Build Your Own Malware Analysis Pipeline Using New Open Source Tools

Jarosław Jedynak
Michał Praszmo
Paweł Srokosz

CERT.PL

hack.lu
18th October 2023

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**

Prerequisites

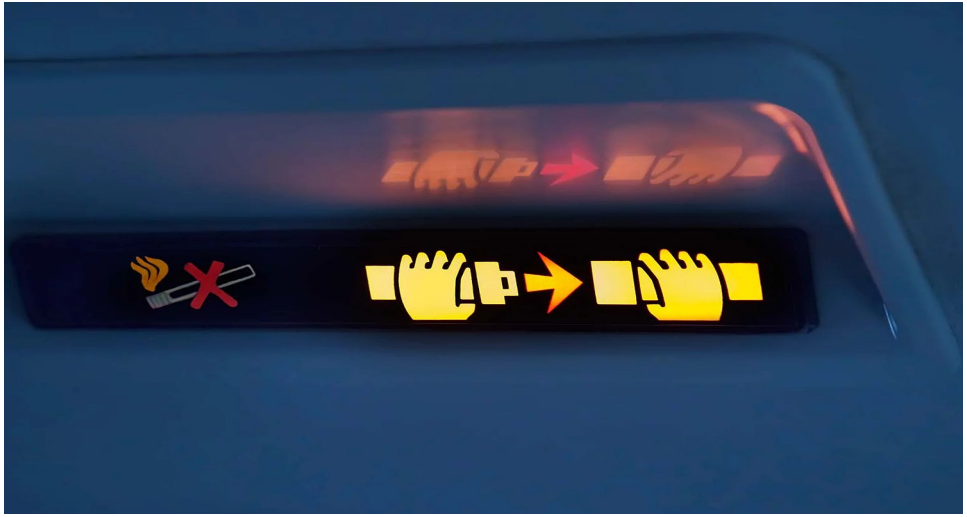
Open a terminal and check if these tools are installed:

- `$ python3 -m pip`
- `$ git`
- `$ docker-compose`
  https://docs.docker.com/engine/install/ubuntu/
  https://docs.docker.com/compose/install/

Bookmark this URL: https://training-mwdb.readthedocs.io/

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

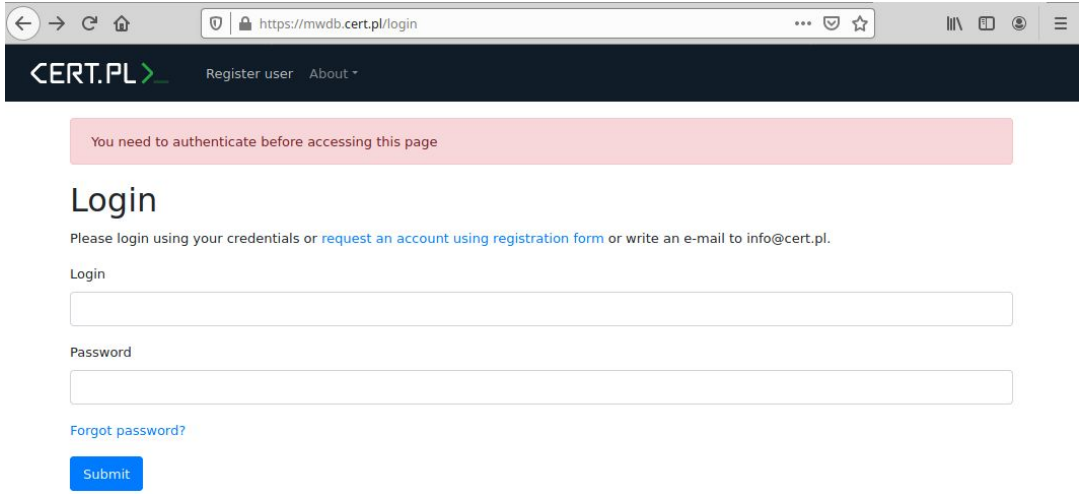
https://mwdb.cert.pl



mwdb.cert.pl

Login with your individual credentials

(Check your inbox for confirmation email)
if you do not have credentials let us know now



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

mwdb.cert.pl

Exercise #1.0: Getting familiar with the interface

Link to sample details

Click on tags to filter (or filter out)

CERT.PL Samples Configs Blobs Upload Yara search Search Groups Statistics About Logged as: demo Profile Logout

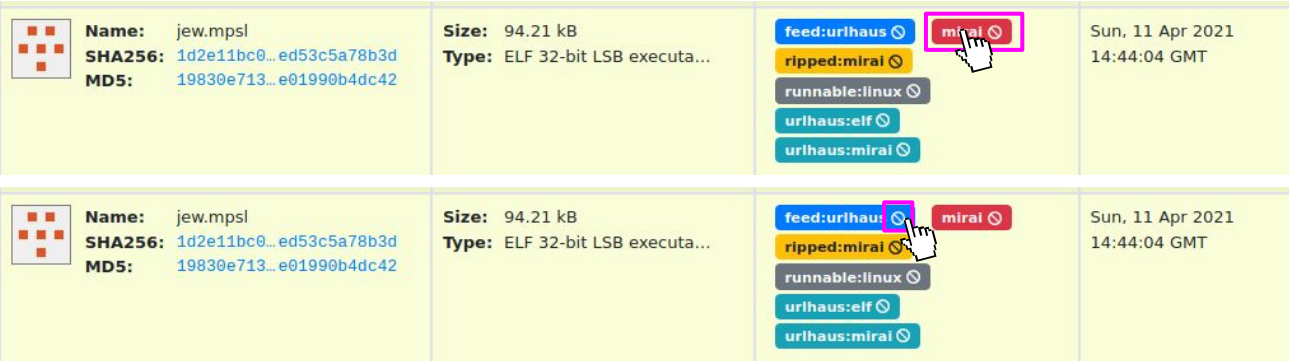
X Search (Lucene query or hash)... Search ?

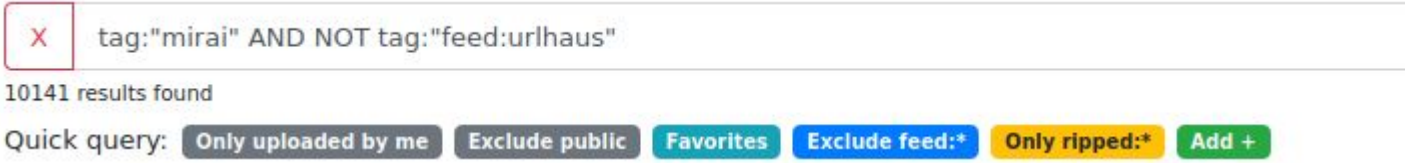
Quick query: Only uploaded by me | Exclude public | Favorites | Exclude feed:* | Only ripped:* | Add +

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…	feed:urlhaus runnable:win32:exe urlhaus:exe urlhaus:finderbot urlhaus:opendir	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 executa…	feed:urlhaus mirai ripped:mirai runnable:linux urlhaus:elf urlhaus:mirai	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 executa…	feed:urlhaus mirai ripped:mirai runnable:linux urlhaus:elf urlhaus:mirai	Sun, 11 Apr 2021 14:44:03 GMT

mwdb.cert.pl

Exercise #1.0: Getting familiar with the interface





mwdb.cert.pl

Exercise #1.0: Getting familiar with the interface

X 5762523a60685aafa8a681672403fd19 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

X md5:5762523a60685aafa8a681672403fd19 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

mwdb.cert.pl

Exercise #1.1: Filtering samples by tags

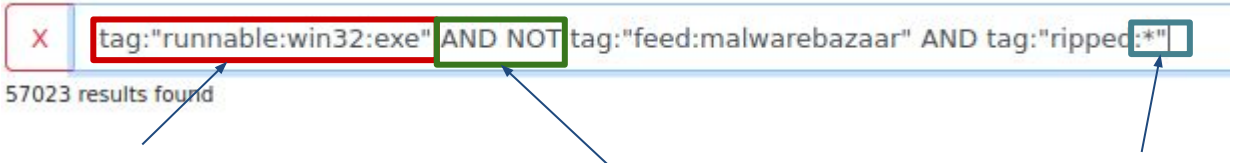
formbook	Simple tag, mostly used for marking artifacts that are associated with malware family
feed:sample	Tag describing the source of malware sample
ripped:formbook	Tag for the original sample, indicating recognized malware family
runnable:win32:exe	Tag describing the type of sample
yara:win_formbook	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

mwdb.cert.pl

Exercise #1.1: Filtering samples by tags

Lucene-based query syntax



conditions

`<field>:<value>`

operators

`AND, OR, NOT`

`(uppercase only)`

wildcards

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

mwdb.cert.pl

Everything related with formbook

X | tag:*formbook* | Search | ?

