Malware config extraction at scale – building malware analysis pipelines

Michał Praszmo

JSAC 2025

Agenda

- **mwdb**
  - What the heck is **MWDB?**
  - Tour de **mwdb.cert.pl**
  - Hosting your own instance
- **malduck**
  - About **malduck**
  - Config extraction with **malduck**
  - Integrate config extractor in **karton/mwdb**

About me: security engineer at CERT Polska (cert.pl)

CERT Polska: One of 3 national CERTs of Poland, part of NASK research institute

Warning

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://**bzcat.eu**

What the heck is MWDB?

- Malware repository
- Designed with automated analysis in mind
- Highly extensible

About

Malware repository component for samples & static configuration with REST API interface.

mwdb.readthedocs.io/

Similar projects/services

Core object types

Core object types

Samples | Configs | Blobs

File name	2024-12-22_b74eefcf1a45d1e338ec795822c1be4d_wannacry
Variant file names	
File size	4.8 MB
File type	PE32 executable (GUI) Intel 80386, for MS Windows, 4 sections
md5	b74eefcf1a45d1e338ec795822c1be4d
sha1	4e9a12f80f7ca7fc6b24859cce4139ce350abaa4
sha256	821364de16646ba2ceda8c343fe9a595c3587bcf0726d84a998f93a34c9b9fd1
sha512	3ab3109aa81c1a6b40ff636dec523e9275a4a9a862395723f980cc7cdc845ada60c4951d53e238b0bceb569f5ac28b2291e9e0bde3886ebca358ef4ae957e942
crc32	22aada3d
ssdeep	12288:nQhMbaIMu7L5NVErCA4z2g6rTcbckPU82900Ve7zw+K+DzUgZLHJ98kDs:nQhfdmMSirYbcMNgef0yD8kI
Upload time	Sun, 22 Dec 2024 12:51:29 GMT

Core object types

Family	revengerat
Config type	static
+ cncs	[{ "host": "necatisoff-36486.portmap.host", "port": "36486" }]…
+ id	Guest
+ key	Revenge-RAT
+ mutex	RV_MUTEX-ETUKIWwiejYAo
+ type	revengerat
Upload time	Tue, 07 Jan 2025 18:41:44 GMT

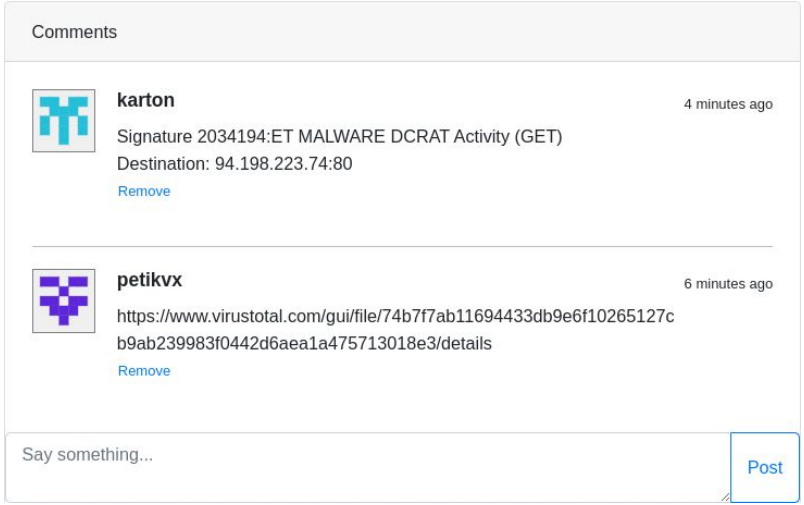

```
{
    "cncs": [
        {
            "host": "necatisoff-36486.portmap.host",
            "port": "36486"
        }
    ],
    "id": "Guest",
    "key": "Revenge-RAT",
    "mutex": "RV_MUTEX-ETUKIWwiejYAo",
    "type": "revengerat"
}
```

Core object types

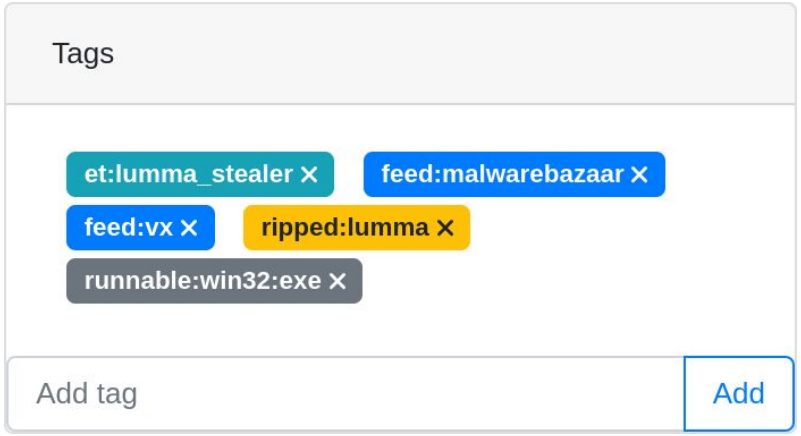
Blob name	agent_tesla_strings
Blob size	12.02 kB
Blob type	raw_cfg
First seen	Sat, 21 Dec 2024 23:51:02 GMT
Last seen	Sat, 21 Dec 2024 23:51:05 GMT

```
134 \Data\Tor\torrc¬
135 p=¬
136 %PostURL%¬
137 127.0.0.1¬
138 POST¬
139 +¬
140 %2B¬
141 application/x-www-form-urlencoded¬
142 m.briceno@s-s.cl¬
143 Systems@2011¬
144 mail.s-s.cl¬
145 chrisluda76@yandex.ru¬
146 image/jpg¬
147 %ftphost%/¬
148 %ftpuser%¬
149 %ftppassword%¬
150 STOR¬
151 Write¬
152 Close¬
153 Addchat_idcaption¬
154 /sendDocument¬
155 document¬
156 ----------------------------¬
157 x¬
158 ¬
159 --¬
160 ¬
```

