Yaraexporter Documentation

Release 1.0.0

Nils Kuhnert

Mar 24, 2017

Contents

Contents:

CHAPTER 1

Yaraexporter module documentation

Yaraexporter

This little program exports attributes (regkey, regkey|value, pattern-in-file and mutex) from misp events and creates yara files that are usable with Thor. Only attributes that are not proposed to delete and marked as 'export to ics' are exported.

This script can be used with different parameters from the cli:

```
usage: yaraexporter.py [-h] -u URL [-k] [-s SSL] -a ATTRIBUTE [-f FILE] [-c]
                       [-d] [-i IGNORE]

Connects to an MISP instance and exports yara rules based on givenattribute
types.

optional arguments:
  -h, --help            show this help message and exit
  -u URL, --url URL     The url of the MISP instance.
  -k, --key             Prompts for the API key. If not given read MISP_KEY
                        from env.
  -s SSL, --ssl SSL     Path to certificate file for validation, if not
                        globally trusted.
  -a ATTRIBUTE, --attribute ATTRIBUTE
                        Which attribute to export. (Currently supported:
                        regkey, pattern-in-file, mutex)
  -f FILE, --file FILE  Path to output file for the yara rules. If not given,
                        rules are printed to stdout.
  -c, --compile         Compile the rules and place *.yas next to FILE.
  -d, --debug           Turn on debug mode.
  -i IGNORE, --ignore IGNORE
                        Comma separated list of events to ignore.

Thanks for using! CERT-Bund, 2017
```

The preferred way to call this script should be:

```
MISP_KEY=Thisisyourmispapikey12345 ./yaraexporter.py [PARAMS]
```

But you're able to call it with the -k param and enter the api key in the cli:

```
./yaraexporter.py -k -u https://your.misp-instance.com -a mutex
Enter API Key for https://your.misp-instance.com:
```

Attribute types

Regkey and Regkey|value

For the registry yara rules, all attributes containig registry relevant values are exported. While doing so, the hive part (e.g. HKEY_LOCAL_MACHINE etc.) are cutted, because Thor loads them seperately. Also, the rules containing 'registry' in the name. Apart from that, the rules are similar to the normal pattern-in-file rules.

Mutex

Mutex rules include the parameter 'limit = "Mutex"' in the meta part of the rule to select them for mutex enumeration.

Pattern-in-file

Nothing special here. Remember not to use too generic values in misp.

Class functions

class yaraexporter.**Yaraexporter**(*url: str*, *key: str*, *ssl: typing.Union[bool, str] = True*, *debug: bool = False*, *ignore: typing.Union[str, NoneType] = None*)

Creates a pymisp instance to fetch given attributes and create yara rules from it. This can also be imported to other modules.

Parameters

- **url** – Url to MISP instance.
- **key** – API key
- **ssl** – True for validation, False to skip or path to self-signed cert.
- **debug** – If set to true, it can be used for locating errors.
- **ignore** – Comma separated list of MISP eventIds to ignore.

_create_mutex_rule(*searchresults: list*) → str

Create mutex rule. Double the strings for ascii and wide search.

Parameters **searchresults** – Results from MISP search (pymisp.PyMisp().search())

Returns yara rules as a string

_create_pattern_rule(*searchresults: list*) → str

Create simple pattern matching yara rule

Parameters **searchresults** – Results from MISP search (pymisp.PyMisp().search())

Returns yara rules as a string

_create_regkey_rule(*searchresults: list*) → str
Create regkey rules. Delete symbols, that are not allowed for yara rules and format it according to thor manual.

Parameters **searchresults** – Results from MISP search (pymisp.PyMisp().search()

Returns yara rules as a string

_debug(*string: str*) → None
For debugging.

_debug_and_stop(*string: str*) → None
For debugging.

_search_for_type(*type_attribute: str*) → list
The actual request to misp. Skip attributes which are proposed for deletion.

Parameters **type_attribute** – MISP event attribute type to search for

Returns List of values per event matching type_attribute

get_rules_for_type(*type_attribute: str*) → str
Sends the request to misp using pymisp api and call the specific rule creation function

CHAPTER 2

Introduction

The APT Scanner Thor can be extended using yara rules containing values to search for in registry, mutexes and filepatterns. To be able to quickly export our MISP database regarding the mentioned values and to create yara rules in the specific Thor format, this tool was created. More information about Thor yara rule formatting can be obtained in their documentation.

Remember to apply the correct keyword to the file name: `registry` for regkey rules and `keyword` for mutex rules which have `limit = "Mutex"` set in the rule itself. Pattern-in-file rules don't need a special filename.

If there are too many false positives, you can always propose attributes to delete, remove the IDS exporting flag or use the ignore parameter.

Be aware that this tool is more a quick&dirty hack than a complete software.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

y

Index

Symbols

G

Y