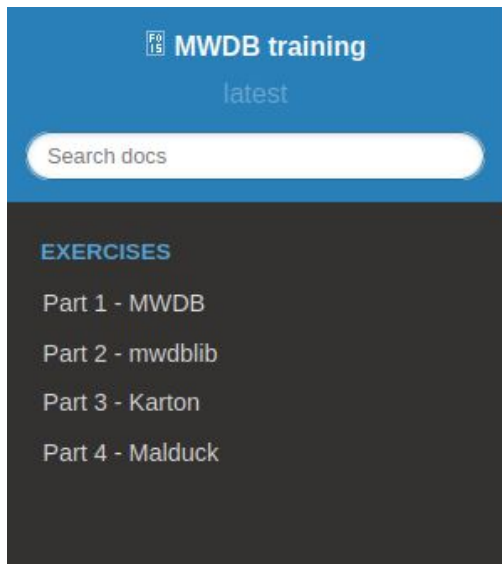
Training materials (very important)

<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](#).

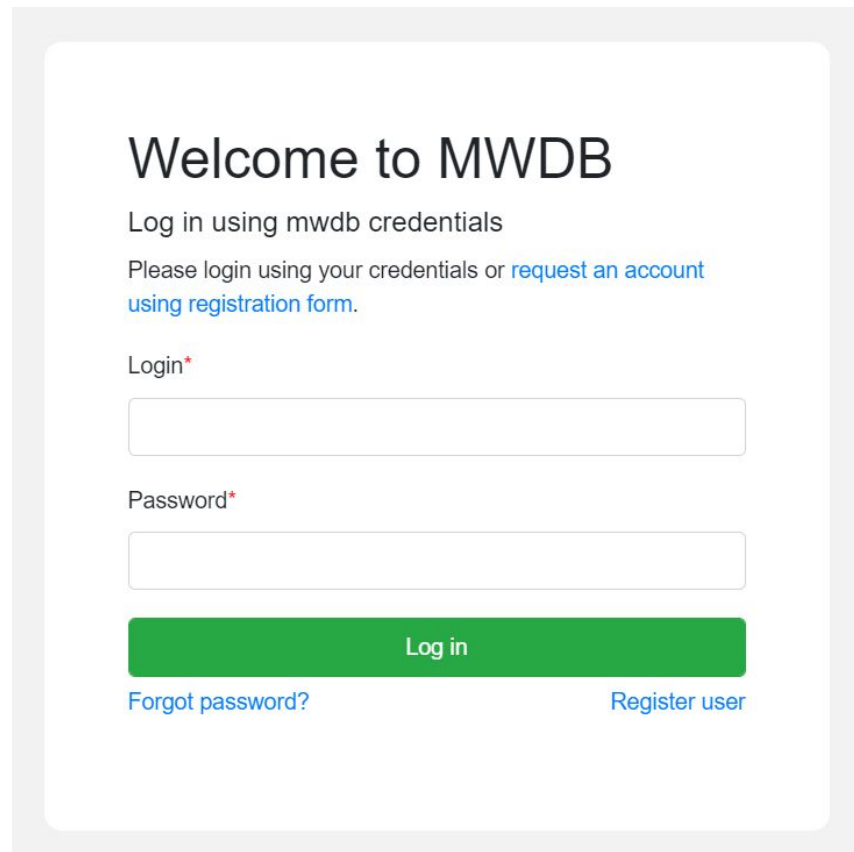
MWDB account

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.

A screenshot of the MWDB login page. The page has a white background with a light gray border. At the top, it says "Welcome to MWDB" in a large, dark gray font. Below this, it says "Log in using mwdb credentials" in a smaller, dark gray font. Then, it says "Please login using your credentials or [request an account using registration form.](#)" in a smaller, dark gray font. Below this, there are two input fields: "Login*" and "Password*", both with red asterisks. The "Login*" field is empty. The "Password*" field is empty. Below the input fields, there is a green button with the text "Log in" in white. At the bottom, there are two links: "[Forgot password?](#)" and "[Register user](#)", both in blue text.

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

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

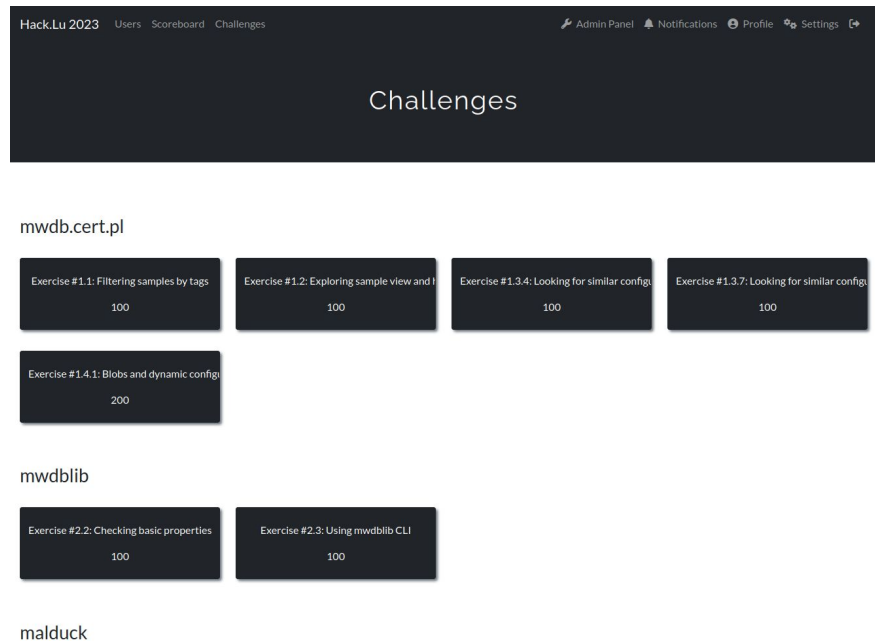
CTFd (optional but recommended)

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.

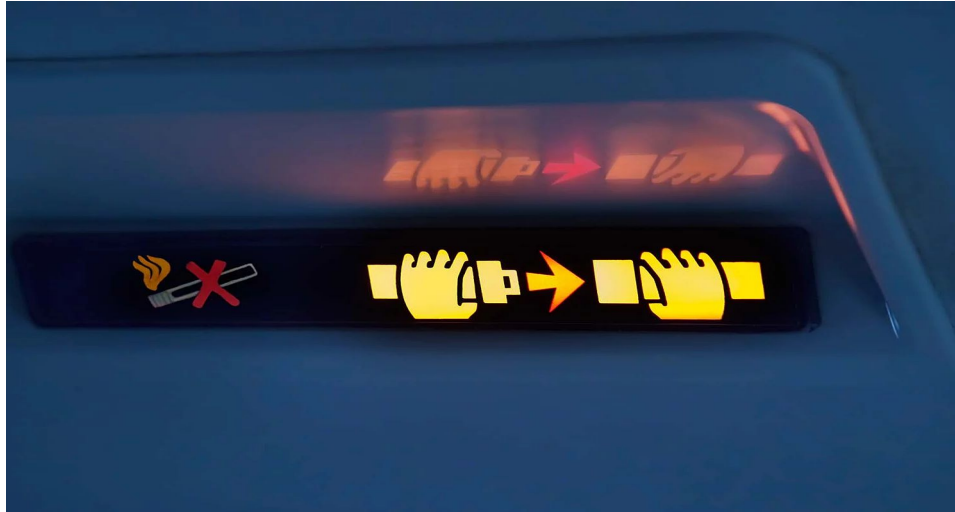


mwdb.cert.pl



#1. Tour de mwdb.cert.pl

<https://mwdb.cert.pl>



mwdb.cert.pl



Let's start!

Login to MWDB

The screenshot shows a web browser window with the address bar displaying `https://mwdb.cert.pl/login`. The page has a dark blue header with the **CERT.PL** logo and links for "Register user" and "About". A pink message box states: "You need to authenticate before accessing this page". Below this is the "Login" section, which includes the text: "Please login using your credentials or [request an account using registration form](#) or write an e-mail to info@cert.pl". There are two input fields: "Login" and "Password". A link for "Forgot password?" is located below the password field. A blue "Submit" button is at the bottom of the login form.