Secondary objects - Comments



Secondary objects - Tags

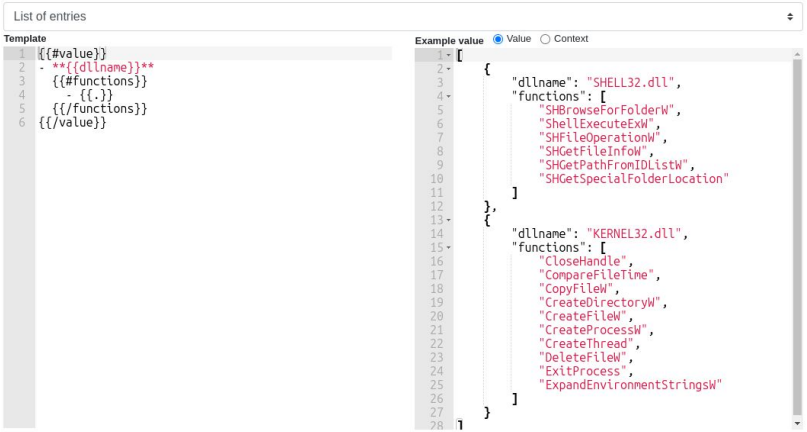


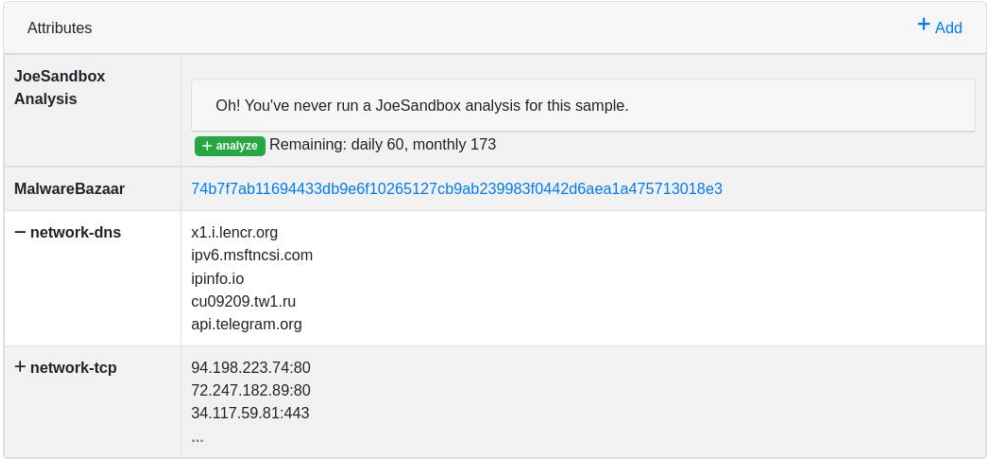
Secondary objects - Tags

lumma ✕	Simple tag, mostly used for marking artifacts that are associated with malware family
feed:malwarebazaar ✕	Tag describing the source of malware sample
ripped:lumma ✕	Tag for the original sample, indicating recognized malware family
runnable:win32:exe ✕	Tag describing the type of sample
et:lumma_stealer ✕	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

Secondary objects - Attributes

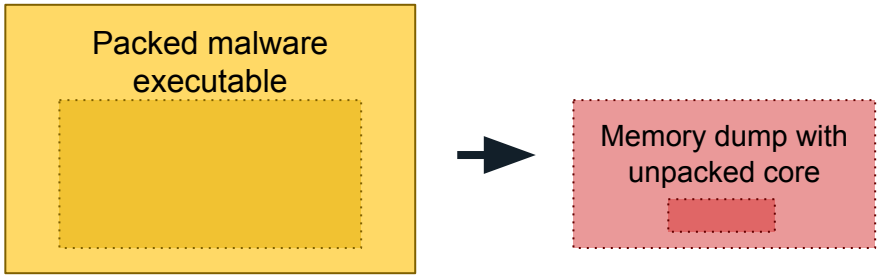




DRAKVUF Sandbox

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.)

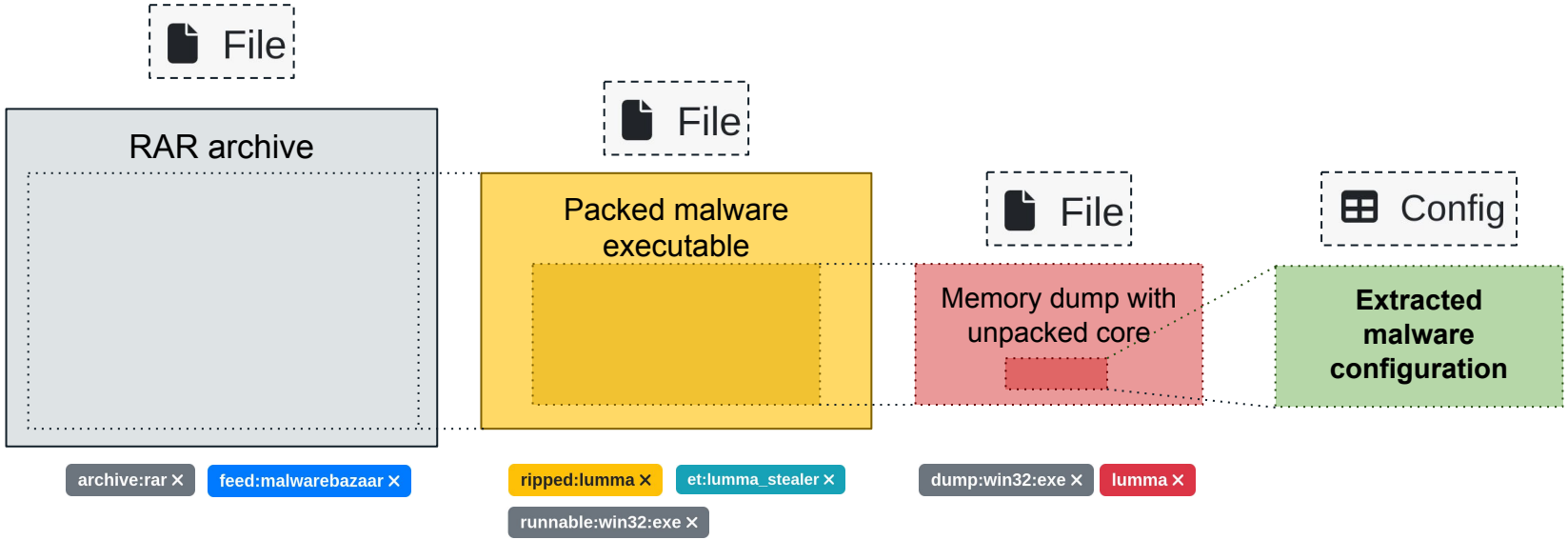
Uses various heuristics for choosing memory regions that may contain unpacked code.