8472 results found

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

Name/Hash	Size/Type	Tags	First seen
Name: 3310000_55996a8917b... **SHA256:** 55996a891...c1b1122ca8d7 **MD5:** cedc1abf7...2d89abd544fd	**Size:** 154.75 kB **Type:** data	dump:win32:exe formbook	Sun, 11 Apr 2021 08:13:01 GMT
Name: 5f85c45be7b228140ce5... **SHA256:** b9a1fe9ef...bccca5623704 **MD5:** 430a47302...5a335d184bc4	**Size:** 551 kB **Type:** PE32 executable (GUI) I...	et:formbook feed:malwarebazaar ripped:formbook runnable:win32:exe yara:win_formbook	Sun, 11 Apr 2021 07:59:08 GMT

mwdb.cert.pl

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:

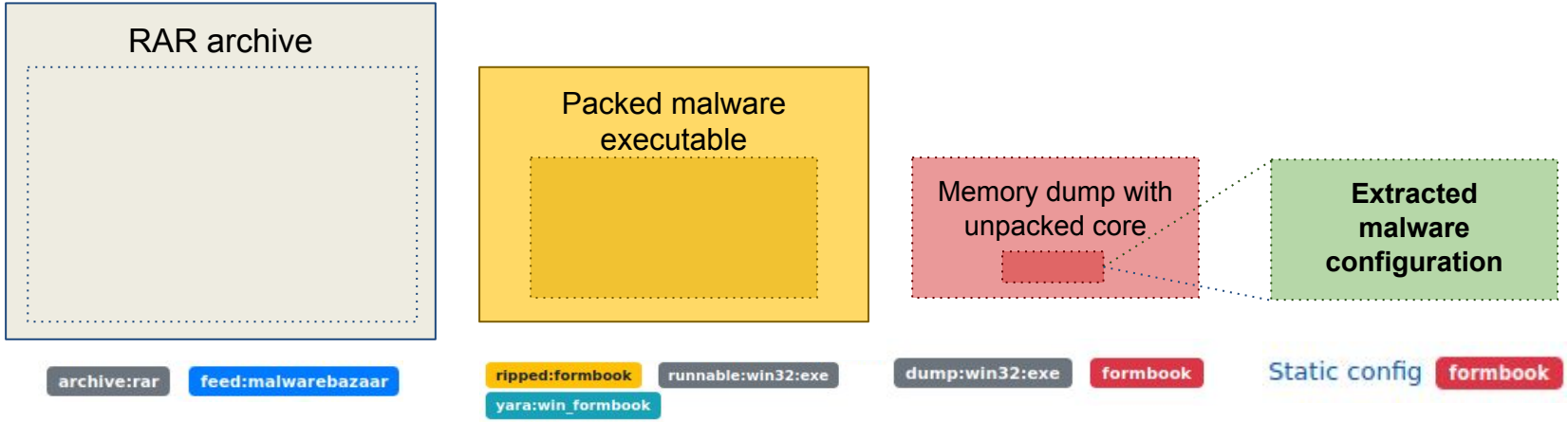
- https://training-mwdb.readthedocs.io/en/latest/part-1.html#exercise-1-1-filtering-samples-by-tags

Exercise #1.2: Exploring sample view and hierarchy

Introduction

mwdb.cert.pl

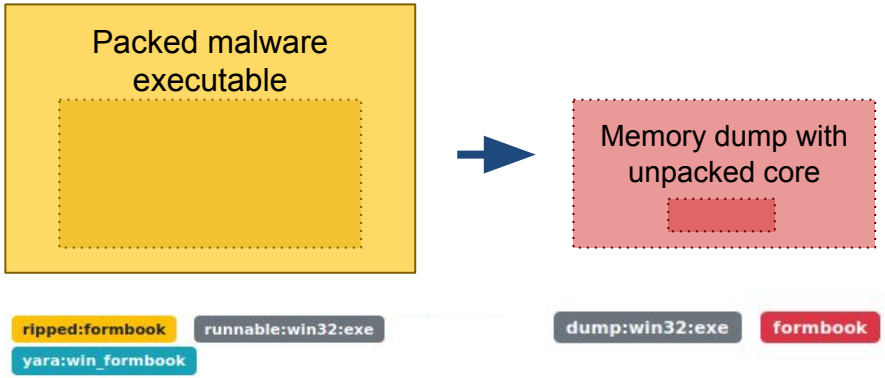
Exercise #1.2: Exploring sample view and hierarchy



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.

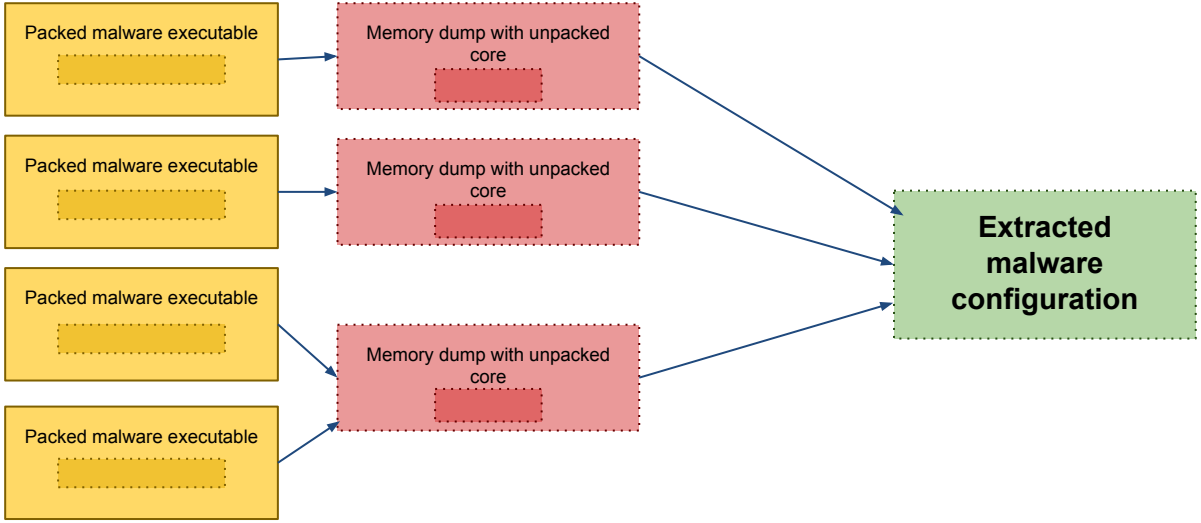


https://github.com/CERT-Polska/drakvuf-sandbox

https://github.com/tklengyel/drakvuf

mwdb.cert.pl

Exercise #1.2: Exploring sample view and hierarchy



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:

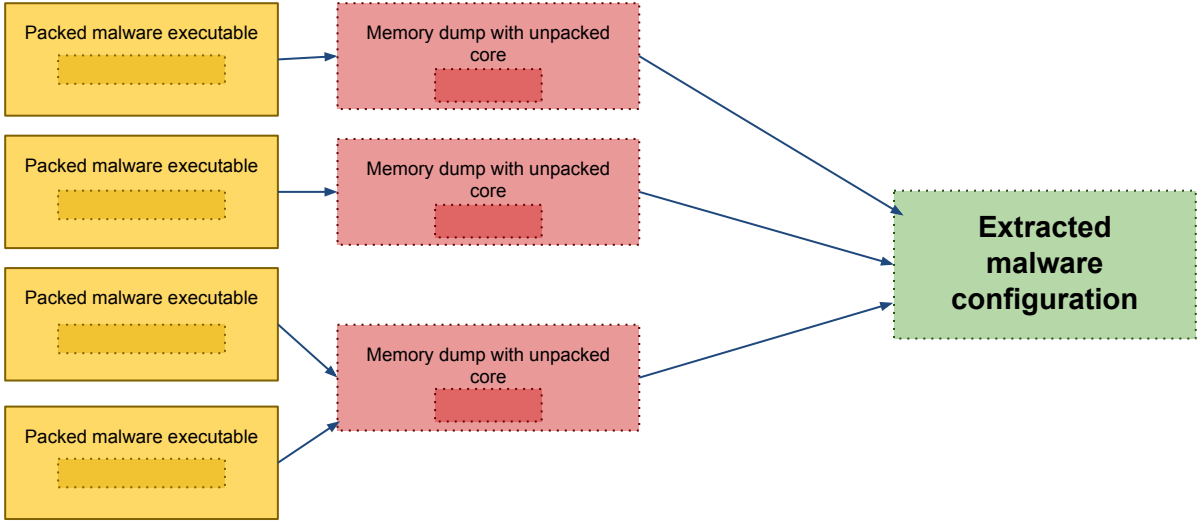
- https://training-mwdb.readthedocs.io/en/latest/part-1.html#exercise-1-3-looking-for-similar-configurations