← → ↻ 🏠 🔒 `https://mwdb.cert.pl/login` ⋮ 📧 ☆ 📄 🖨️ ☰

CERT.PL > Register user About ▾

You need to authenticate before accessing this page

Login

Please login using your credentials or [request an account using registration form](#) or write an e-mail to info@cert.pl.

Login

Password

[Forgot password?](#)

Submit

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






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>



Exercise #1.0: Getting familiar with the interface

CERT.PL >_			
Samples Configs Blobs Upload Yara search ▾ Search Groups Statistics About ▾ Logged as: Profile Logout			
<div>X Search (Lucene query or hash)...</div> <div>Quick query: Only uploaded by me Exclude public Favorites Exclude feed:* Only ripped:* Add +</div>			
Name/Hash	Size/Type	Tags	First seen
 Name: K8I57NYN3.exe SHA256: 902eb186a...1669bc377c4f MD5: b9842537f...68da8e0092f2	Size: 1.47 MB Type: PE32 executable (GUI) I...	<div>feed:urlhaus runnable:win32:exe urlhaus:exe urlhaus:finderbot urlhaus:opendir</div>	Sun, 11 Apr 2021 14:44:28 GMT
 Name: jew.mpsl SHA256: 1d2e11bc0...ed53c5a78b3d MD5: 19830e713...e01990b4dc42	Size: 94.21 kB Type: ELF 32-bit LSB execut...	<div>feed:urlhaus mirai ripped:mirai runnable:linux urlhaus:elf urlhaus:mirai</div>	Sun, 11 Apr 2021 14:44:04 GMT
 Name: jew.mips SHA256: 081228dfb...ce6f03037316 MD5: 0d105f802...2959bfd827ef	Size: 90.21 kB Type: ELF 32-bit MSB execut...	<div>feed:urlhaus mirai ripped:mirai runnable:linux urlhaus:elf urlhaus:mirai</div>	Sun, 11 Apr 2021 14:44:03 GMT

Link to
sample
details

Click on
tags to filter
(or filter out)



Exercise #1.0: Getting familiar with the interface

 Name: jew.mpsl SHA256: 1d2e11bc0...ed53c5a78b3d MD5: 19830e713...e01990b4dc42	Size: 94.21 kB Type: ELF 32-bit LSB execut...	feed:urlhaus ripped:mirai runnable:linux urlhaus:elf urlhaus:mirai mirai	Sun, 11 Apr 2021 14:44:04 GMT
 Name: jew.mpsl SHA256: 1d2e11bc0...ed53c5a78b3d MD5: 19830e713...e01990b4dc42	Size: 94.21 kB Type: ELF 32-bit LSB execut...	feed:urlhaus ripped:mirai runnable:linux urlhaus:elf urlhaus:mirai mirai	Sun, 11 Apr 2021 14:44:04 GMT



tag:"mirai" AND NOT tag:"feed:urlhaus"

10141 results found

Quick query:

[Only uploaded by me](#)

[Exclude public](#)

[Favorites](#)

[Exclude feed:*](#)

[Only ripped:*](#)


[Add +](#)



Exercise #1.0: Getting familiar with the interface

Search bar: Search


Quick query: Only uploaded by me Exclude public Favorites Exclude feed:* Only ripped:* Add +

Name/Hash	Size/Type	Tags	First seen
 Name: 88527fb710478e1c54c6... SHA256: 88527fb71...7fa4102b8411	Size: 3.38 MB Type: Zip archive data, at lea...	runnable:android:apk	Sun, 11 Apr 2021 15:18:08 GMT

Search bar: Search ?

1 results found

Quick query: Only uploaded by me Exclude public Favorites Exclude feed:* Only ripped:* Add +

Name/Hash	Size/Type	Tags	First seen
 Name: 7f3d1f38b49054fa1a3fc4... SHA256: d457da57c... 47c99283d9c1 MD5: 576252... 81672403fd19	Size: 319.73 kB Type: RAR archive data, v4, os...	archive:rar feed:malwarebazaar	Thu, 08 Apr 2021 07:05:21 GMT



Exercise #1.1: Filtering samples by tags

Introduction



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 ⓘ

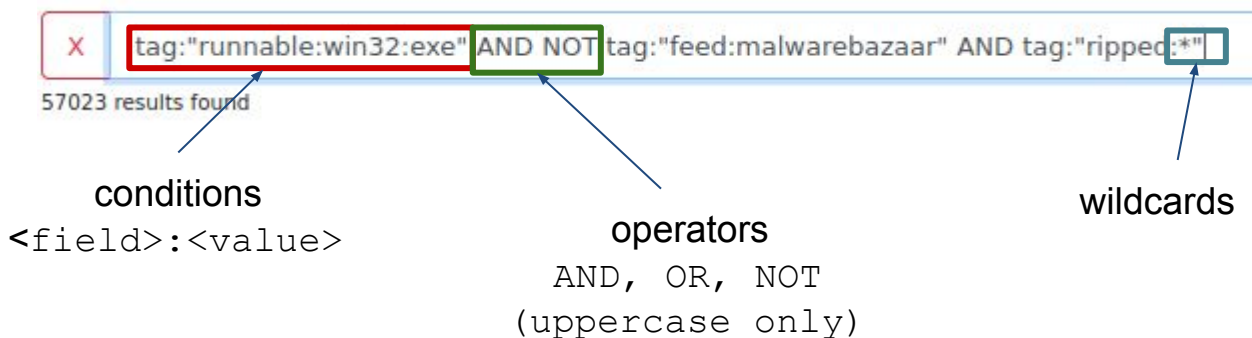
Generic metadata tag with additional information that are useful for filtering

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



Exercise #1.1: Filtering samples by tags

Lucene-based query syntax



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>



Everything related with formbook

<div><div>X</div><div>tag:*formbook*</div><div>Search</div><div>?</div></div>			
8472 results found			
Quick query: <div>Only uploaded by me</div> <div>Exclude public</div> <div>Favorites</div> <div>Exclude feed:*</div> <div>Only ripped:*</div> <div>Add +</div>			
Name/Hash	Size/Type	Tags	First seen
 Name: 3310000_55996a8917b... SHA256: 55996a891...c1b1122ca8d7 MD5: cedc1abf7...2d89abd544fd	Size: 154.75 kB Type: data	<div>dump:win32:exe</div> <div>formbook</div>	Sun, 11 Apr 2021 08:13:01 GMT
 Name: 5f85c45be7b228140ce5... SHA256: b9a1fe9ef...bccca5623704 MD5: 430a47302...5a335d184bc4	Size: 551 kB Type: PE32 executable (GUI) I...	<div>et:formbook</div> <div>feed:malwarebazaar</div> <div>ripped:formbook</div> <div>runnable:win32:exe</div> <div>yara:win_formbook</div>	Sun, 11 Apr 2021 07:59:08 GMT



Exercise #1.1: Filtering samples by tags

Ranges

X size:[10000 TO 15000]

X size:[10kB TO 15kB]