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

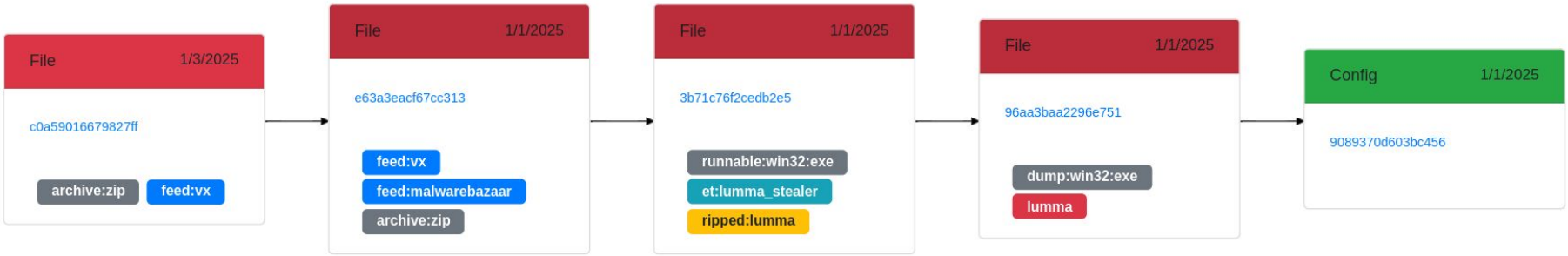
https://github.com/tklengyel/drakvuf

Object Relationships



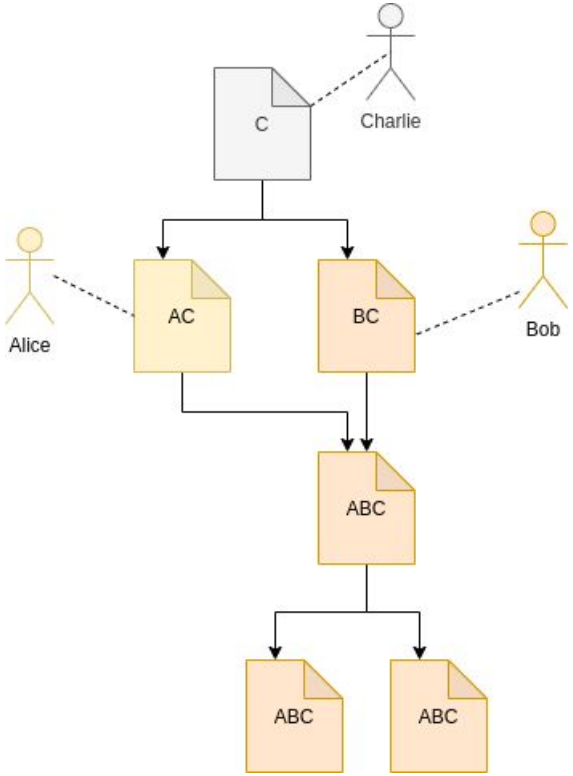
Object Relationships

Relations

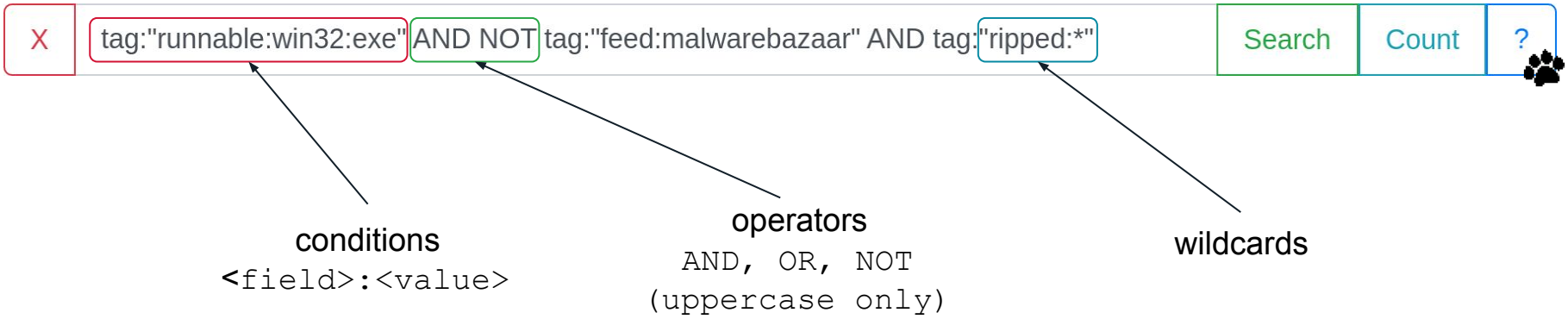


Object Relationships - Permissions

Shares	
Added by:	
wtfurl	Sat, 11 Jan 2025 19:37:44 GMT
karton	Sat, 11 Jan 2025 19:37:44 GMT
Shared by wtfurl:	
public	Sat, 11 Jan 2025 19:37:44 GMT
Shared by karton:	
certpl-systems	Sat, 11 Jan 2025 19:37:44 GMT
certpl	Sat, 11 Jan 2025 19:37:44 GMT
Share with group	Add