Exercise #1.4: Blobs and dynamic configurations

Introduction

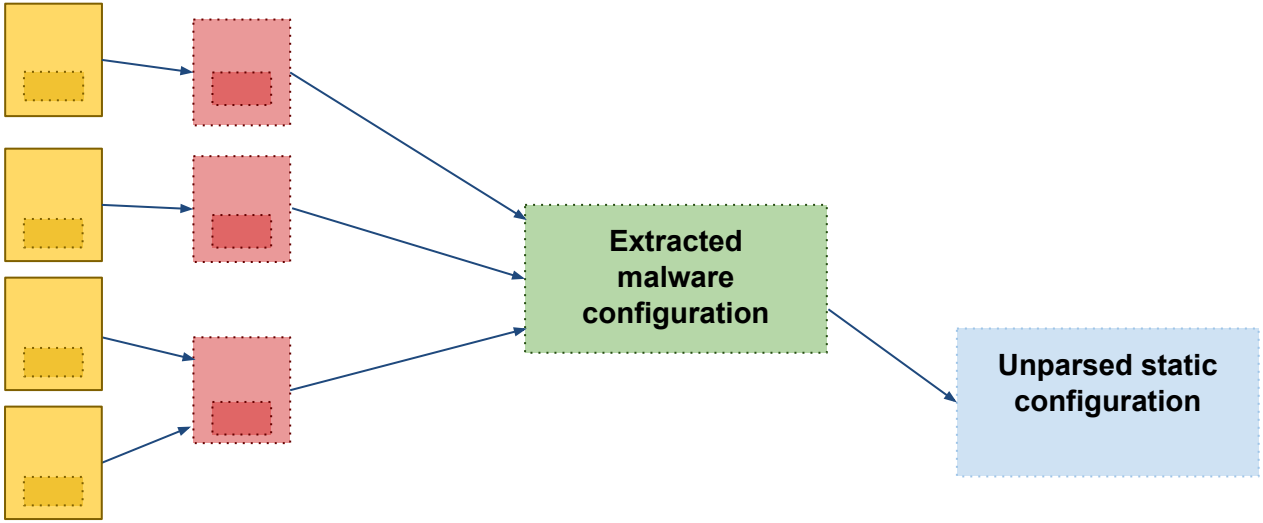
mwdb.cert.pl

Exercise #1.4: Blobs and dynamic configurations



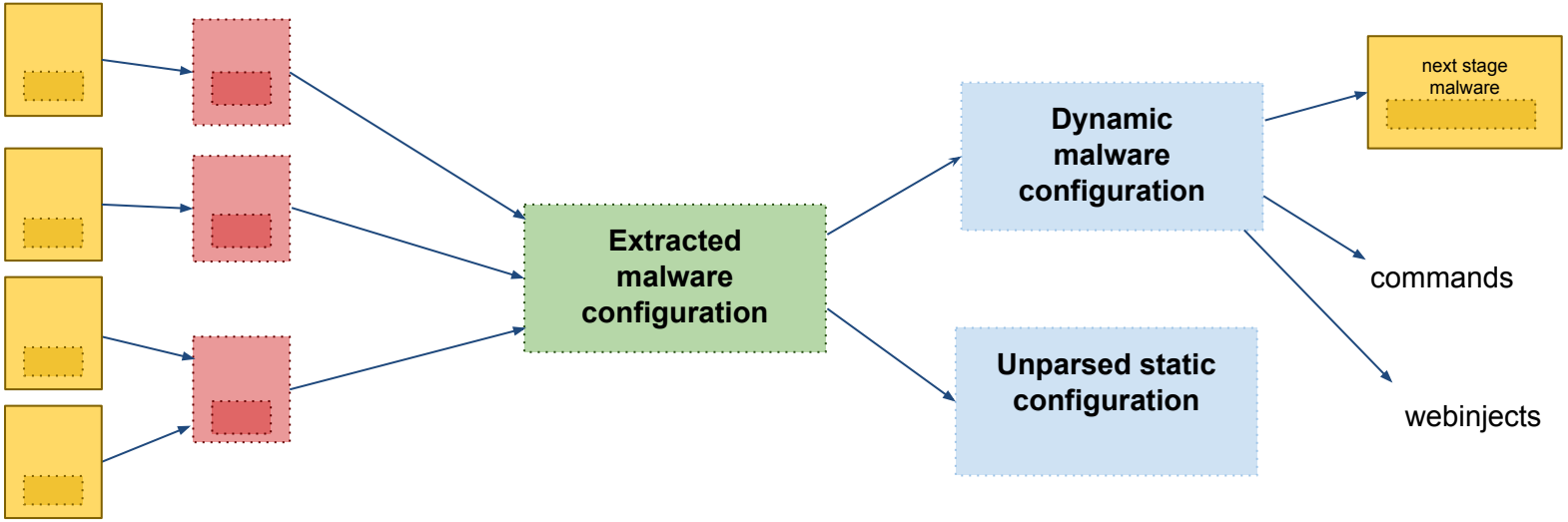
mwdb.cert.pl

Exercise #1.4: Blobs and dynamic configurations



mwdb.cert.pl

Exercise #1.4: Blobs and dynamic configurations



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:

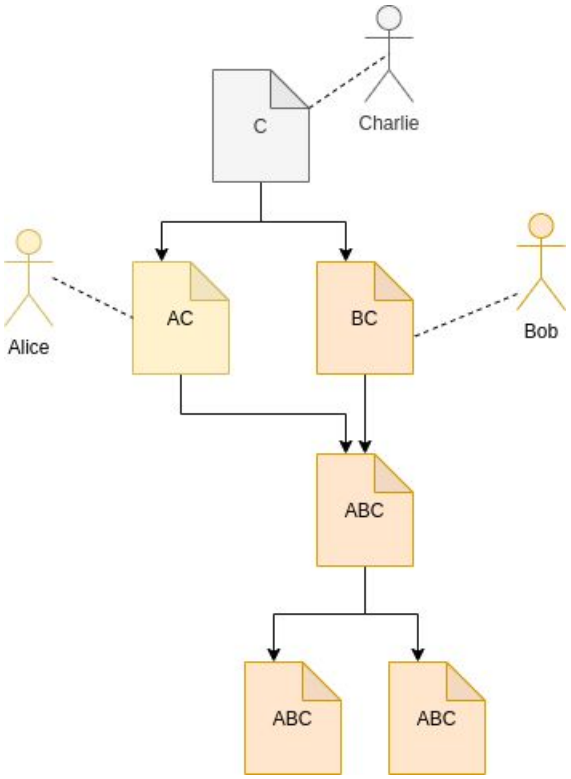
- https://training-mwdb.readthedocs.io/en/latest/part-1.html#exercise-1-4-blobs-and-dynamic-configurations

Exercise #1.5: Let’s upload something!

Introduction

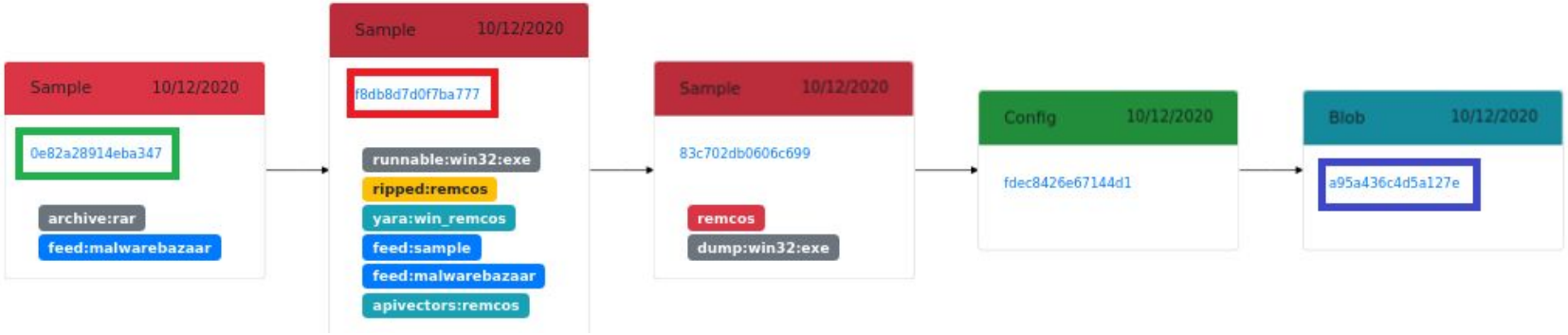
mwdb.cert.pl

Exercise #1.5: Let’s upload something!