X size:<=10kB

X upload_time:<=2020-01-01

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



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>

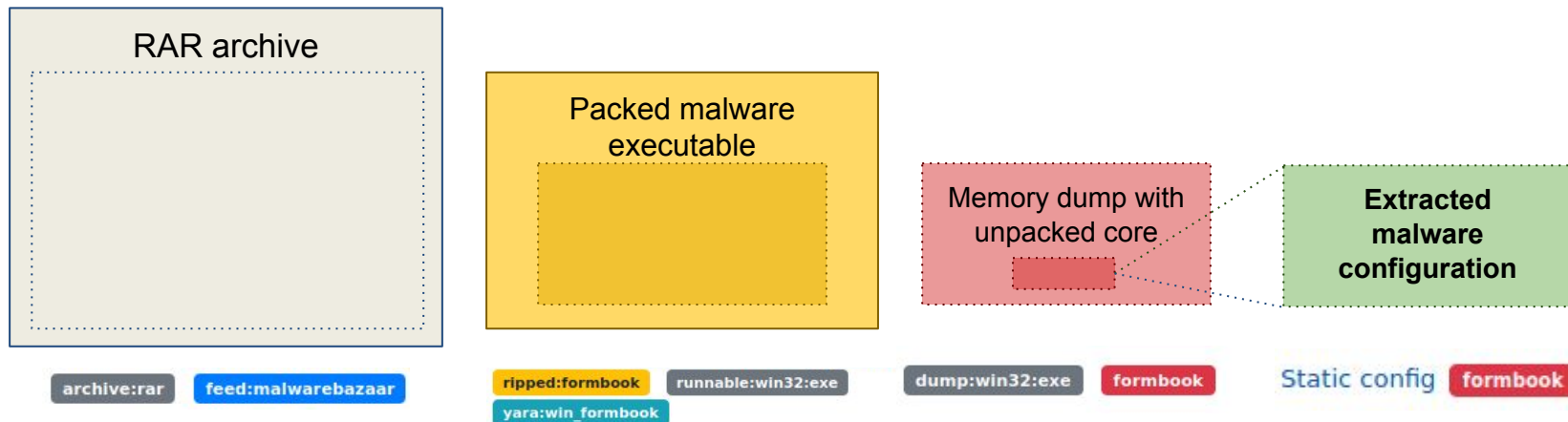


Exercise #1.2: Exploring sample view and hierarchy

Introduction



Exercise #1.2: Exploring sample view and hierarchy



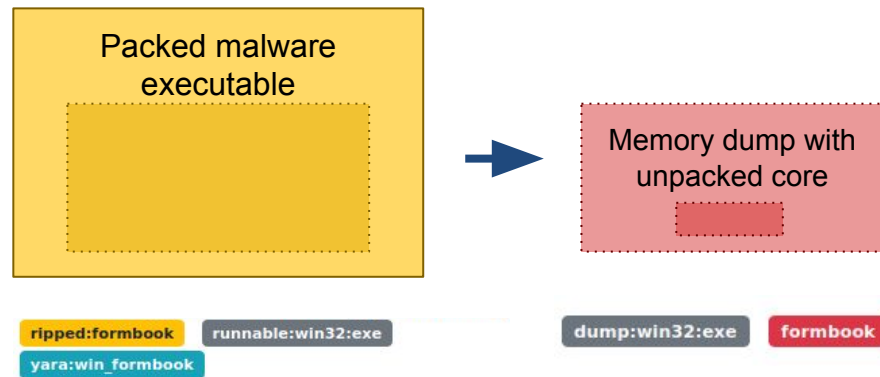
DRAKVUF Sandbox

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.

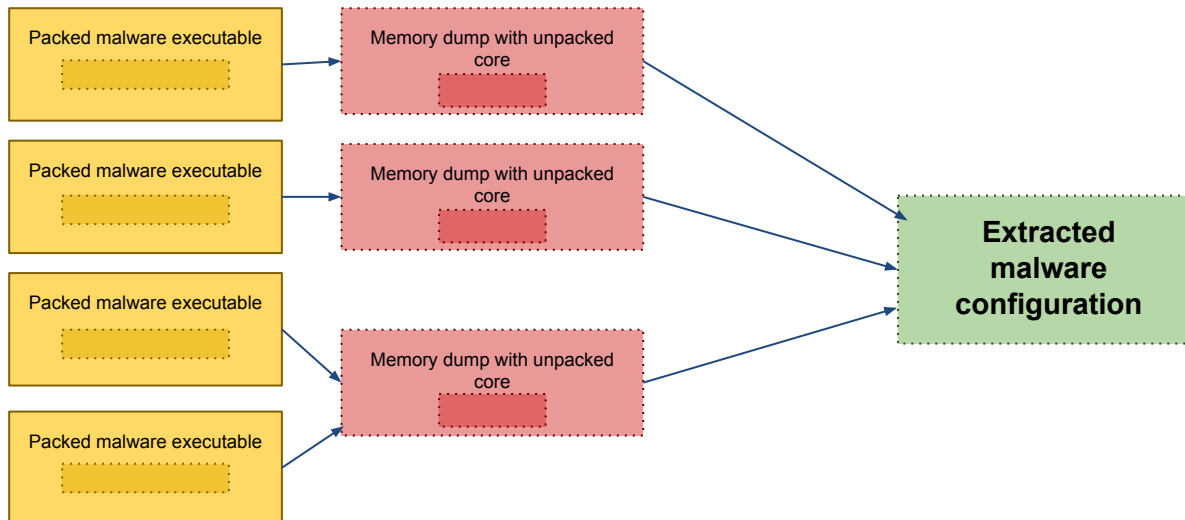
 <https://github.com/CERT-Polska/drakvuf-sandbox>