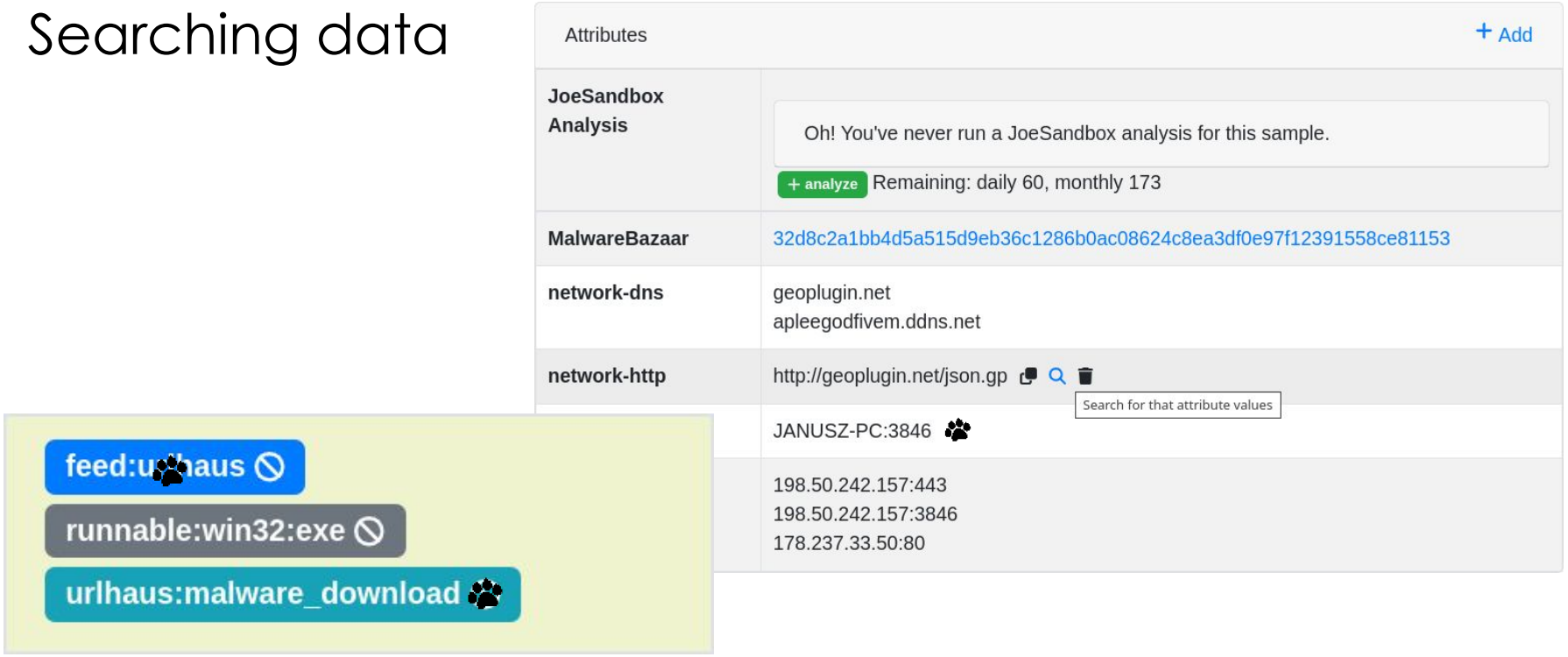


Searching data - Lucene search

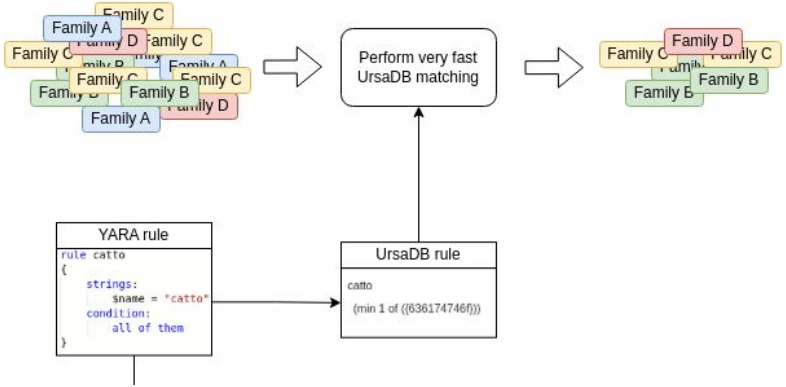


https://mwdb.readthedocs.io/en/latest/user-guide/7-Lucene-search.html

Searching data



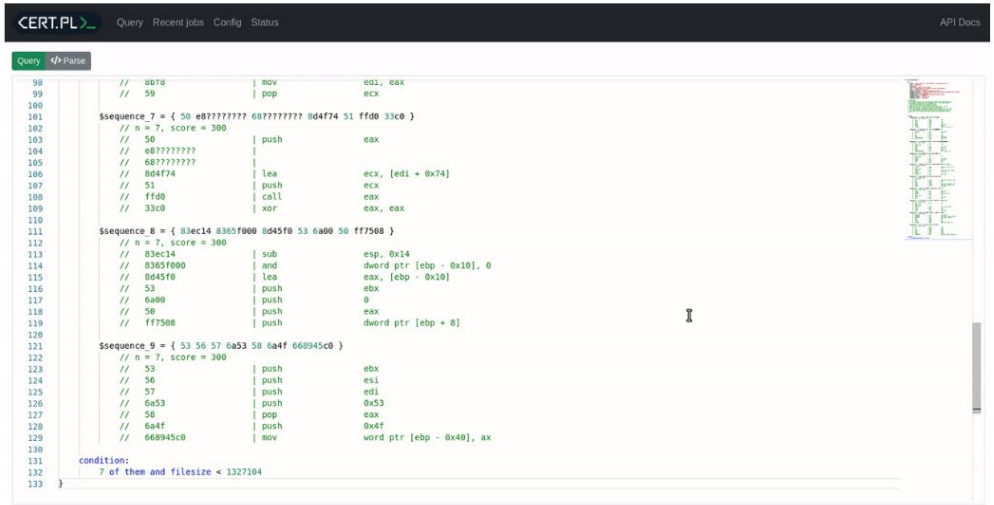
Searching data - other methods



mquery: Blazingly fast Yara queries for malware analysts

Ever had trouble searching for malware samples? Mquery is an analyst-friendly web GUI to look through your digital warehouse.

It can be used to search through terabytes of malware in a blink of an eye:



Automation

- **CLI** -> quick bash scripts
- **Python API** -> production integrations
- **REST API** -> other languages

```
# Create virtualenv and activate
$ python3 -m venv venv
$ . venv/bin/activate
```

```
# Install mwdblib with CLI extras
(venv) $ pip install mwdblib[cli]
(venv) $ mwdb login
```

```
(venv) $ mwdb search {files, configs, blobs}
(venv) $ mwdb list {files, configs, blobs}
(venv) $ mwdb upload
(venv) $ mwdb fetch
```

