

---

# **Malware strings analyzer Documentation**

***Release 0.1***

**Yann Ferrere**

**Aug 22, 2016**

## CONTENTS

<b>1</b>	<b>/</b>	<b>1</b>
1.1	msa.py . . . . .	1
<b>2</b>	<b>/lib/</b>	<b>2</b>
2.1	vt.py . . . . .	2
2.2	rules.py . . . . .	3
2.3	progress_bar.py . . . . .	3
2.4	parser.py . . . . .	4
2.5	database.py . . . . .	5
2.6	core.py . . . . .	6
<b>3</b>	<b>/lib/allRules/</b>	<b>7</b>
3.1	cmd.py . . . . .	7
3.2	formatStr.py . . . . .	7
3.3	id.py . . . . .	7
3.4	ipAddr.py . . . . .	8
3.5	message.py . . . . .	8
3.6	path.py . . . . .	9
3.7	section.py . . . . .	9
3.8	symbol.py . . . . .	9
3.9	undefined.py . . . . .	10
3.10	url.py . . . . .	10
	<b>Python Module Index</b>	<b>11</b>
	<b>Index</b>	<b>12</b>

## msa.py

`msa.exit_error()`

Call the `usage()` function and exit `msa.py` with an error return value.

`msa.getopts(argv)`

Parse and extract set options.

**Parameters** `argv` (*List*) – List of parameters set by the user.

**Returns** All valid parameters entered by the user or exit in case of unknown parameter. :rtype: List

`msa.start(argv)`

Analyze users options and run the malware string analyzer.

**Parameters** `argv` – Users options.

**Type** List

**Returns** True if `msa` success, False otherwise.

**Return type** Boolean

`msa.usage()`

Display the usage message of `msa.py`

## vt.py

**class** `lib.vt.Vt` (*api\_key, malware\_path*)

The Vt class is used in order to perform all request to the Virus Total API.

**`__Vt__malware_md5`** ()

Private method that perform the md5 algorithm on the malware binary accessible by using the absolute path on the `malware_path` attribut.

**Returns** MD5 hash of the analyzed malware.

**Return type** string

**`__Vt__print_ip_rslt`** (*json, key, msg*)

Private method that display a green “yes” or a red “no” depending of the results sent by Virus Total.

**Parameters**

- **json** – Returned json from Virus Total API.
- **key** – Json field to analyze.
- **msg** – Description of the Json field currently analyzed.

**`__Vt__send_req`** (*url, parameters*)

Private method that perform a HTTP request to a specific url with parameters.

**Parameters**

- **url** (*string*) – Contains targeted url.
- **parameters** (*dict*) – Contains a list of parameters to be sent.

**Returns** Response of the HTTP request formated in json.

**Return type** dict

**`__init__`** (*api\_key, malware\_path*)

Initialize `api_key` and `malware_path` attributs of the class.

**Parameters**

- **api\_key** (*string*) – The user’s Virus Total API key.
- **malware\_path** (*string*) – The absolute path of the malware binary to analyze.

**`file_analysis`** ()

Perform a malware analysis of the selected malaware binary by using Virus Total API.

**Returns** Notify the success or the failure of this analysis.

**Return type** Boolean

**ip\_analysis** (*ip*)

Perform an ip analysis of a specified ip by using Virus Total API.

**Parameters** **ip** (*string*) – IP to analyze.

**Returns** Notify the success or the failure of this analysis.

**Return type** Boolean

**isActivate** ()

Check if the api\_key has been entered by the user and consequently if the Virus Total analysis has to be executed.

**Returns** If the Virus Total option is activated or not.

**Return type** Boolean

**url\_analysis** (*url\_to\_analyze*)

Perform an url analysis of a specified url by using Virus Total API.

**Parameters** **url\_to\_analyze** (*string*) – Url to analyze.

**Returns** Notify the success or the failure of this analysis.

**Return type** Boolean

## rules.py

**class** `lib.rules.Rules`

This object is inherited by all specific string analysis rules. It also contained an instance of the db object, used by all rules.