 <https://github.com/tklengyel/drakvuf>





Exercise #1.2: Exploring sample view and hierarchy





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>



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>

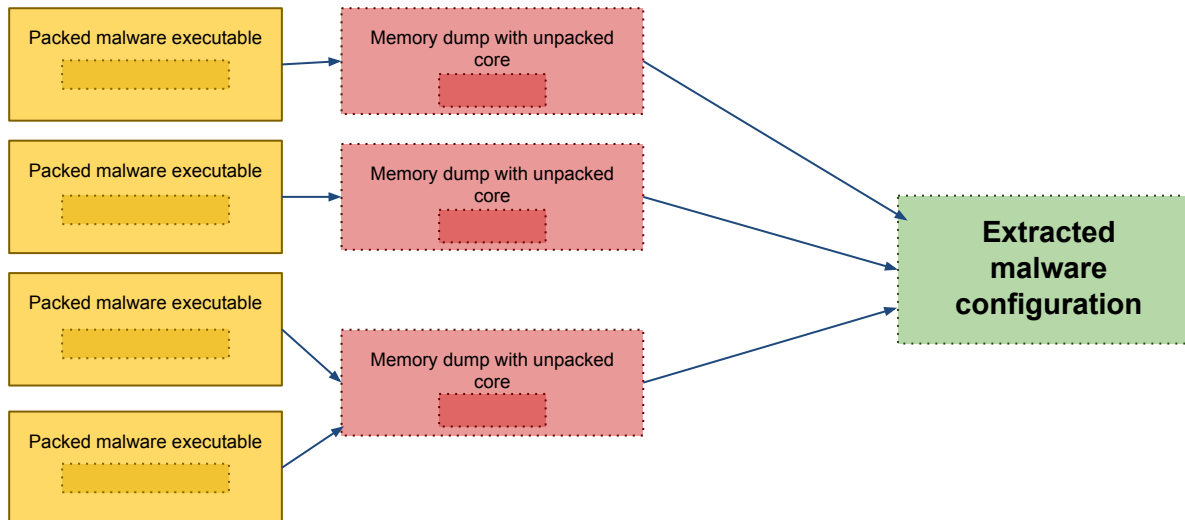


Exercise #1.4: Blobs and dynamic configurations

Introduction

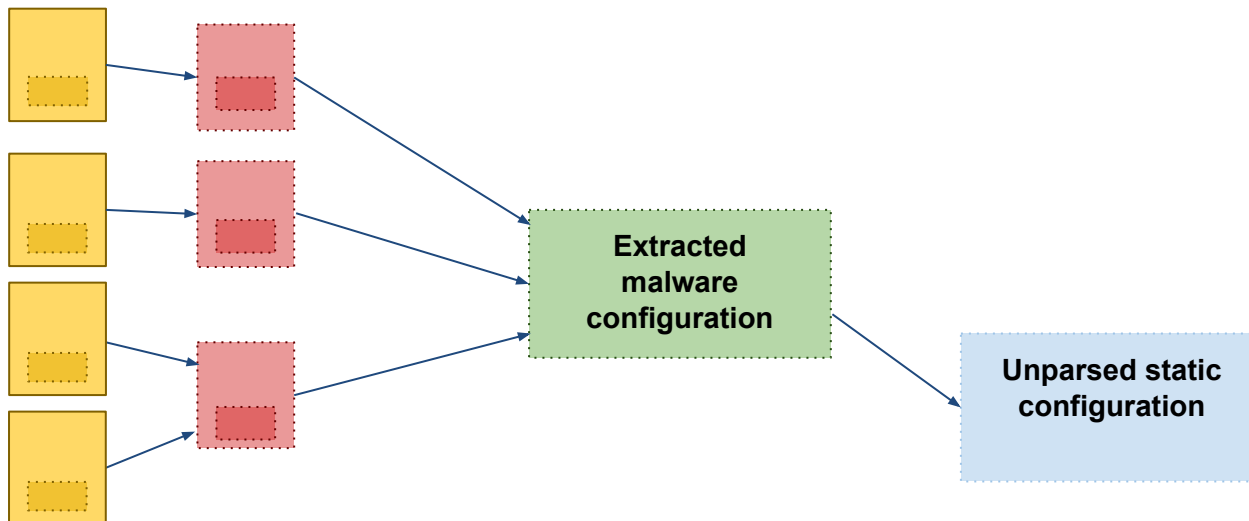


Exercise #1.4: Blobs and dynamic configurations



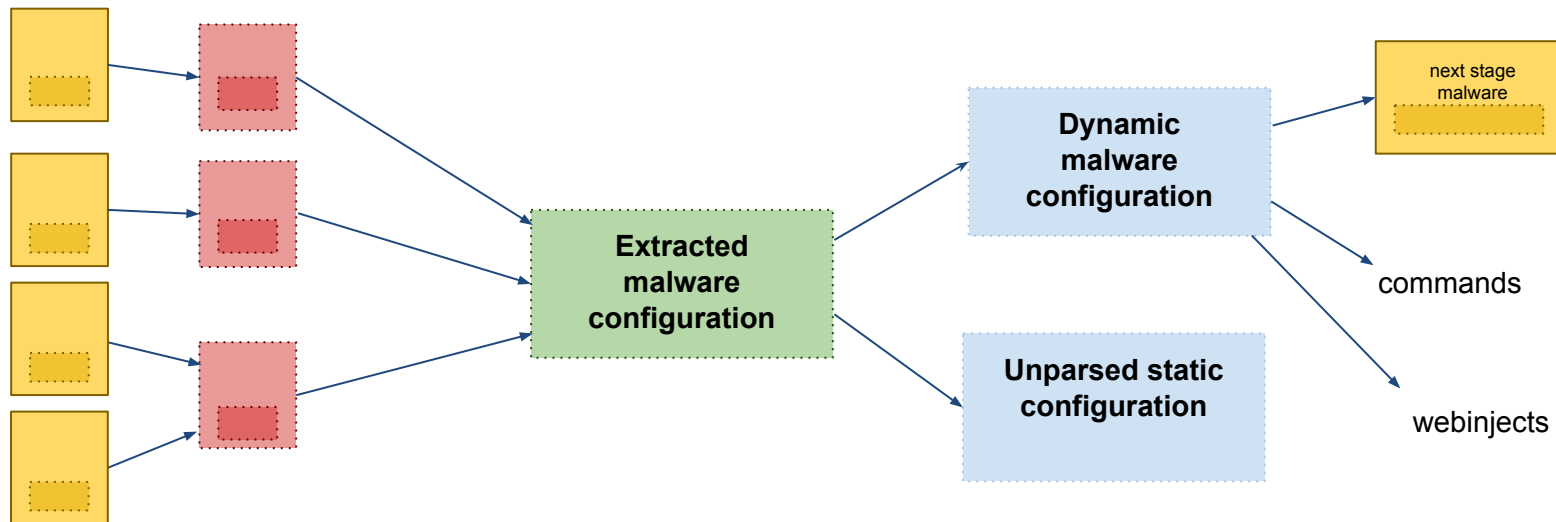


Exercise #1.4: Blobs and dynamic configurations



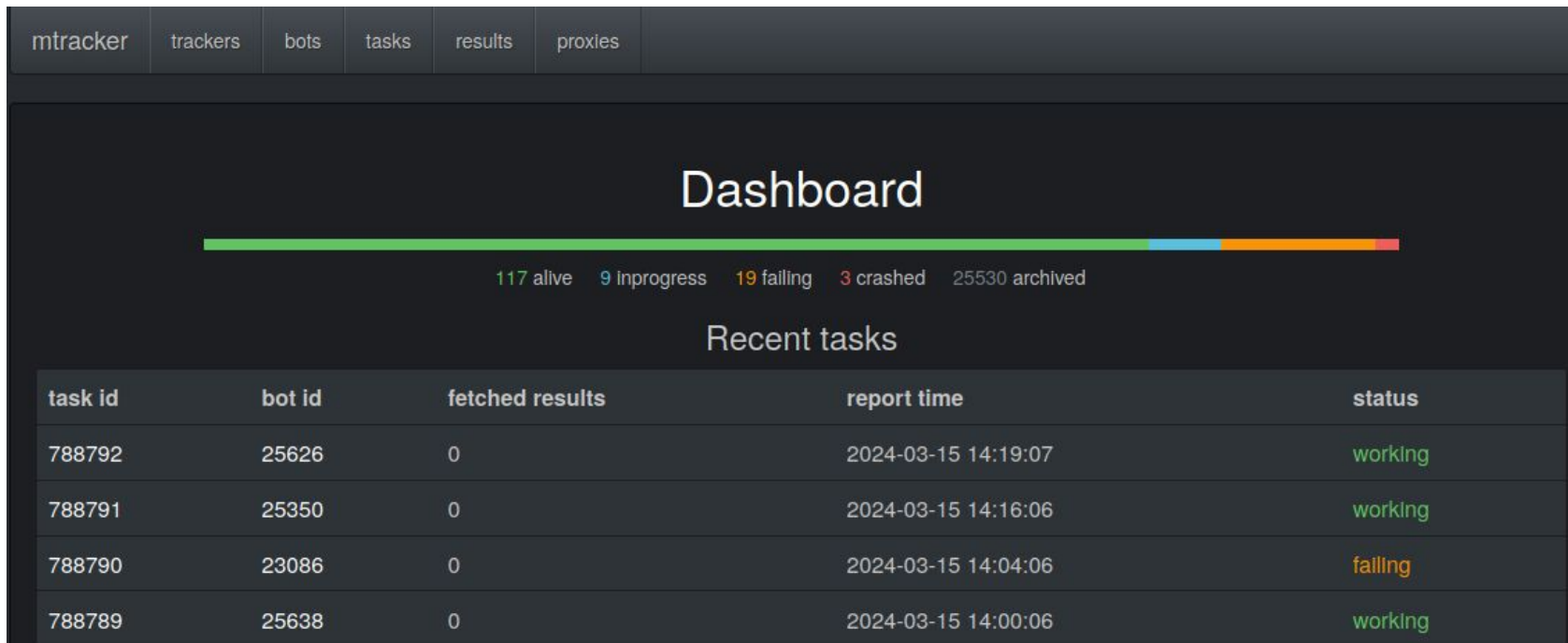


Exercise #1.4: Blobs and dynamic configurations



mtracker

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





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

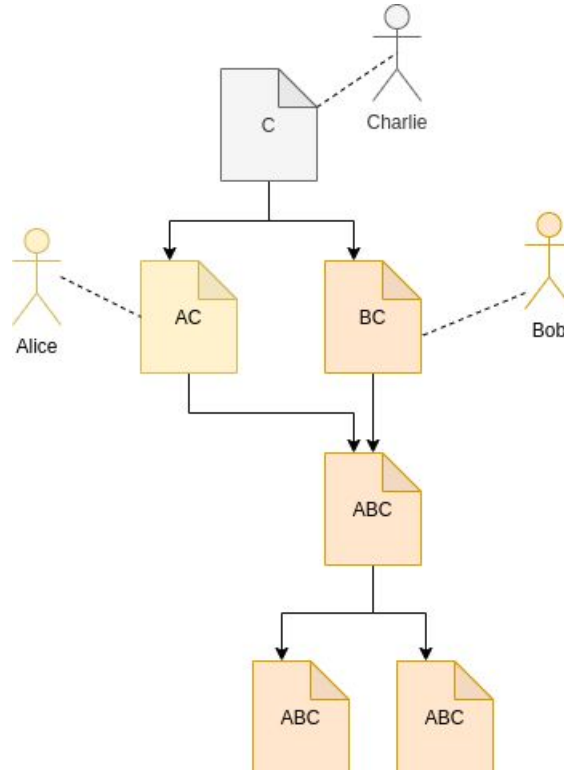


Exercise #1.5: Let's upload something!