Automation

Q000112241_R02_October-24.exe f0ee4e613fe0c318e302dced6e2e3e459a71585100290da904f5575e73a2be8	974.9 kB	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections ripped:remcos yara:win_remcos runnable:win32:exe yara:indicator_kb_cert_7c1118cbbadc95da3752c46e47a27438 feed:vx	Nov 06
PO238109.exe 25d5929f0ef894bf532d5c21e03474a7f7db7cc0be168a2d618a40bb47de9643	959.5 kB	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections et:remcos ripped:remcos feed:malwarebazaar runnable:win32:exe yara:win_remcos feed:vx	Nov 06
INQ9970.exe d682eeadb7f5d9c10016bbe8ee8f8f16938d3f7c7b33b9703225efd552df6d5b	964.1 kB	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections runnable:win32:exe et:remcos ripped:remcos feed:malwarebazaar feed:vx yara:win_remcos	Nov 06
SecuriteInfo.com.Win32.CrypterX-gen.28521.19527.exe f5a51a5492d785c8e485251c34b7ccef2f676bc507794c219403e750c788fbe9	950.8 kB	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections feed:malwarebazaar yara:win_remcos ripped:remcos runnable:win32:exe feed:vx	Nov 06
ATH0000878718.pdf.exe 7d9a4ed0a06ed9371e27c634f50c6aed4ebc1869c4a094287b03a6be4b810c63	3.9 MB	PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows, 2 sections feed:malwarebazaar yara:win_remcos runnable:win64:exe ripped:remcos feed:vx	Nov 05
6Ctc0o7vhqKgjU7.exe 63ac85fa66152f936244088e40eb124a6888336a4508f8d3d63d818ad30e4280	1.0 MB	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections feed:malwarebazaar runnable:win32:exe feed:vx yara:win_remcos ripped:remcos yara:exeinfection	Nov 05
B89BAC1CC03E354616B2ABB93E46B630.exe b9f9560d6685fc8b8140b21d45f4a7c0db161fdb9d21f6e8f2761d96e4369d0f	46.1 kB	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections feed:malwarebazaar runnable:win32:exe yara:win_asyncrat ripped:asyncrat asyncrat yara:win_remcos ripped:remcos et:asyncrat yara:indicator_suspicious_exe_asep_reg_reverse feed:vx yara:windows_trojan_asyncrat_11a11ba1 yara:sandboxdetect_misc yara:windows_generic_threat_ce98c4bc	Nov 02
New_Order_070824_Order_November-2024-pdf.exe 6e2e6d5f67b4bd2df522de36e957e5d0b3f97eb8b3a6a5460cc3aa32cf112cdc	3.6 MB	PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows, 2 sections yara:win_remcos runnable:win64:exe ripped:remcos feed:malwarebazaar feed:vx	Nov 02
1730477226c46d247f8149bb08962a395eff3ba2277df18f1516091fac7e907c6a25be5f0f687.dat-decoded.exe 9d83a44f6b584d99d1ffebd7c9b915f68056d697f0b1debe9386ffb78aae0deb	494.6 kB	PE32 executable (GUI) Intel 80386, for MS Windows, 7 sections feed:malwarebazaar yara:win_remcos runnable:win32:exe ripped:remcos remcos feed:vx yara:indicator_suspicious_exe_uacbypass_cmstpcom yara:win_remcos_auto yara:windows_trojan_remcos_b296e965	Nov 01
z1ProductSampleRequirement.exe 1682ee7703dd036cbdf6ad6daa38ddb7a4e7ab567b273f9ee209672f339feb2d	1.0 MB	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections yara:win_remcos feed:malwarebazaar runnable:win32:exe ripped:remcos feed:vx	Nov 01
17304052250b9baaf5a761ccc772d95d677ec70f56bbae9f30fbbf26b5b71b9b9867fc8bb2802.dat-decoded.exe 82d05f7c1e3d16ba7e22348af4c14533cce64567b024e2149b511c62a85c81bc	494.6 kB	PE32 executable (GUI) Intel 80386, for MS Windows, 7 sections feed:malwarebazaar ripped:remcos remcos yara:win_remcos runnable:win32:exe feed:vx yara:windows_trojan_remcos_b296e965 yara:win_remcos_auto yara:indicator_suspicious_exe_uacbypass_cmstpcom yara:remcos	Oct 31

15 min to explore the platform and try some challenges

- Challenges hosted on https://**ctf.bzcat.eu** (use whatever username/email you like)
- “mwdb” category
- Tasks contain links to solutions/tips
- Links to projects/docs/slides available at https://**bzcat.eu**

Malduck 🦆

Per GitHub:

“Malduck is your ducky companion in malware analysis journeys”

Actually not a bad summary 🤔

Malduck 🦆

cat README.md | grep Features

- 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 🦆

cat README.md | grep Features

- 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 🦆 - saving time

```
.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
```

Malduck 🦆 - saving time

```
.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 🦆

cat README.md | grep Features

- 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 🦆 - fixed ints

- **Unsigned types:**

  `UInt64` (`QWORD`), `UInt32` (`DWORD`), `UInt16` (`WORD`), `UInt8` (`BYTE` or `CHAR`)

- **Signed types:**

  `Int64` , `Int32` , `Int16` , `Int8`

```python
def decrypt_data(key_1, key_2, data):

    out = []
    for chunk in chunks(data, 16):
        key = key_2
        key = xor(key, key_1)

        a, b, c, d = (UInt32 * 4).unpack(key)
        for _ in range(16):
            x = (b + a) ^ b.rol(5)
            y = (d + c) ^ d.rol(8)
            z = x + d + c

            a = (y + (a + b).rol(16))
            d = (a ^ y.rol(13))
            b = (z ^ x.rol(7))
            c = (z.rol(16))

        key = a.pack() + b.pack() + c.pack() + d.pack()
        key = xor(key, key_1)
        out.append(xor(key, chunk))
        key_2 = derive_key(key_2)
    return b"".join(out)
```

Malduck 🦆

cat README.md | grep Features

- 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 🦆

cat README.md | grep Features

- 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 🦆 - Memory model objects

- Abstraction for memory-mapped binaries
- Supports many platforms/architectures
- Makes code very reusable

- Memory model objects (procmem)
  - ProcessMemory (procmem)
    - `procmem`
    - `ProcessMemory`
    - `Region`
  - ProcessMemoryPE (procmempe)
    - `procmempe`
    - `ProcessMemoryPE`
  - ProcessMemoryELF (procmemelf)
    - `procmemelf`
    - `ProcessMemoryELF`
  - CuckooProcessMemory (cuckoomem)
    - `cuckoomem`
    - `CuckooProcessMemory`
  - IDAProcessMemory (idamem)
    - `idamem`
    - `IDAProcessMemory`

Malduck 🦆

Extracting configuration

files / memory dumps

What exactly happens here?

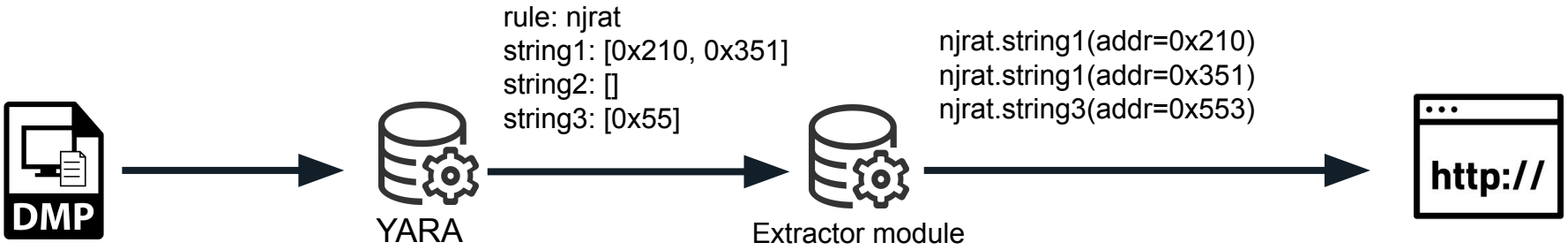
configuration (IOC)

Malduck 🦆

Extracting configuration

Malduck 🦆

Extracting configuration



```python
class Njrat(Extractor):
    yara_rules = ("win_njrat",)
    family = "njrat"