**\_\_init\_\_** ()

Initialize the bar attributs that allow all rules to use the ProgressBar object.

**run\_analysis** (*string\_list*)

This method is used by the core to run analyze and extract strings matching with a specific type.

**Parameters** **string\_list** (*List*) – All strings to analyse.

**Returns** A list of string without strings previously matched.

**Return type** List

## progress\_bar.py

**class** `lib.progress_bar.ProgressBar`

This class is used to display a Progress bar to evaluate the progression of the string analysis.

**\_\_ProgressBar\_\_cur\_percentage** ()

Compute the current percentage of string analysis progression.

**Returns** Current percentage value.

**Return type** int

**\_\_ProgressBar\_\_cur\_progression** ()

Build the display of current string analysis progression.

**Returns** String representing the current progression.

**Return type** string

**`__ProgressBar__display_bar()`**

Format and display the progression bar.

**`__init__()`**

Initialize all value needed to display the progress bar.

**`close(str, endedCorrectly)`**

This method is called in the end of the progression bar to clean the display. It also display if the analysis have succeeded or not.

**`init(iterable, init_str)`**

Initialize the progression bar.

**Parameters**

- **`iterable`** (*List*) – List of strings to analyze.
- **`init_str`** (*string*) – String to display in the begin of the progression bar.

**`update()`**

This method increment the progression bar by adding 1 percent.

## parser.py

**`class lib.parser.Parser`**

This class is used to perform all tasks concerning the string analysis.

**`static getCmd(string)`**

Static method that extract strings detected as a linux command line.

**Parameters** **`string`** (*String*) – A string to be analyzed.

**Returns** None if the string doesn't match or the part of the string which corresponds.

**Return type** string

**`static getFormatStr(string)`**

Static method that extract strings detected as a format string.

**Parameters** **`string`** (*String*) – A string to be analyzed.

**Returns** None if the string doesn't match or the part of the string which corresponds.

**Return type** string

**`static getId(string)`**

Static method that extract strings detected as an identifier.

**Parameters** **`string`** (*String*) – A string to be analyzed.

**Returns** None if the string doesn't match or the part of the string which corresponds.

**Return type** string

**`static getIp(string)`**

Static method that extract strings detected as an IP address.

**Parameters** **`string`** (*String*) – A string to be analyzed.

**Returns** None if the string doesn't match or the part of the string which corresponds.

**Return type** string

**static getMessage** (*string*)

Static method that extract strings detected as an english message.

**Parameters** **string** (*String*) – A string to be analyzed.

**Returns** None if the string doesn't match or the part of the string which corresponds.

**Return type** string

**static getPath** (*string*)

Static method that extract strings detected as a file path.

**Parameters** **string** (*String*) – A string to be analyzed.

**Returns** None if the string doesn't match or the part of the string which corresponds.

**Return type** string

**static getSection** (*string*)

Static method that extract strings detected as a elf binary section.

**Parameters** **string** (*String*) – A string to be analyzed.

**Returns** None if the string doesn't match or the part of the string which corresponds.

**Return type** string

**static getUrl** (*string*)

Static method that extract strings detected as an URL.

**Parameters** **string** (*String*) – A string to be analyzed.

**Returns** None if the string doesn't match or the part of the string which corresponds.

**Return type** string

**static isValidBin** (*path*)

Detect if a file is an elf binary.

**Parameters** **path** (*string*) – Absolute path to an elf binary.

**Returns** First value determine if its a binary or not and the second contains the error message in case of error.

**Return type** Boolean, string

**static strings** (*path*)

Extract all readable strings from a binary.

**Parameters** **path** – Path to the binary to analyze.

**Type** string

**Returns** List that contains all strings.

**Return type** List

## database.py

**class** lib.database.**Database** (*malware\_path*)

This class is used to interact with database.