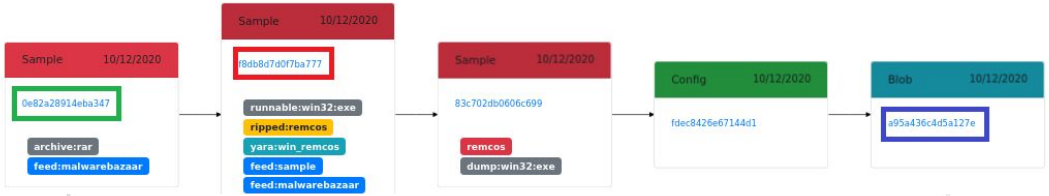
mwdb.cert.pl

Exercise #1.5: Let’s upload something!



mwdb.cert.pl

Exercise #1.5: Let’s upload something!



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: `f8db8d7d0f7ba777f5f18452579f40a224a055c6624066b5dce501121cc91c5c`

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

CERT.PL

Exercise #1.5: Let’s upload something

Goals: Learn how object sharing and access inheritance work.

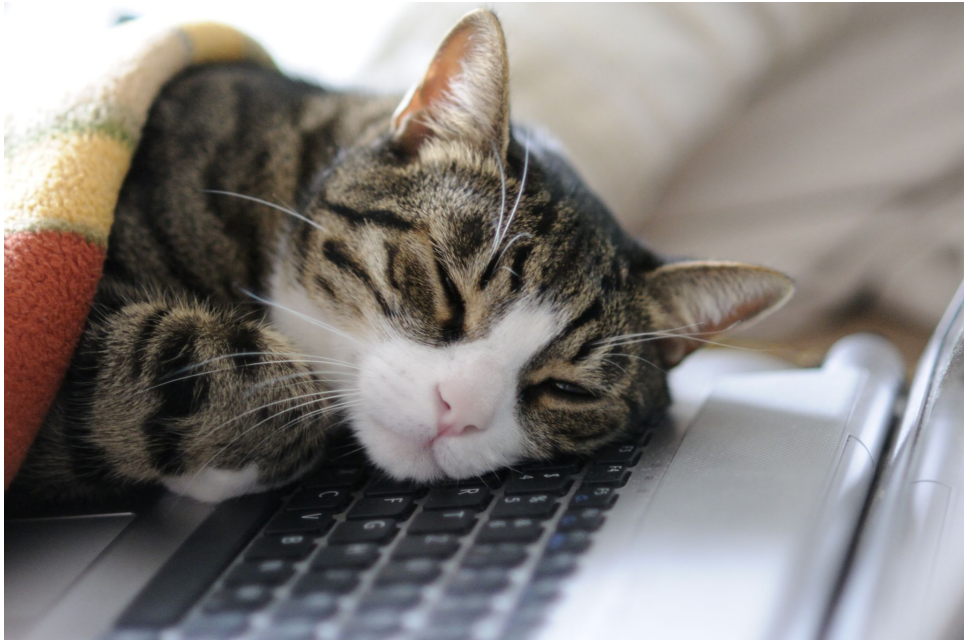
- Download malware sample from https://github.com/CERT-Polska/training-mwdb/raw/main/ex5malware.zip
- Upload to MWDB and check Shares tab
- Go to the child sample. What shares tab shows?

Materials:

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

mwdb.cert.pl

Coffee break



Prepare for later:

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

$ cd karton-playground

$ docker-compose pull

(it can take a long 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) alsohas 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.

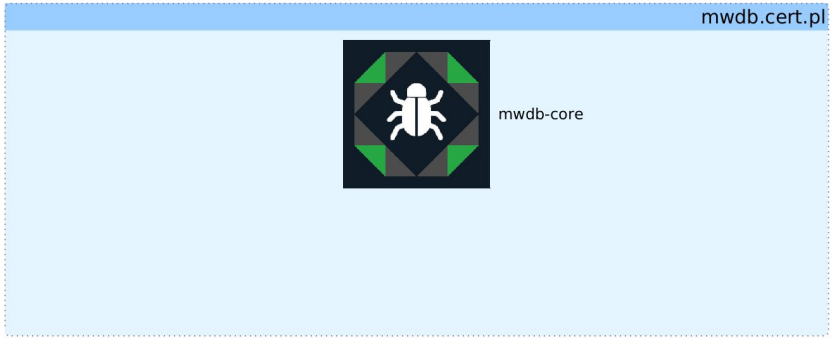
Agenda

- **mwdb.cert.pl**
  - What the heck is **MWD**
  - 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**

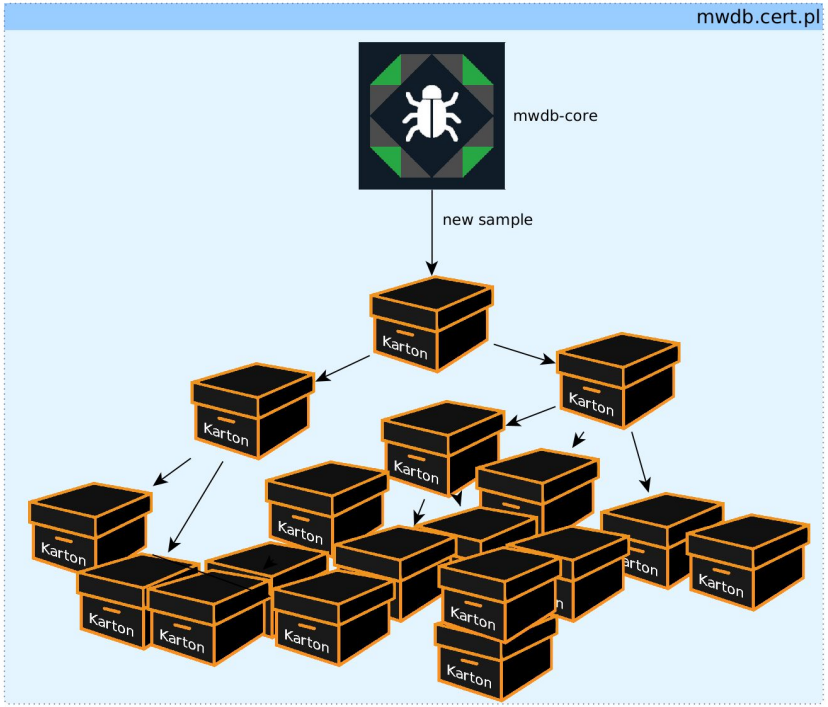
Learn karton with the **karton-playground**