    @Extractor.extractor
    def op_config_07d(self, p: procmem, match: int) -> Dict[str, Any]:
        # Skip to the config address by adding number of bytes
        # in the op_config.
        start_addr = match + 17
        str_list = read_wide_dotnet_strings(p, 8, start_addr)

        str_list[0] = base64.decode(str_list[0])
        config = {
            "id": str_list[0],
            "version": str_list[1],
            "filename": str_list[2],
            "directory": str_list[3],
            "registry": str_list[4],
            "cncs": [{"host": str_list[5], "port": str_list[6]}],
            "registry_key": str_list[7],
        }

        return config
```

```
rule win_njrat {
```

```python
class Njrat(Extractor):
    yara_rules = ("win_njrat",)
    family = "njrat"

    @Extractor.extractor
    def op_config_07d(self, p: procmem, match: int) -> Dict[str, Any]:
        # Skip to the config address by adding number of bytes
        # in the op_config.
        start_addr = match + 17
        str_list = read_wide_dotnet_strings(p, 8, start_addr)

        str_list[0] = base64.decode(str_list[0])
        config = {
            "id": str_list[0],
            "version": str_list[1],
            "filename": str_list[2],
            "directory": str_list[3],
            "registry": str_list[4],
            "cncs": [{"host": str_list[5], "port": str_list[6]}],
            "registry_key": str_list[7],
        }

        return config
```

```
rule win_njrat {
```

```
$op_config_07d = { 46 69 78 00 6B 00 57 52 4B
```

```python
class Njrat(Extractor):
    yara_rules = ("win_njrat",)
    family = "njrat"

    @Extractor.extractor
    def op_config_07d(self, p: procmem, match: int) -> Dict[str, Any]:
        # Skip to the config address by adding number of bytes
        # in the op_config.
        start_addr = match + 17
        str_list = read_wide_dotnet_strings(p, 8, start_addr)

        str_list[0] = base64.decode(str_list[0])
        config = {
            "id": str_list[0],
            "version": str_list[1],
            "filename": str_list[2],
            "directory": str_list[3],
            "registry": str_list[4],
            "cncs": [{"host": str_list[5], "port": str_
            "registry_key": str_list[7],
        }

        return config
```

```
rule win_njrat {
```

```
$op_config_07d = { 46 69 78 00 6B 00 57 52 4B
```

Family	njrat
Config type	static
+ cncs	[{ "host": "rusia.duckdns.org", "port": "1994" }]
+ id	NYAN CAT
+ registry	aed0817703934
+ type	njrat
+ version	0.7NC
Upload time	Wed, 24 Apr 2024 15:08:20 GMT

15 min to play with ducks and try some challenges

- Challenges hosted on https://**ctf.bzcat.eu**
- “malduck” category
- Tasks contain links to solutions/tips
- Links to projects/docs/slides available at https://**bzcat.eu**

What the heck is Karton?

- Framework for creating pipelines
- Created for malware, works for other things as well
- Many microservices, each doing one thing right

About

Distributed malware processing framework based on Python, Redis and S3.

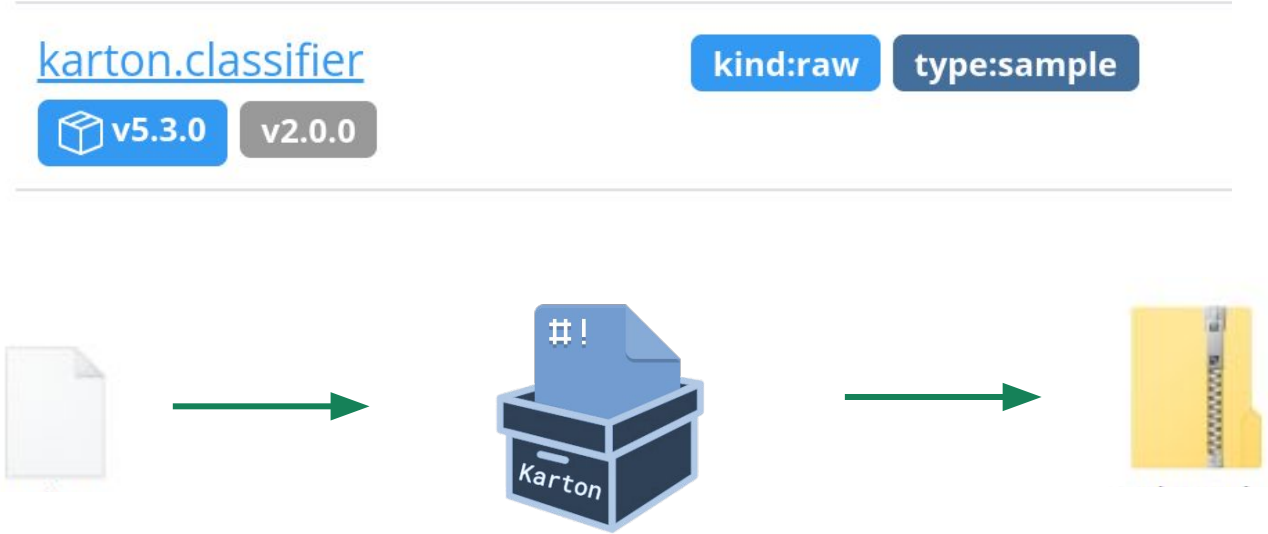
karton-core.readthedocs.io/en/latest/

Similar projects

ASSEMBLYLINE

INTELMQ

Behind the scenes - Karton



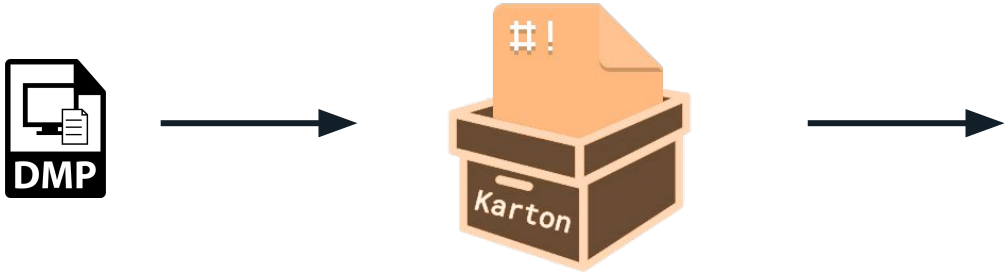
Behind the scenes - Karton

Behind the scenes - Karton

karton.config-extractor

v5.5.1 v2.2.0

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:runnable platform:freebsd stage:recognized type:sample
kind:runnable platform:netbsd stage:recognized type:sample
kind:runnable platform:openbsd stage:recognized type:sample
kind:runnable platform:solaris stage:recognized type:sample
type:analysis



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

Behind the scenes - Karton