Introduction

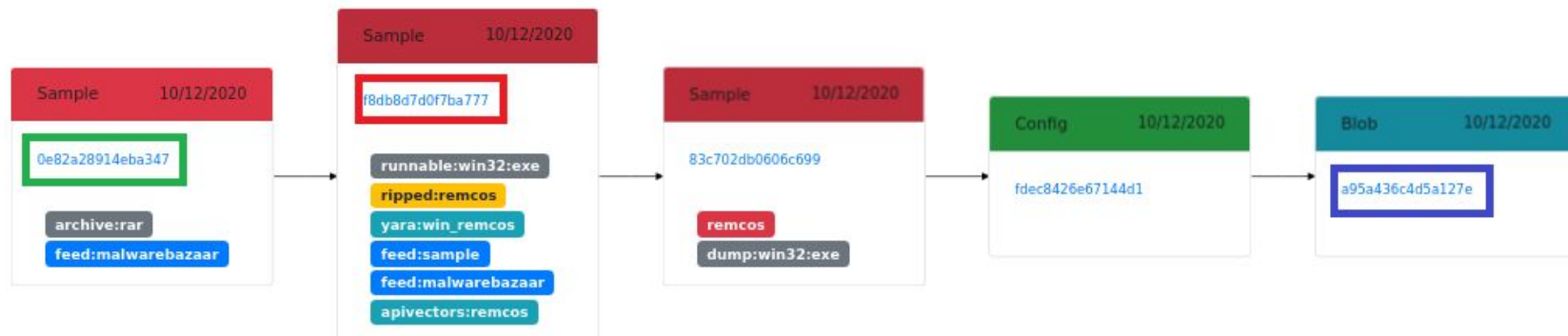


Exercise #1.5: Let's upload something!





Exercise #1.5: Let's upload something!





Exercise #1.5: Let's upload something!



Added by **karton**: **a95a436c4d5a127ed90a9c382476d9bc7c136efa39edbd912ab8935dc10d1b7d**

everything	Mon, 12 Oct 2020 09:23:40 GMT
certpl-systems	Mon, 12 Oct 2020 09:23:40 GMT
karton (uploader)	Mon, 12 Oct 2020 09:23:40 GMT

Added by **abusech**: **0e82a28914eba347b95549eb4b7c5d642c2ee6f747407b3a0d8c8c8ff988c215**

public	Mon, 12 Oct 2020 09:23:40 GMT
abusech (uploader)	Mon, 12 Oct 2020 09:23:40 GMT

Added by **cert-sissden**: **f8db87d0f7ba777f5f18452579f40a224a055c6624066b5dce501121cc91c5c**

cert-sissden (uploader)	Tue, 13 Oct 2020 01:19:55 GMT
-------------------------	-------------------------------



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-lets-upload-something>

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**
 - Distributed collaboration with mwdb **remotes**
 - Advanced programming techniques with **malduck**



YOU ARE
HERE

mwdb.cert.pl



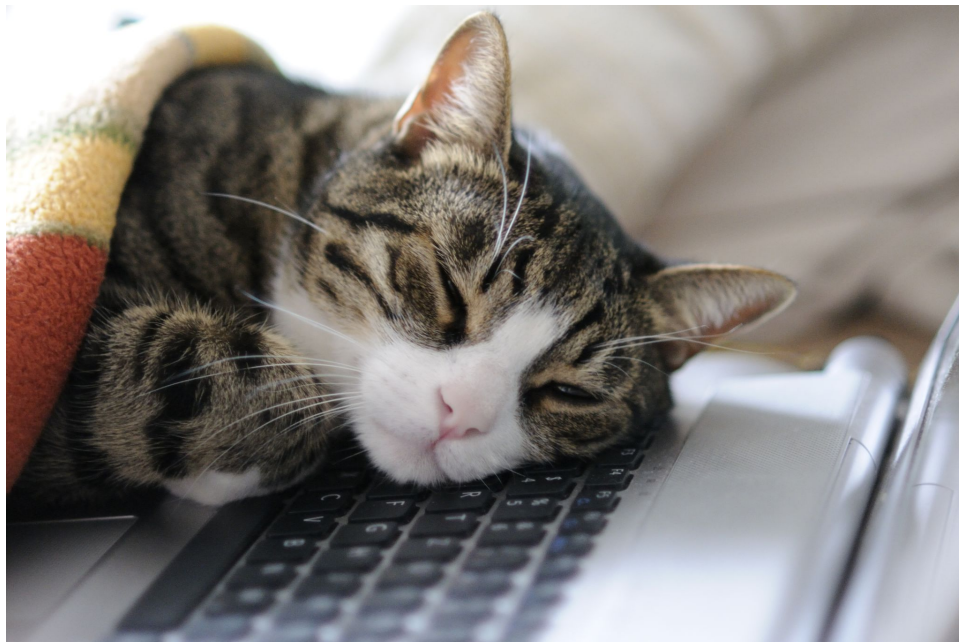
Coffee break

Prepare for later:

```
$ git clone \  
https://github.com/CERT-Polska/karton-playground.git
```

```
$ cd karton-playground
```

```
$ docker-compose pull  
(or “docker compose pull”)  
(it will take a while)
```



#2. Scripting and automation with **mwdblib**



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](#)



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/>



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?



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)?



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>

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**



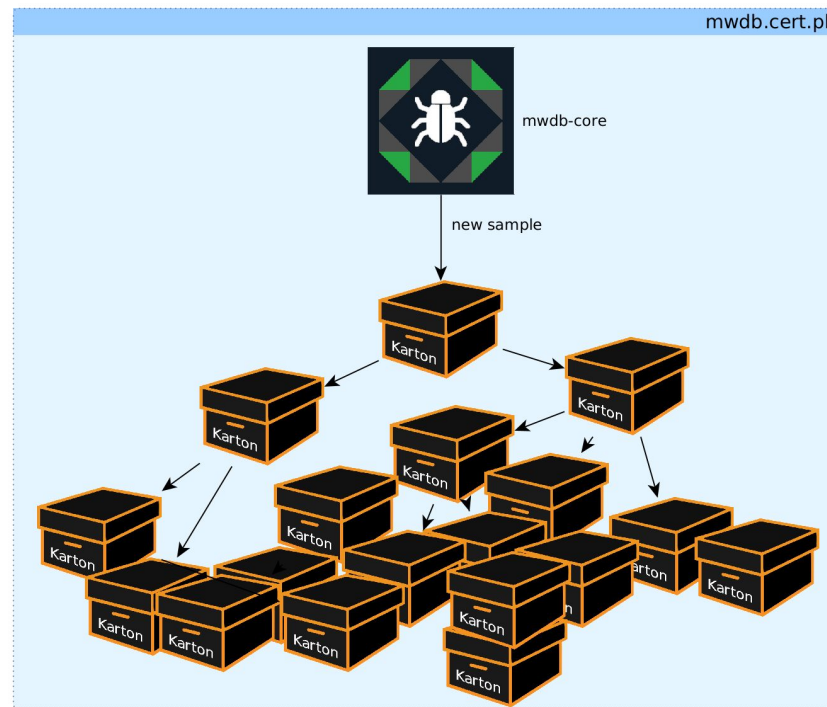
Run a **self-hosted** mwdb-core and kartaon instance