**\_\_init\_\_** (*malware\_path*)

Initialize the database and create a table corresponding to the malware to analyze.

**Parameters** **malware\_path** – Absolute path to the malware binary.

**close** ()

Close the database.

**createEntry** (*string, extract, type*)

This methods is used by all rules to create an entry in the current table.

**Parameters**

- **string** – String that has been analyzed.
- **extract** – Part of the string corresponding to as specific rule.
- **type** – Type of the rule corresponding to the extracted element.

**getIpAddresses** ()

Return all strings detected as an IP address.

**Returns** All IP address in the current table.

**Type** List

**getUrls** ()

Return all strings detected as an URL.

**Returns** All URLs in the current table.

**Return type** List

## core.py

**class** `lib.core.Core` (*malware\_path, vt\_key*)

class core

**\_\_init\_\_** (*malware\_path, vt\_key*)

**run** ()

Entry point of the MSA functions.

**Parameters** **self** (*object*) – blabla

**Returns** If the function success or not.

**Return type** Boolean



**/LIB/ALLRULES/**

## cmd.py

**class** `lib.allRules.cmd.Cmd`

This class is used to extract linux command line from malware binaries strings. It also inherit the Rules object used to gather all shared functions and variables.

**\_\_init\_\_** ()

Initialize type and info\_msg attributes which respectively represent the type of extracted information (here command line - cmd), and the message to display when the rule is initialized.

**run\_analysis** (*string\_list*)

This method is used by the core to run analyze and extract strings matching with the “command line” type.

**Parameters** **string\_list** (*List*) – All strings to analyse.

**Returns** A list of string without strings previously matched.

**Return type** List

## formatStr.py

**class** `lib.allRules.formatStr.FormatStr`

This class is used to extract format strings from malware binaries strings. It also inherit the Rules object used to gather all shared functions and variables.

**\_\_init\_\_** ()

Initialize type and info\_msg attributes which respectively represent the type of extracted information (here format strings - formatStr), and the message to display when the rule is initialized.

**run\_analysis** (*string\_list*)

This method is used by the core to run analyze and extract strings matching with the “format string” type.

**Parameters** **string\_list** (*List*) – All strings to analyse.

**Returns** A list of string without strings previously matched.

**Return type** List

## id.py

**class** `lib.allRules.id.Id`

This class is used to extract identifiers (username/password) from malware binaries strings. It also inherit the

Rules object used to gather all shared functions and variables.

**\_\_init\_\_** ()

Initialize type and info\_msg attributes which respectively represent the type of extracted information (here identifiers - Id), and the message to display when the rule is initialized.

**run\_analysis** (*string\_list*)

This method is used by the core to run analyze and extract strings matching with the “identifiers” type.

**Parameters** **string\_list** (*List*) – All strings to analyse.

**Returns** A list of string without strings previously matched.

**Return type** List

## ipAddr.py

**class** lib.allRules.ipAddr.**IpAddr**

This class is used to extract IP addresses from malware binaries strings. It also inherit the Rules object used to gather all shared functions and variables.

**\_\_init\_\_** ()

Initialize type and info\_msg attributes which respectively represent the type of extracted information (here format ip addresses - IpAddr), and the message to display when the rule is initialized.

**run\_analysis** (*string\_list*)

This method is used by the core to run analyze and extract strings matching with the “ip addresses” type.

**Parameters** **string\_list** (*List*) – All strings to analyse.

**Returns** A list of string without strings previously matched.

**Return type** List

## message.py

**class** lib.allRules.message.**Msg**

This class is used to extract english message from malware binaries strings. It also inherit the Rules object used to gather all shared functions and variables.

**\_\_init\_\_** ()

Initialize type and info\_msg attributes which respectively represent the type of extracted information (here format messages - Msg), and the message to display when the rule is initialized.

**run\_analysis** (*string\_list*)

This method is used by the core to run analyze and extract strings matching with the “message” type.