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

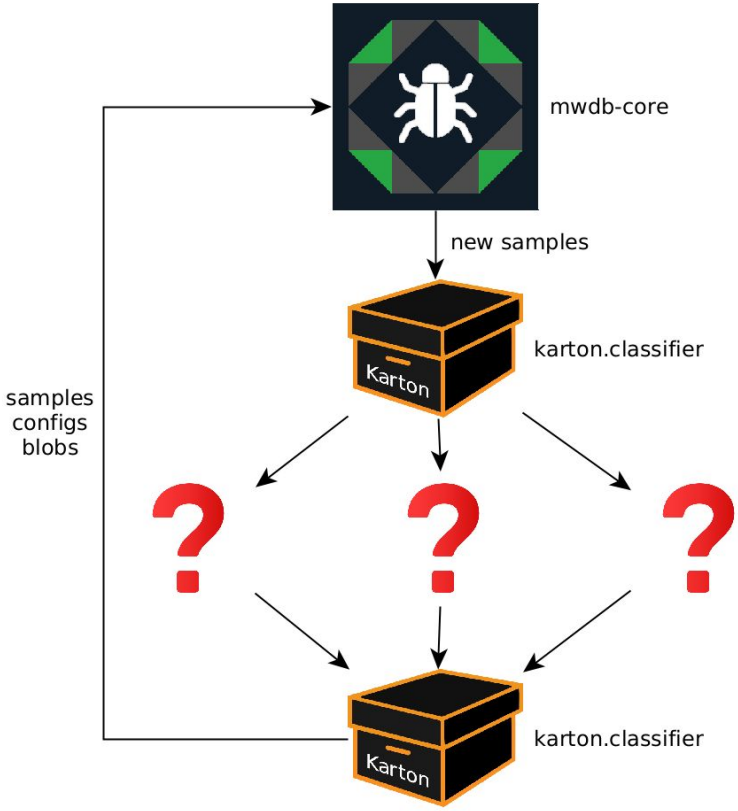
Karton Playground

```
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

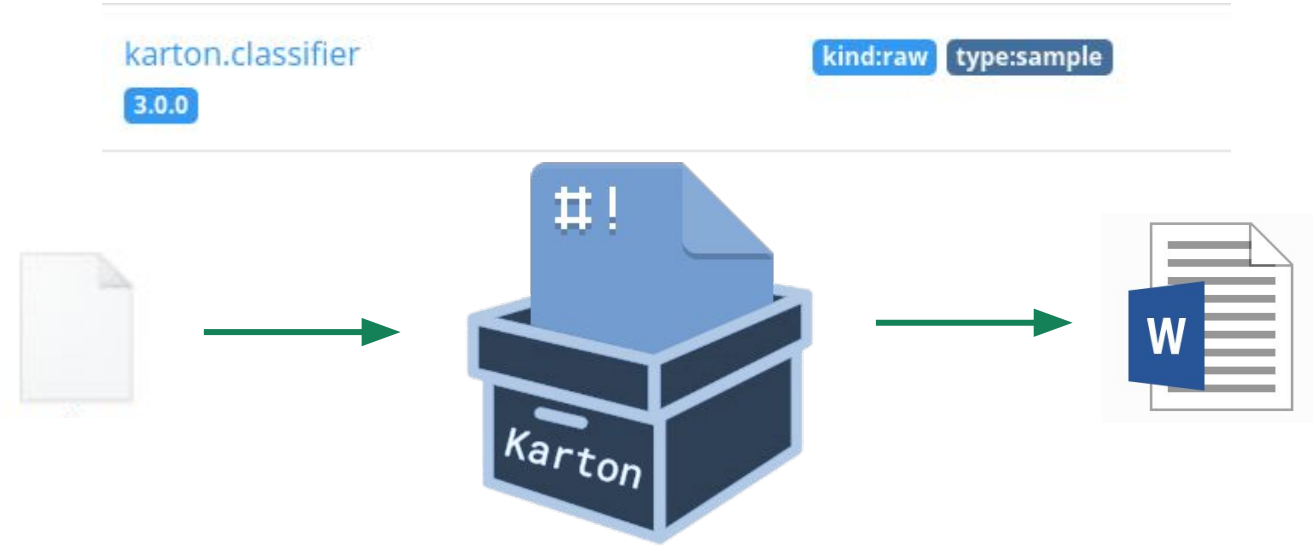
Karton pipeline

karton.extractor

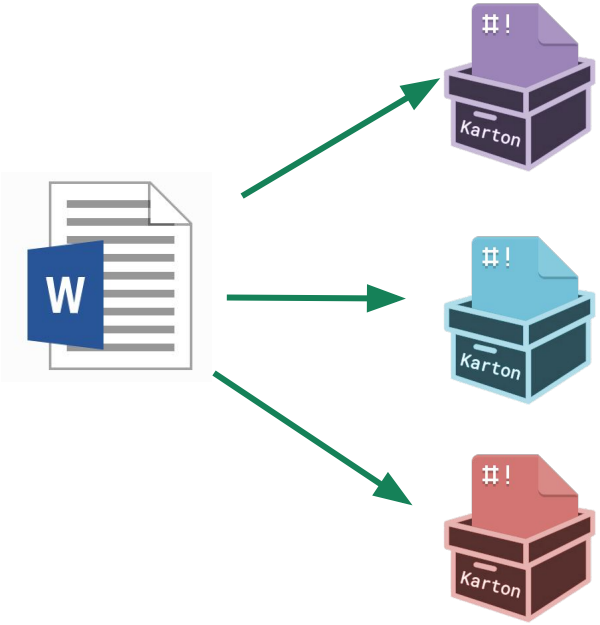
2.x.x

kind:archive stage:recognized type:sample

Karton pipeline



Karton pipeline



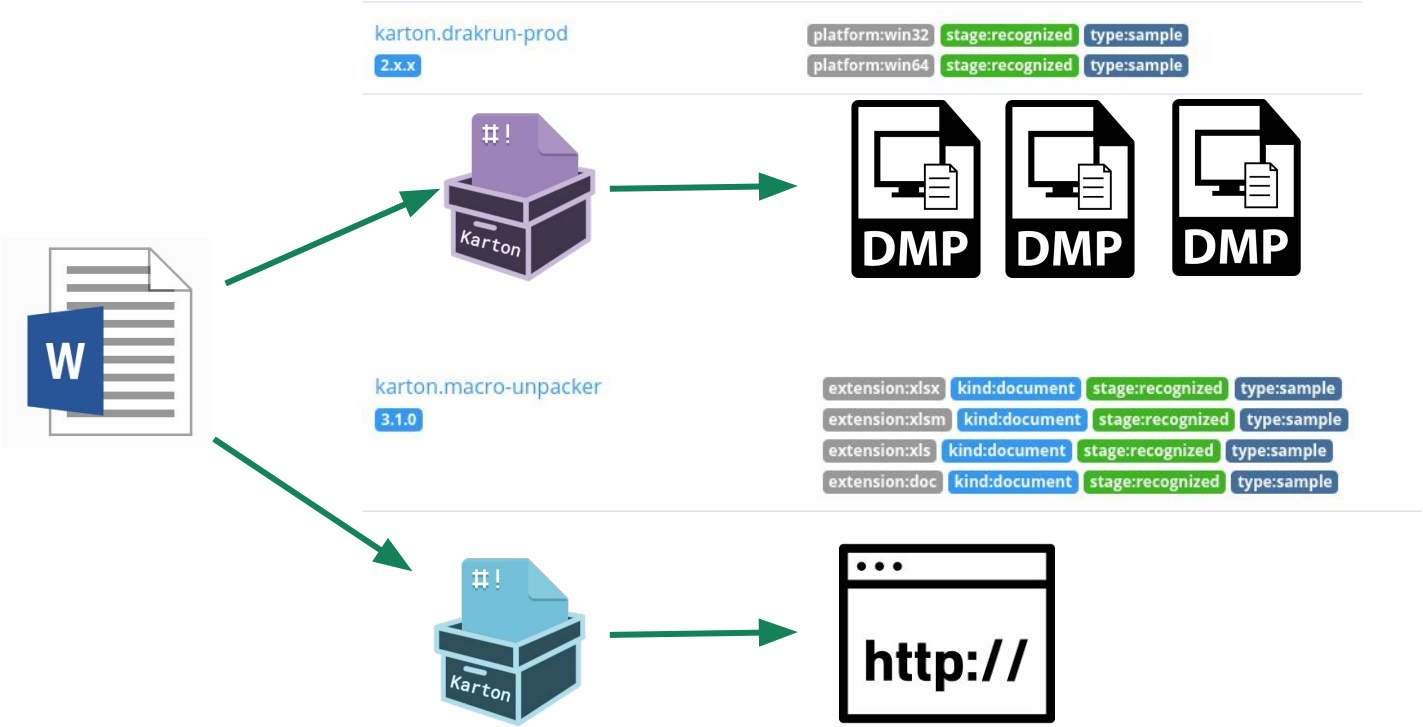
karton.cuckoo1
2.x.x
platform:win32 quality:high stage:recognized type:sample

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.roach-ripper

2.x.x

kind:runnable platform:win32 stage:recognized type:sample

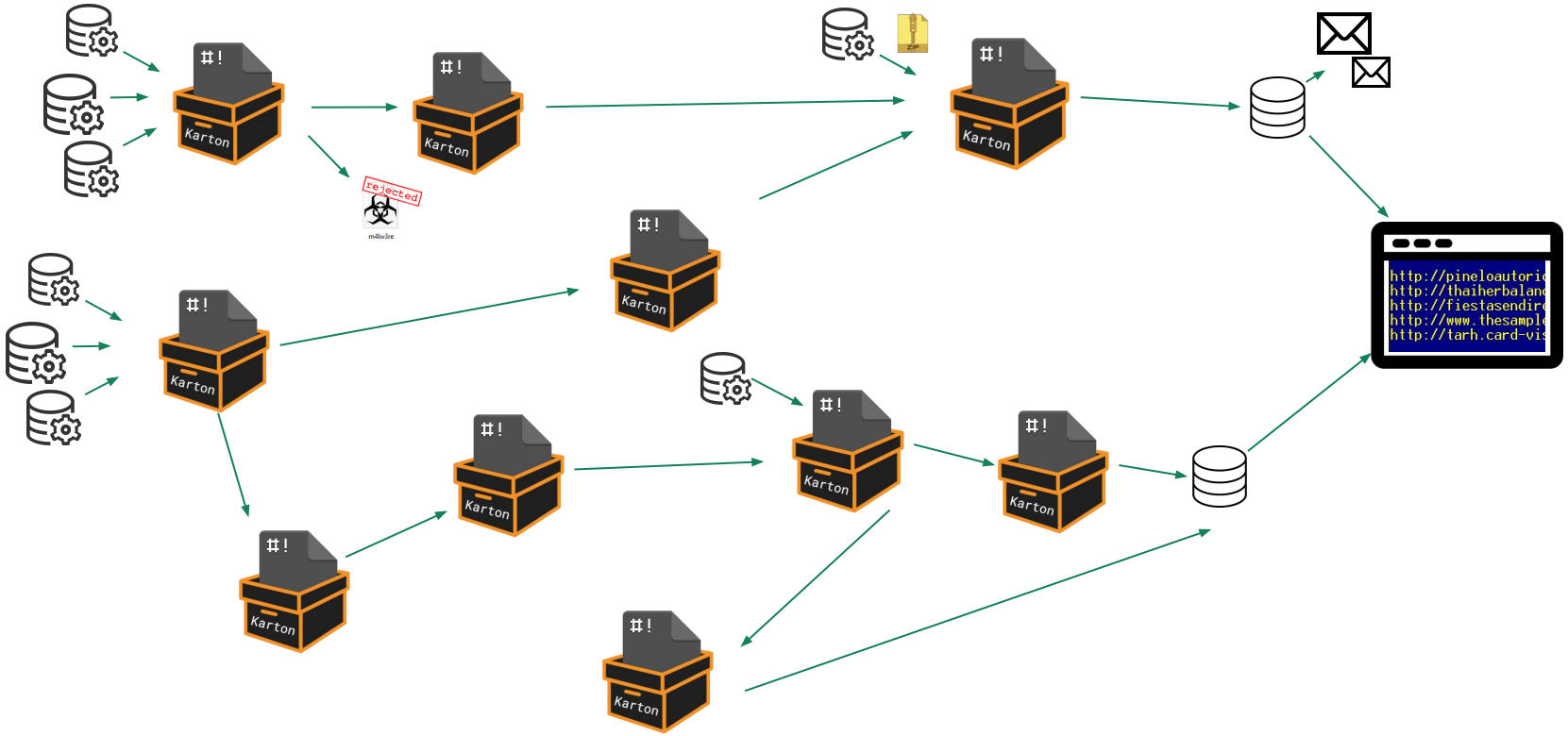
kind:runnable platform:win64 stage:recognized type:sample

kind:runnable platform:linux stage:recognized type:sample

kind:drakrun-prod type:analysis

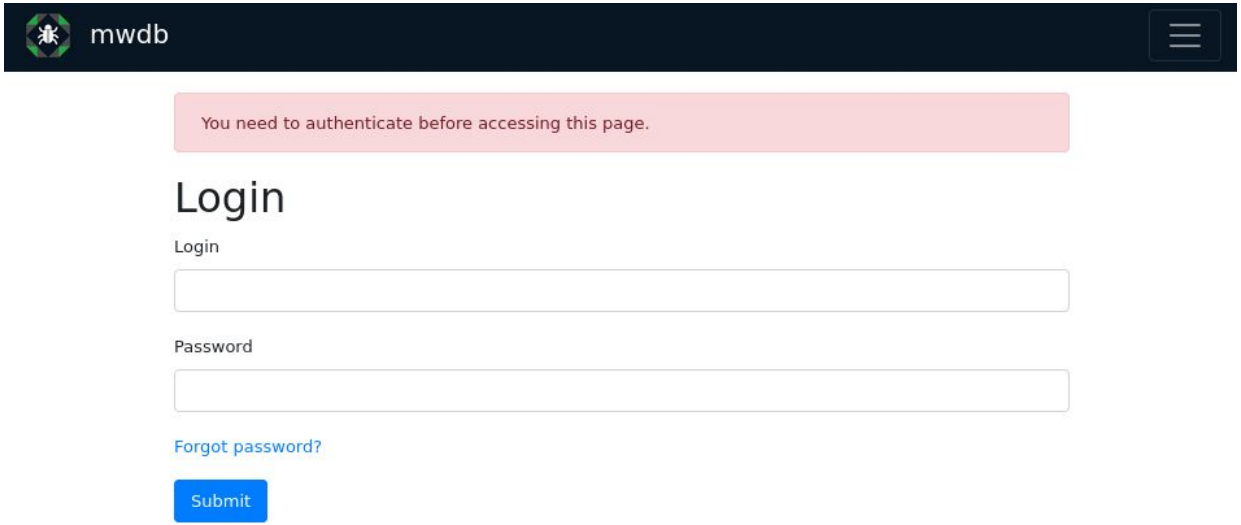
Family	zloader