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

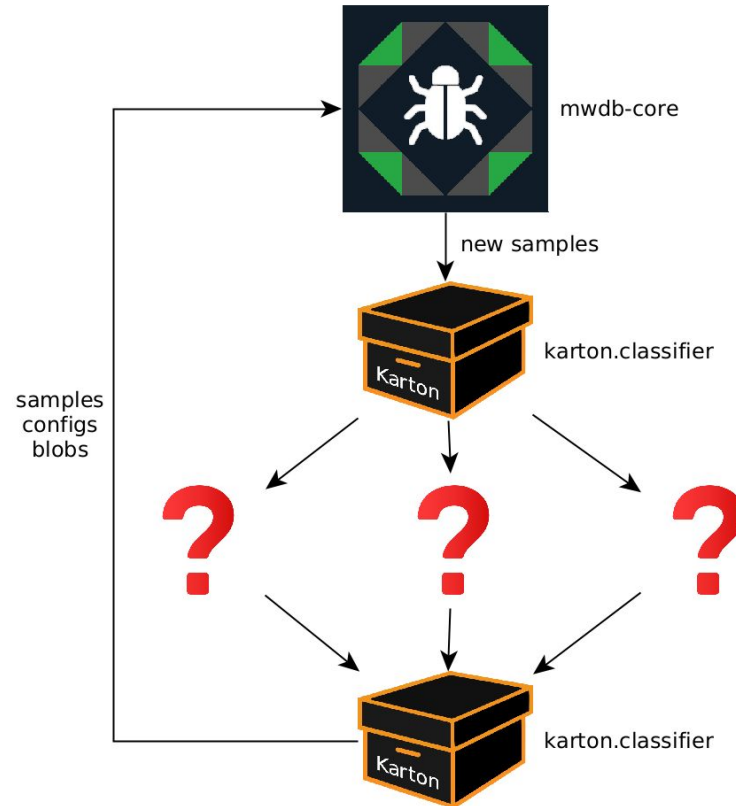
Karton Playground

```
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

Karton Playground

While you wait...



Karton pipeline

karton.classifier

3.0.0

kind:raw

type:sample



Karton pipeline

karton.extractor

2.x.x

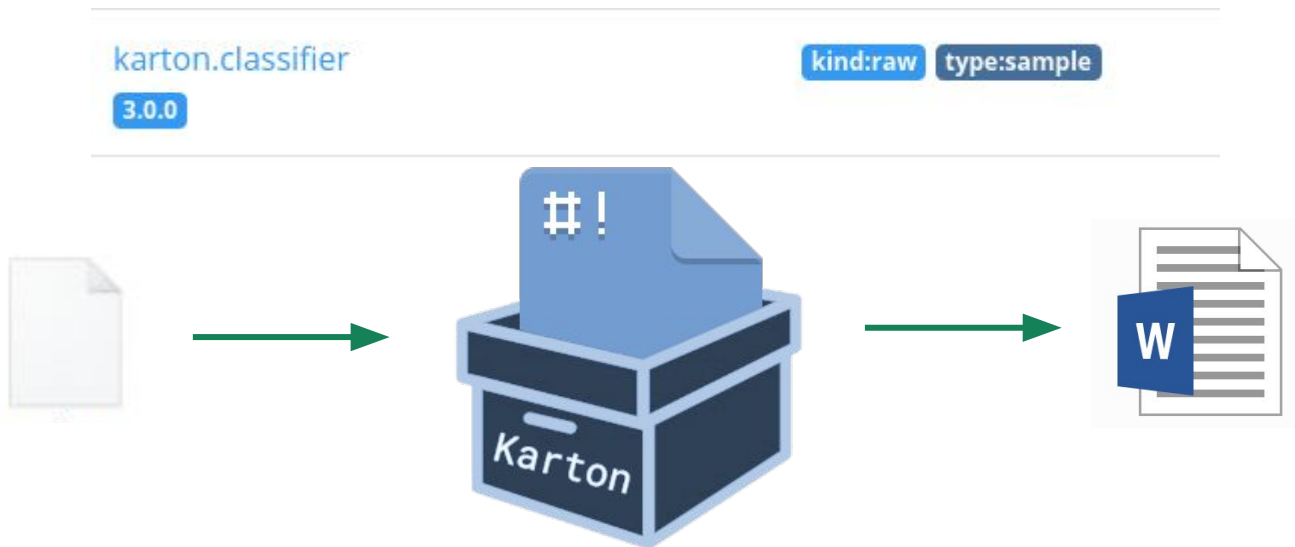
kind:archive

stage:recognized

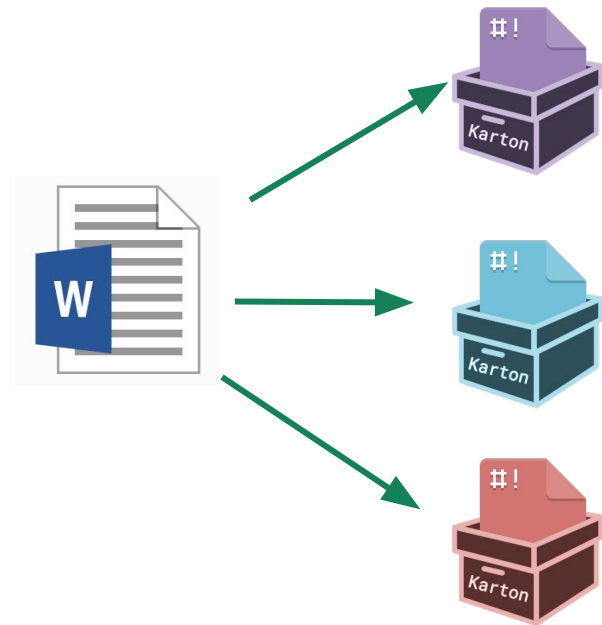
type:sample



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

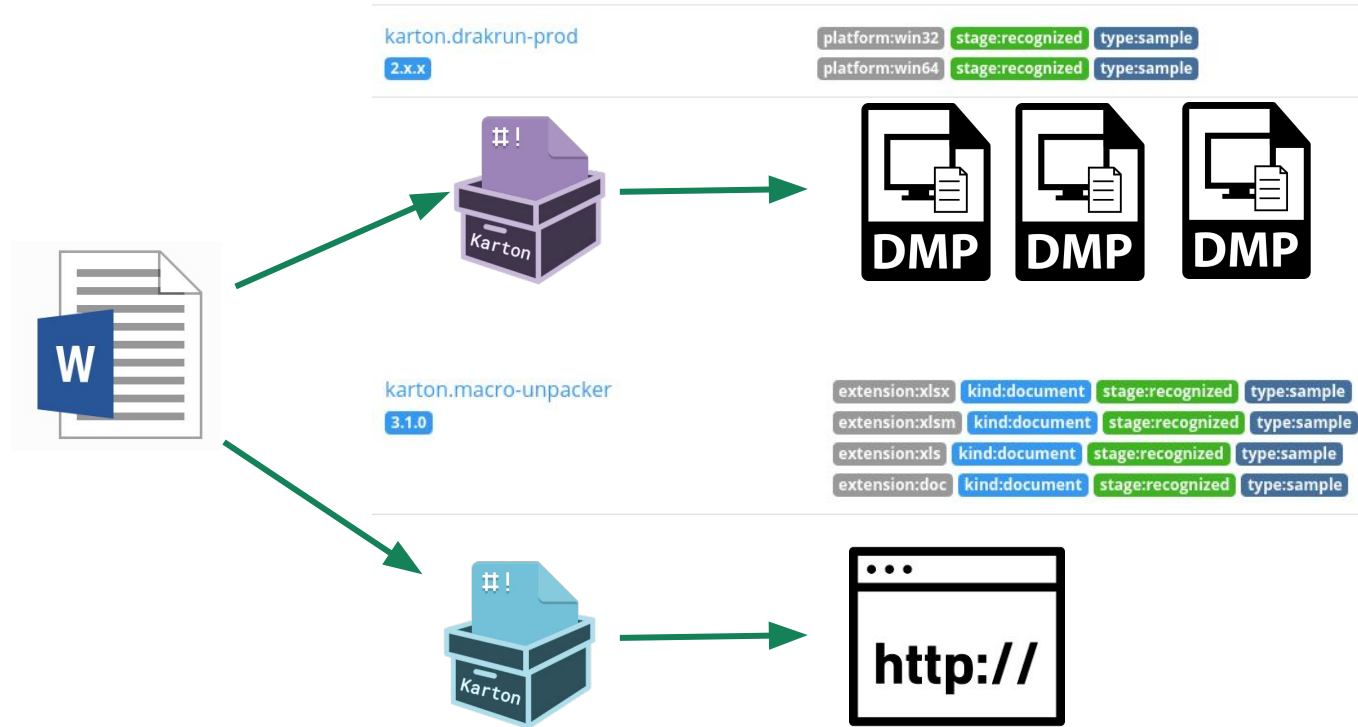
karton.emodoc

2.x.x

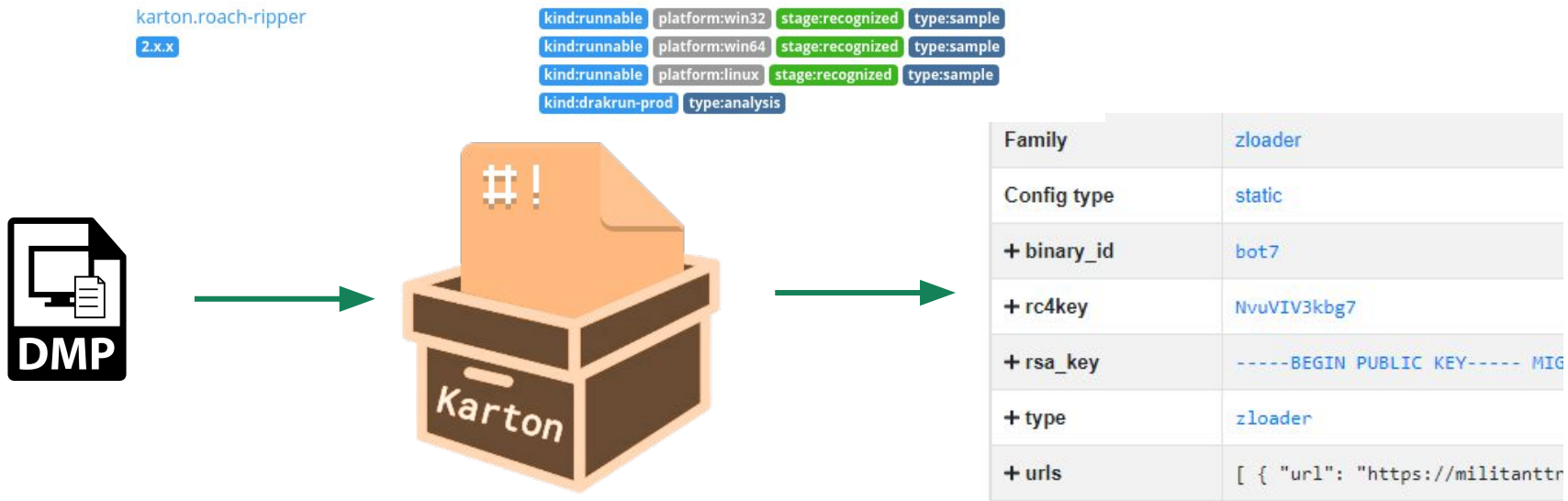
extension:doc kind:document stage:recognized type:sample

extension:docx kind:document stage:recognized type:sample

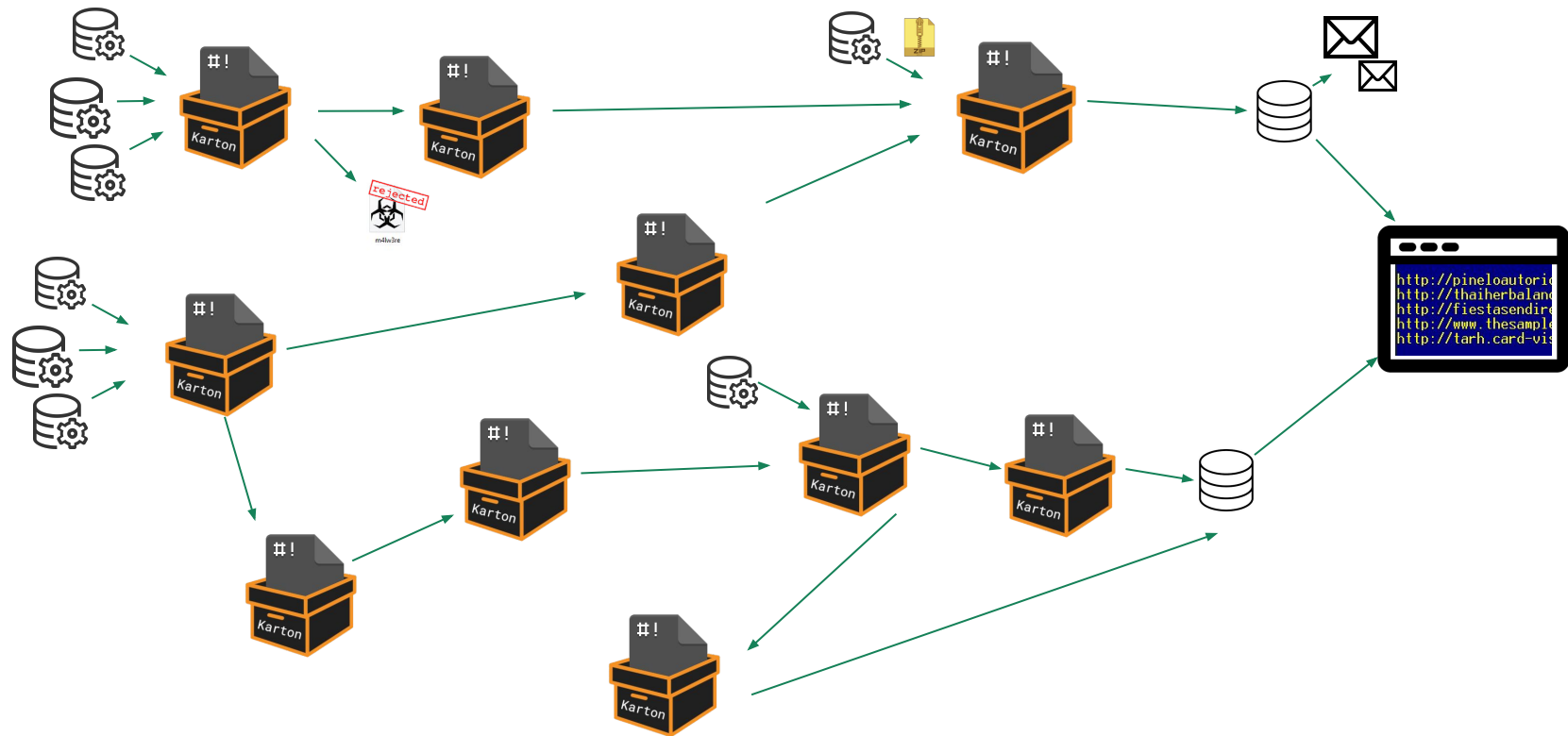
Karton pipeline



Karton pipeline





Karton pipeline in the real world



Karton Playground: click around

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

 mwdb 

You need to authenticate before accessing this page.

Login

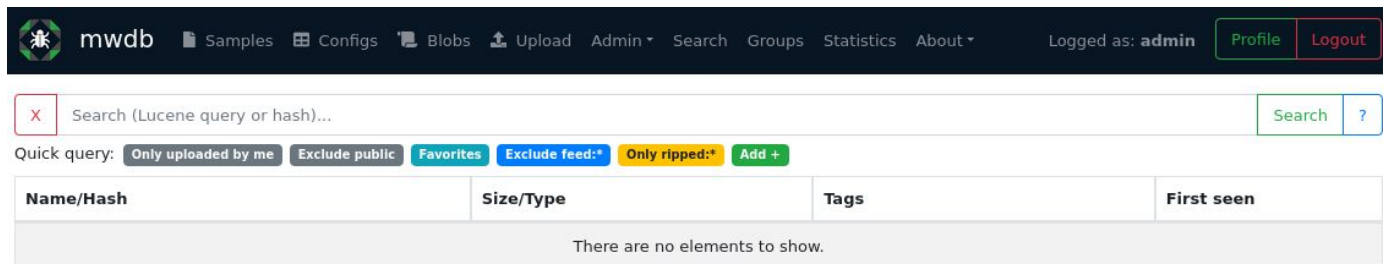
Login

Password

[Forgot password?](#)

Karton Playground: click around

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