```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.main()  # Start the karton service
```

README BSD-3-Clause license

Config-extractor karton service

Extracts static configuration from samples and memory dumps using the malduck engine.

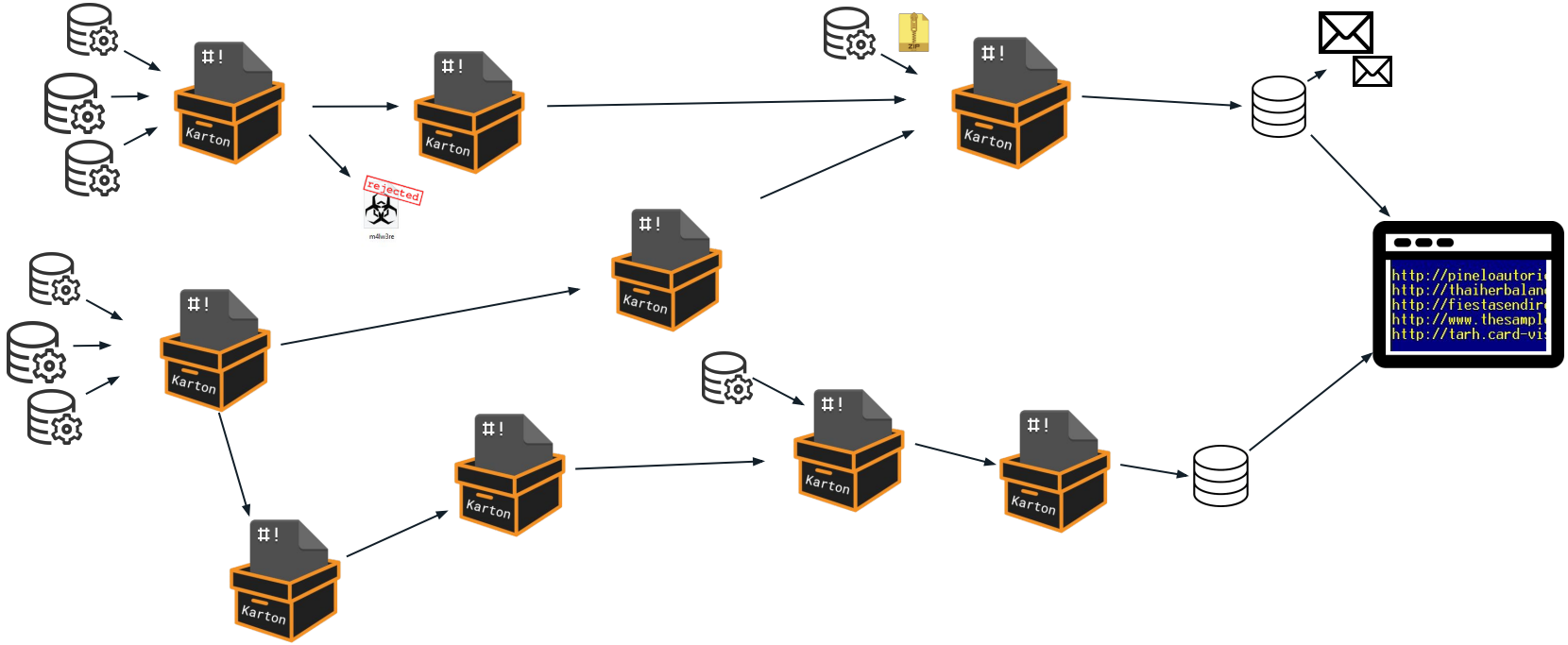
Author: CERT.pl

Maintainers: nazywam, psrok1, msm

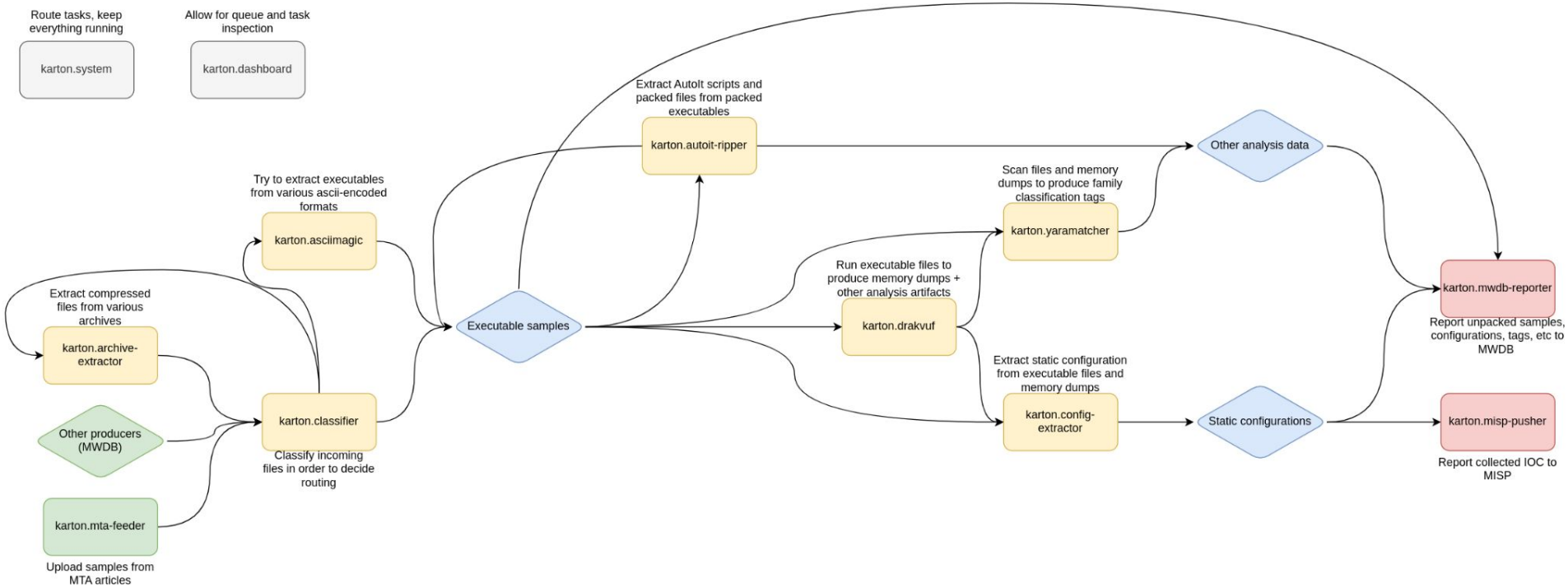
Consumes:

```
{
    "type": "sample",
    "stage": "recognized",
    "kind": "runnable",
    "platform": "win32"
},
{
    "type": "sample",
    "stage": "recognized",
    "kind": "runnable",
    "platform": "win64"
},
```

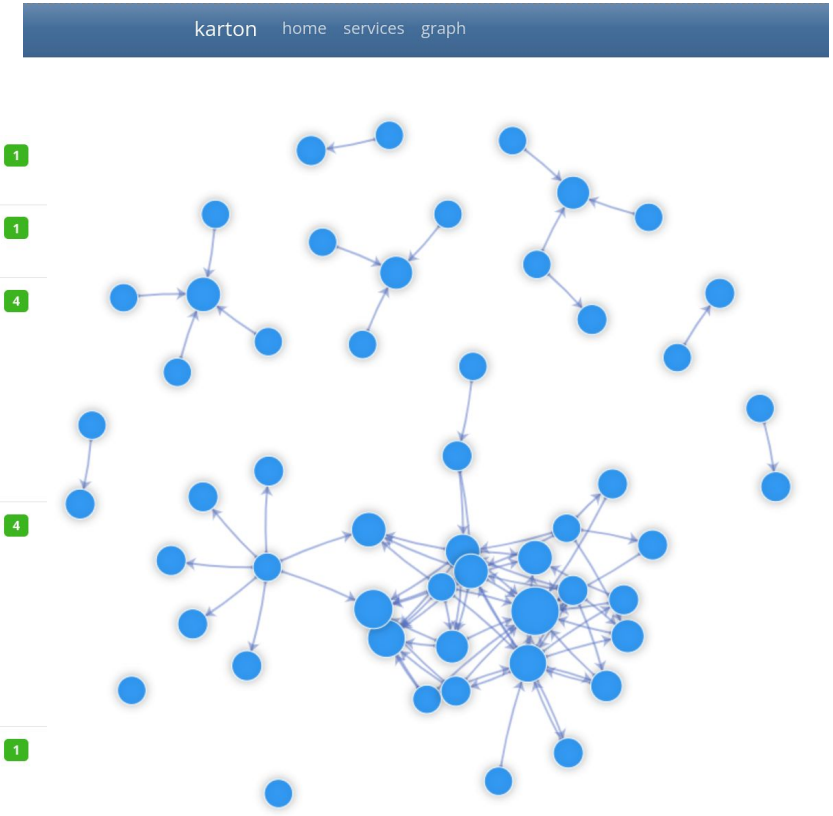
Karton pipeline



Real word karton pipeline



Real-er karton pipeline

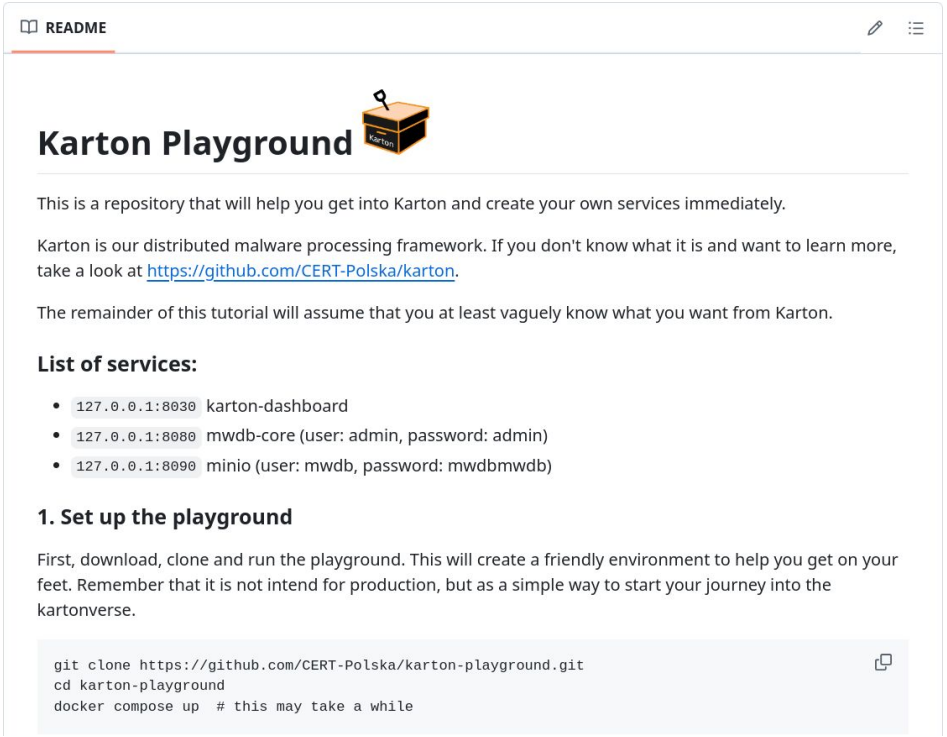


Hosting your own instance

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

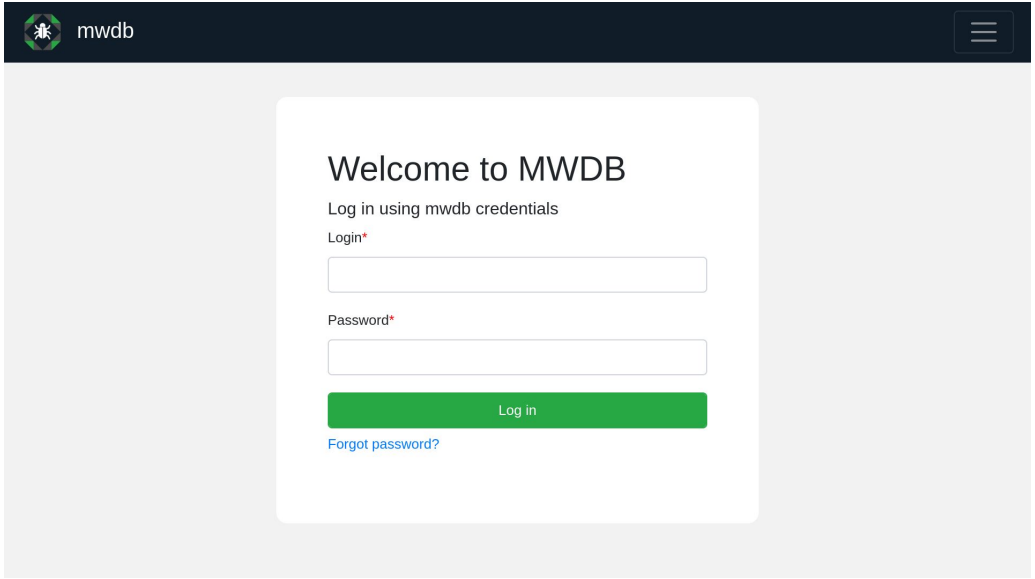
$ cd karton-playground

$ docker compose up



Hosting your own instance

address: http://localhost:8080
credentials: admin:admin



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/

michal.praszmo@cert.pl
info@cert.pl

XY min to test the local instance and check out extra challenges

- Challenges hosted on https://**ctf.bzcat.eu**
- “extra” category
- Links to projects/docs/slides available at

  https://**bzcat.eu**