Config type	static
+ binary_id	bot7
+ rc4key	NvuVIV3kbg7
+ rsa_key	-----BEGIN PUBLIC KEY----- MIG
+ type	zloader
+ urls	[{ "url": "https://militanttr

Karton pipeline in the real world



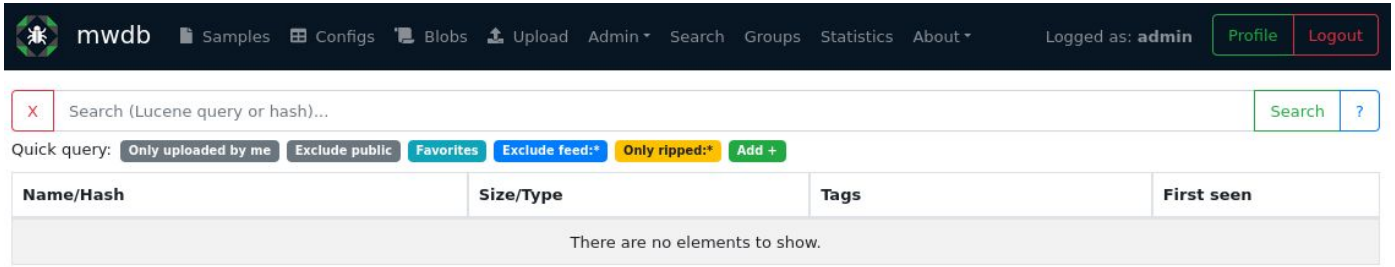
Karton Playground: click around

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!

Karton Playground: click around

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

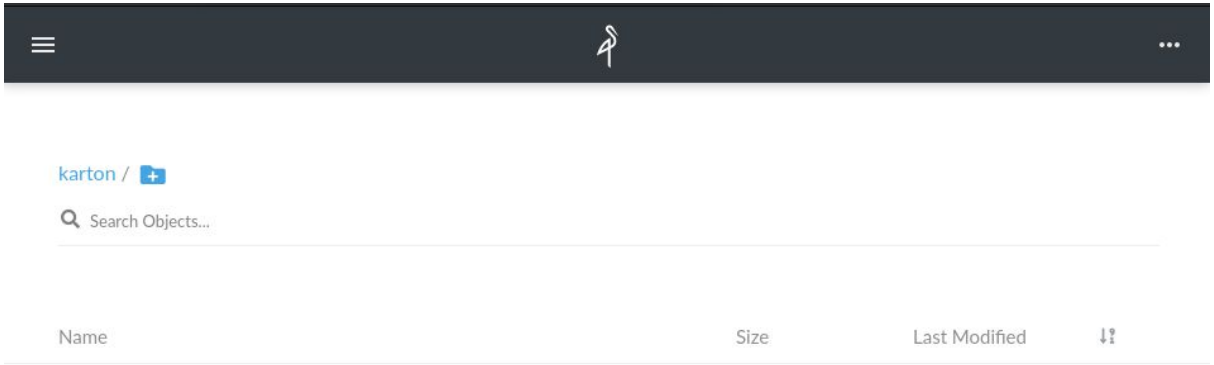
karton

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:

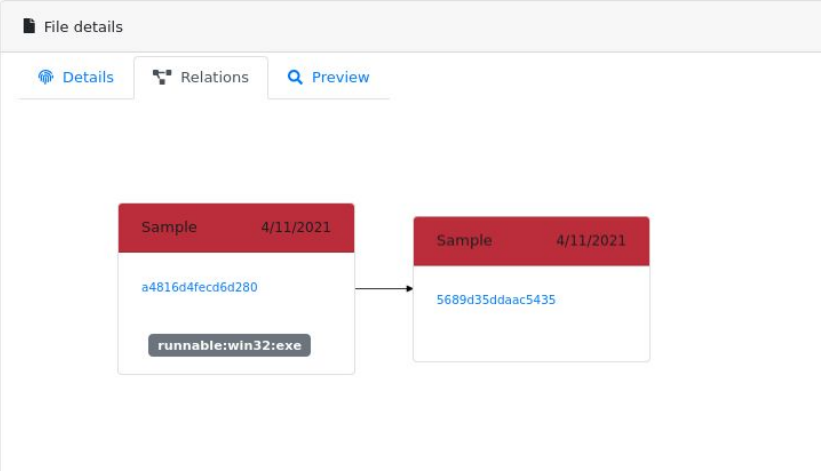
```
$ wget https://github.com/CERT-Polska/training-mwdb/blob/main/autoit-malware.bin
$ sha256sum autoit-malware.bin
a4816d4fecd6d2806d5b105c3aab55f4a1eb5deb3b126f317093a4dc4aab88a1 autoit-malware.bin
```

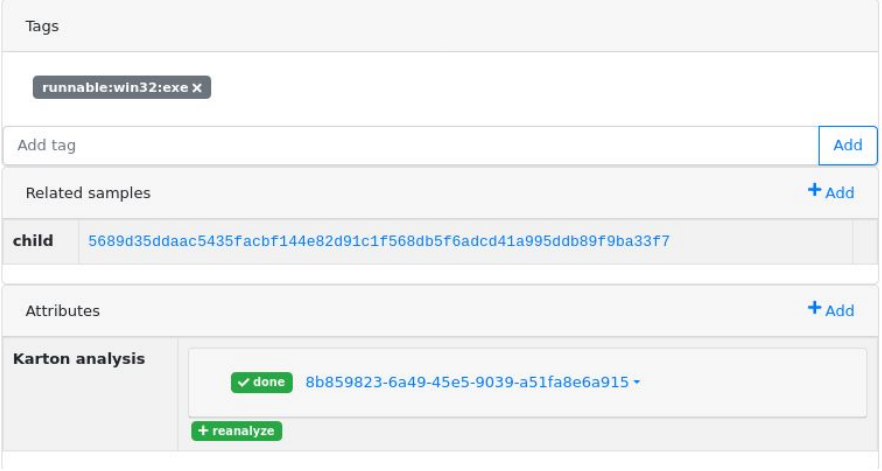
Finally, upload it to your local mwdb (http://127.0.0.1:8080, admin:admin)