The screenshot shows the mwdb web interface. The top navigation bar includes links for Samples, Configs, Blobs, Upload, Admin, Search, Groups, Statistics, and About. The user is logged in as 'admin' and has buttons for Profile and Logout. Below the navigation bar is a search bar with a placeholder 'Search (Lucene query or hash)...' and a 'Search' button. Under the search bar, there are quick query filters: 'Only uploaded by me', 'Exclude public', 'Favorites', 'Exclude feed:*', 'Only ripped:*', and 'Add +'. Below the filters is a table with the following columns: Name/Hash, Size/Type, Tags, and First seen. The table is currently empty, displaying the message 'There are no elements to show.'

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

Karton Playground: click around

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

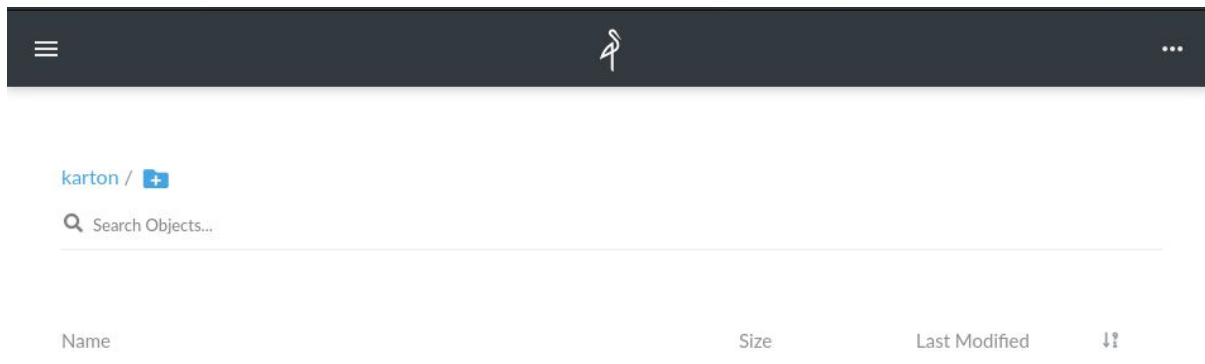
karton

binds

identity	filters	tasks	errors	replicas
karton.classifier 4.0.4	kind:raw type:sample	0	0	1
karton.mwdb-reporter 4.0.4	stage:recognized type:sample stage:analyzed type:sample type:config	0	0	1

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>

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
a4816d4fec6d6d2806d5b105c3aab55f4a1eb5deb3b126f317093a4dc4aab88a1 autoit-malware.bin
```