```
$ 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
```

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
```

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
```

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

Details | Relations | Preview | Remove | + Upload child | Favorite | Download

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

runnable:win32:exe ×

Add tag | Add

Related samples | + Add

child	3eec0da26f86d1ae53d594d52adc3b50a596ed79e8701d130f52870c46a1d709

Karton Playground exercise: solution

mwdb Samples Configs Blobs Upload Admin Search Groups Statistics About

File details

Details Relations Preview Hex view Remove Upload child Favorite Download

```
!This program cannot be run in DOS mode.
iRichu
.text
`.rdata
@.data
.ndata
.rsrc
5Xp@
 s495
5`r@
tBj\V
uv9]
t——9]
tDH;
PShr
jHjZ
t=9]
t——j"
```

Coffee break

Malduck 🦆

Malduck 🦆

PSA: We'll be working with real malware samples (old ones though)

- Be careful (don't use Windows if you're not sure)
- If case some files go missing - Windows Defender might be the culprit

Malduck 🦆

What is it exactly?

Malduck 🦆

Per GitHub:

“Malduck is your ducky companion in malware analysis journeys”

Malduck 🦆

Per GitHub:

“Malduck is your ducky companion in malware analysis journeys”

Actually not a bad summary 🤔

Malduck 🦆

```
curl https://raw.githubusercontent.com/CERT-Polska/malduck/master/README.md | grep Features -A 8
```

- Cryptography (AES, Blowfish, Camelie, ChaCha20, Serpent and many others)
- Compression algorithms (aPLib, gzip, LZNT1 (RtlDecompressBuffer))
- 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)
- Fixed integer types (like Uint64) and bitwise utilities
- String operations (chunks, padding, packing/unpacking etc)
- Hashing algorithms (CRC32, MD5, SHA1, SHA256)

Malduck 🦆

```
curl https://raw.githubusercontent.com/CERT-Polska/malduck/master/README.md | grep Features -A 8
```

- Cryptography (AES, Blowfish, Camelie, ChaCha20, Serpent and many others)
- Compression algorithms (aPLib, gzip, LZNT1 (RtlDecompressBuffer))
- 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)
- Fixed integer types (like Uint64) and bitwise utilities
- String operations (chunks, padding, packing/unpacking etc)
- Hashing algorithms (CRC32, MD5, SHA1, SHA256)

- Time savers!

Malduck 🦆

```
curl https://raw.githubusercontent.com/CERT-Polska/malduck/master/README.md | grep Features -A 8
```

- Cryptography (AES, Blowfish, Camelie, ChaCha20, Serpent and many others)
- Compression algorithms (aPLib, gzip, LZNT1 (RtlDecompressBuffer))
- 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)
- Fixed integer types (like Uint64) and bitwise utilities
- String operations (chunks, padding, packing/unpacking etc)
- Hashing algorithms (CRC32, MD5, SHA1, SHA256)

- Time savers!
- Sanity savers!

Malduck 🦆

```
curl https://raw.githubusercontent.com/CERT-Polska/malduck/master/README.md | grep Features -A 8
```

- Cryptography (AES, Blowfish, Camelie, ChaCha20, Serpent and many others)
- Compression algorithms (aPLib, gzip, LZNT1 (RtlDecompressBuffer))
- 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)
- Fixed integer types (like Uint64) and bitwise utilities
- String operations (chunks, padding, packing/unpacking etc)
- Hashing algorithms (CRC32, MD5, SHA1, SHA256)

- Time savers!
- Sanity savers!
- Work savers!

Malduck 🦆

```
.data:000000000060107F                 db    0
.data:0000000000601080                 public key
.data:0000000000601080 key             db 'ZCUNIwR',0          ; DATA XREF: main+6B↑o
.data:0000000000601088                 align 20h
.data:00000000006010A0                 public ciphertext
.data:00000000006010A0 ; char ciphertext[1]
.data:00000000006010A0 ciphertext      db 'a'                  ; DATA XREF: main+44↑o
.data:00000000006010A0                                         ; main+66↑o
.data:00000000006010A1                 db 0EBh
.data:00000000006010A2                 db 0BFh
.data:00000000006010A3                 db  6Eh ; n
.data:00000000006010A4                 db 0BDh
.data:00000000006010A5                 db  2Ah ; *
.data:00000000006010A6                 db 0B4h
.data:00000000006010A7                 db  74h ; t
.data:00000000006010A8                 db 0ACh
.data:00000000006010A9                 db  88h
.data:00000000006010AA                 db  4Fh ; O
.data:00000000006010AB                 db 0C4h
.data:00000000006010AC                 db 0B9h
.data:00000000006010AD                 db  92h
.data:00000000006010AE                 db 0F1h
.data:00000000006010AF                 db 0C0h
.data:00000000006010B0                 db 0BBh
.data:00000000006010B1                 db  80h
.data:00000000006010B2                 db 0DCh
.data:00000000006010B3                 db  19h
.data:00000000006010B4                 db  0Dh
.data:00000000006010B5                 db  36h ; 6
.data:00000000006010B6                 db  30h ; 0
.data:00000000006010B7                 db   5
.data:00000000006010B8                 db 0AEh
.data:00000000006010B9                 db   7
.data:00000000006010BA                 db 0A9h
.data:00000000006010BB                 db  93h
.data:00000000006010BC                 db  10h
.data:00000000006010BD                 db  49h ; I
.data:00000000006010BE                 db   3
.data:00000000006010BF                 db 0ACh
.data:00000000006010C0                 db  0Eh
.data:00000000006010C1                 db   0
.data:00000000006010C1 _data           ends
```

```
Python>
Python>
Python>
Python>
Python>
Python>malduck.rc4(get_bytes(0x601080, 7), get_bytes(0x6010A0, 33))
b'flag{27206a210aa187c1c5634d23525}'
Python
```

CERT.PL >_

Malduck 🦆

```
.data:000000000060107F                 db    0
.data:0000000000601080                 public key
.data:0000000000601080 key             db 'ZCUNIwR',0          ; DATA XREF: main+6B↑o
.data:0000000000601088                 align 20h
.data:00000000006010A0                 public ciphertext
.data:00000000006010A0 ; char ciphertext[1]
.data:00000000006010A0 ciphertext      db 'f'                  ; DATA XREF: main+44↑o
.data:00000000006010A0                                         ; main+66↑o
.data:00000000006010A1                 db  6Ch ; 1
.data:00000000006010A2                 db  61h ; a
.data:00000000006010A3                 db  67h ; g
.data:00000000006010A4                 db  7Bh ; {
.data:00000000006010A5                 db  32h ; 2
.data:00000000006010A6                 db  37h ; 7
.data:00000000006010A7                 db  32h ; 2
.data:00000000006010A8                 db  30h ; 0
.data:00000000006010A9                 db  36h ; 6
.data:00000000006010AA                 db  61h ; a
.data:00000000006010AB                 db  32h ; 2
.data:00000000006010AC                 db  31h ; 1
.data:00000000006010AD                 db  30h ; 0
.data:00000000006010AE                 db  61h ; a
.data:00000000006010AF                 db  61h ; a
.data:00000000006010B0                 db  31h ; 1
.data:00000000006010B1                 db  38h ; 8
.data:00000000006010B2                 db  37h ; 7
.data:00000000006010B3                 db  63h ; c
.data:00000000006010B4                 db  31h ; 1
.data:00000000006010B5                 db  63h ; c
.data:00000000006010B6                 db  35h ; 5
.data:00000000006010B7                 db  36h ; 6
.data:00000000006010B8                 db  33h ; 3
.data:00000000006010B9                 db  34h ; 4
.data:00000000006010BA                 db  64h ; d
.data:00000000006010BB                 db  32h ; 2
.data:00000000006010BC                 db  33h ; 3
.data:00000000006010BD                 db  35h ; 5
.data:00000000006010BE                 db  32h ; 2
.data:00000000006010BF                 db  35h ; 5
.data:00000000006010C0                 db  7Dh ; }
.data:00000000006010C1                 db    0
.data:00000000006010C1 _data           ends
```

```
Python>
Python>
Python>
Python>
Python>
Python>data = malduck.rc4(get_bytes(0x601080, 7), get_bytes(0x6010A0, 33))
Python>ida_bytes.patch_bytes(0x6010A0, data)
```

Python

Malduck 🦆

Memory model abstraction

- One interface to rule them all (ELF, PE, IDA)

Malduck 🦆

Memory model abstraction

- One interface to rule them all (ELF, PE, IDA)
- “p” vs “v” suffix (readv, uint32p, etc)

Malduck 🦆

Memory model abstraction

- One interface to rule them all (ELF, PE, IDA)
- “p” vs “v” suffix (readv, uint32p, etc)
- PE files dumped from memory snapshots

Malduck 🦆

Exercise #4.1: Getting familiar with Malduck

1. Explore the CLI
2. Crypto functions
3. Disassembly engine

Malduck 🦆

Exercise #4.1: Getting familiar with Malduck

1. Explore the CLI
2. Crypto functions
3. Disassembly engine

Hints:

- Most commonly used methods are exposed on module level, i.e. you can do `from malduck import disasm`

Malduck 🦆

Extracting configuration

Malduck 🦆

Extracting configuration

files / memory dumps

What exactly happens here?

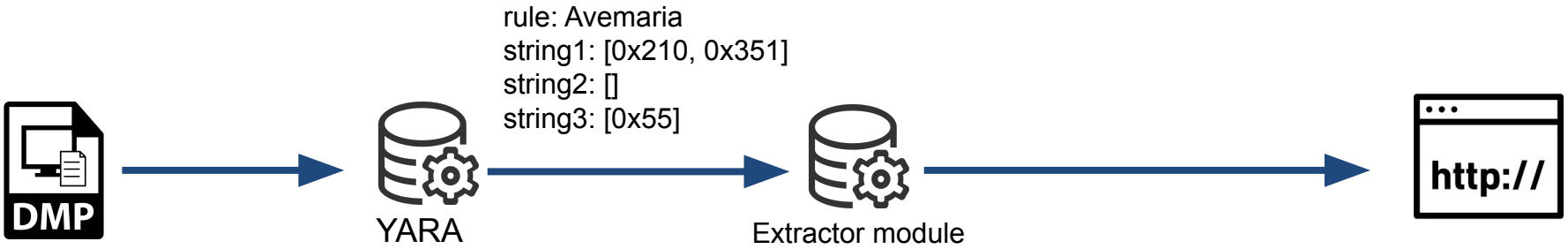
configuration (IOC)

Malduck 🦆

Extracting configuration

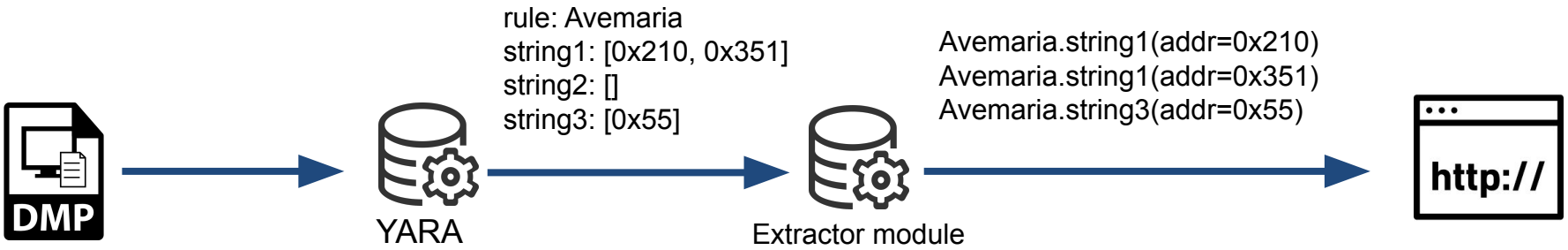
Malduck 🦆

Extracting configuration



Malduck 🦆

Extracting configuration



Malduck 🦆

Exercise #4.2: Extracting Warzone RAT C2 server info

- Decrypt the config using RC4
- Parse the output blob to get C2 host & port

```
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 🦆

Config extraction - General notes

Malduck 🦆

Config extraction - General notes

- It’s much harder in some cases…

Malduck 🦆

Config extraction - General notes

- It’s much harder in some cases… But exactly this easy in many others!

Malduck 🦆

Config extraction - General notes

- It’s much harder in some cases… But exactly this easy in many others!
- Overthinking it is sometimes not worth it

Malduck 🦆

Config extraction - General notes

- It’s much harder in some cases… But exactly this easy in many others!
- Overthinking it is sometimes not worth it
- Samples are usually packed - out of scope for this talk

Malduck 🦆

Exercise #4.3: Creating extraction modules from the ground up

Malduck 🦆

Exercise #4.3: Creating extraction modules from the ground up

Hints

- Start with YARA rules, 3 of them should cover all samples
- All password should match “flag{<hexdigits>}”
- From crypto perspective, you only need xor & RC4
- Useful methods:
  - procmem.readv
  - procmem.asciiz
  - procmem.disasmv

Malduck 🦆

Exercise #4.4: Bonus: Integrating implemented modules into karton-config-extractor

Malduck 🦆

Exercise #4.4: Bonus: Integrating implemented modules into karton-config-extractor

Hints

- Copy all created modules into a single “modules” directory
- Start karton-config-extractor
- Upload all samples to your MWDB instance and watch the configs appear

Malduck 🦆: sharing is caring

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

- But if you create a malduck module and want to share it with us, we’ll be happy to add it to mwdb.cert.pl

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
michal.praszmo@cert.pl
jaroslaw.jedynak@cert.pl
info@cert.pl

CERT.PL >_

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