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

```
$ kartern-autoit-ripper
[2021-04-11 17:19:57,867][INFO] Service kartern.autoit-ripper started
/home/msm/Projects/kartern-playground/venv/lib/python3.8/site-packages/kartern/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
```

Karton Playground exercise

Volia!



Tags

runnable:win32:exe x

Add tag Add

Related samples + Add

child 5689d35ddaac5435facbf144e82d91c1f568db5f6adcd41a995ddb89f9ba33f7

Attributes + Add

Karton analysis

✓ done 8b859823-6a49-45e5-9039-a51fa8e6a915 ▼

+ reanalyze

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>

```
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`

```
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**?

```
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 it is interested in
- Producer indicates the kind of produced task

```
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 = Consumer + Producer

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

```
class Karton(Consumer, Producer):
```

```
    """
```

```
    This glues together Consumer and Producer - which is the most common use case
```

```
    """
```

```
    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

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

```
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
...
```

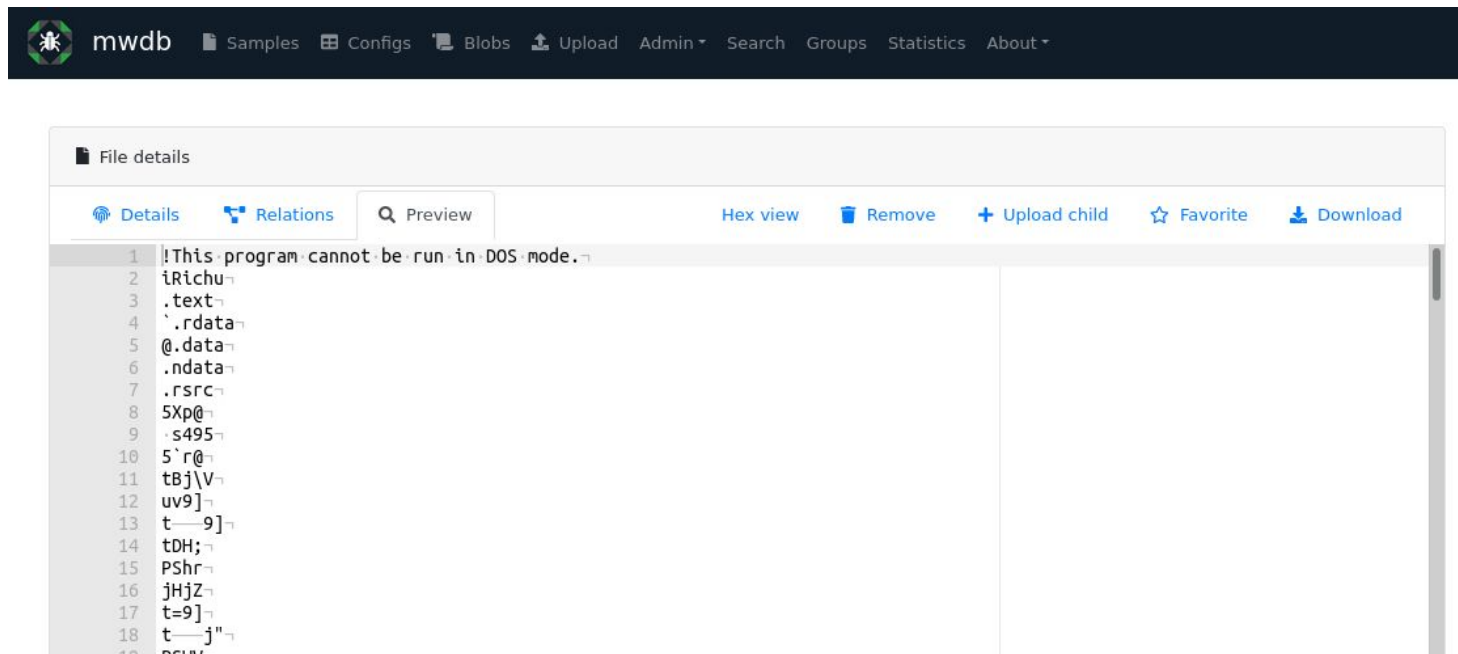
Karton Playground exercise: solution

```
$ python3 karton-template.py
[2021-04-14 20:56:28,927][INFO] Service karton.first started
/home/msm/Projects/karton-playground/venv/lib/python3.8/site-packages/karton/core
warnings.warn("There is no active log consumer to receive logged messages.")
[2021-04-14 20:56:28,928][INFO] Binding on: {'type': 'sample', 'stage': 'recogniz
[2021-04-15 08:45:10,546][INFO] Received new task - c17c9659-49d6-444c-b208-f00fc
[2021-04-15 08:45:10,547][INFO] Hi setup_1.0.61.exe, let me analyse you!
[2021-04-15 08:45:11,100][INFO] Task done - c17c9659-49d6-444c-b208-f00fcd36bc5b
```

File details	
<div><div>Details</div><div>Relations</div><div>Preview</div><div>Remove</div><div>+ Upload child</div><div>Favorite</div><div>Download</div></div>	
File name	setup_1.0.61.exe
File size	15.33 MB
File type	PE32 executable (GUI) Intel 80386, for MS Windows, Nullsoft Installer self-extracting archive
md5	c042ee96b54c8c121165cc1b5e338fe7

Tags	
<div>runnable:win32:exe X</div>	
Add tag	<div>Add</div>
Related samples <div>Add</div>	
child	3eec0da26f66d1ae53d594d52adc3b50a596ed79e8701d130f52870c46a1d709

Karton Playground exercise: solution



Coffee break



Malduck 🦆

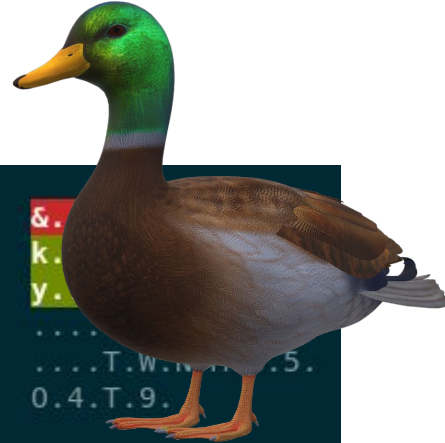
```
00000000 26 00 00 00 73 00 6b 00 79 00 72 00 6f 00 63 00
00000010 6b 00 65 00 74 00 2e 00 6f 00 6f 00 67 00 75 00
00000020 79 00 2e 00 63 00 6f 00 6d 00 c7 0b 00 00 00 00
00000030 00 00 00 00 00 00 00 00 00 00 00 00 d0 07 00 00
00000040 14 00 00 00 54 00 57 00 4e 00 4d 00 55 00 35 00
00000050 4f 00 34 00 54 00 39 00
```

```
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

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
}
```

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
}
```

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
}
```

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 "StartCryptDesktop"
 - 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.

Wrapping 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

Wrapping 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

Wrapping 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



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