**Parameters** **string\_list** (*List*) – All strings to analyse.

**Returns** A list of string without strings previously matched.

**Return type** List

## path.py

**class** `lib.allRules.path.Path`

This class is used to extract file path from malware binaries strings. It also inherit the Rules object used to gather all shared functions and variables.

**\_\_init\_\_** ()

Initialize type and info\_msg attributes which respectively represent the type of extracted information (here file path - path), and the message to display when the rule is initialized.

**run\_analysis** (*string\_list*)

This method is used by the core to run analyze and extract strings matching with the “file path” type.

**Parameters** `string_list` (*List*) – All strings to analyse.

**Returns** A list of string without strings previously matched.

**Return type** List

## section.py

**class** `lib.allRules.section.Section`

This class is used to extract binaries elf sections from malware binaries strings. It also inherit the Rules object used to gather all shared functions and variables.

**\_\_init\_\_** ()

Initialize type and info\_msg attributes which respectively represent the type of extracted information (here elf sections - section), and the message to display when the rule is initialized.

**run\_analysis** (*string\_list*)

This method is used by the core to run analyze and extract strings matching with the “elf section” type.

**Parameters** `string_list` (*List*) – All strings to analyse.

**Returns** A list of string without strings previously matched.

**Return type** List

## symbol.py

**class** `lib.allRules.symbol.Symbol` (*malware\_path*)

This class is used to extract binaries elf symbols from malware binaries strings. It also inherit the Rules object used to gather all shared functions and variables.

**\_\_init\_\_** (*malware\_path*)

Initialize type and info\_msg attributes which respectively represent the type of extracted information (here elf symbols - symbol), and the message to display when the rule is initialized. This class also contains a `malware_path` value containing the absolute path of the analyzed binary.

**Parameters** `malware_path` – Absolute path of the analyzed binary.

**Type** string

**run\_analysis** (*string\_list*)

This method is used by the core to run analyze and extract strings matching with the “elf symbols” type.

**Parameters** `string_list` (*List*) – All strings to analyse.

**Returns** A list of string without strings previously matched.

**Return type** List

## undefined.py

**class** `lib.allRules.undefined.Undefined`

This class is used to extract all undefined strings from malware binaries strings. It also inherit the Rules object used to gather all shared functions and variables.

**\_\_init\_\_** ()

Initialize type and info\_msg attributes which respectively represent the type of extracted information (here undefined strings - undefined), and the message to display when the rule is initialized.

**run\_analysis** (*string\_list*)

This method is used by the core to run analyze and extract strings matching with the “undefined” type.

**Parameters** **string\_list** (*List*) – All strings to analyse.

**Returns** A list of string without strings previously matched.

**Return type** List

## url.py

**class** `lib.allRules.url.Url`

This class is used to extract URLs from malware binaries strings. It also inherit the Rules object used to gather all shared functions and variables.

**\_\_init\_\_** ()

Initialize type and info\_msg attributes which respectively represent the type of extracted information (here URLs - url), and the message to display when the rule is initialized.

**run\_analysis** (*string\_list*)

This method is used by the core to run analyze and extract strings matching with the “url” type.

**Parameters** **string\_list** (*List*) – All strings to analyse.

**Returns** A list of string without strings previously matched.

**Return type** List

**I**

- `lib.allRules.cmd`, 7
- `lib.allRules.formatStr`, 7
- `lib.allRules.id`, 7
- `lib.allRules.ipAddr`, 8
- `lib.allRules.message`, 8
- `lib.allRules.path`, 9
- `lib.allRules.section`, 9
- `lib.allRules.symbol`, 9
- `lib.allRules.undefined`, 10
- `lib.allRules.url`, 10
- `lib.core`, 6
- `lib.database`, 5
- `lib.parser`, 4
- `lib.progress_bar`, 3
- `lib.rules`, 3
- `lib.vt`, 2

**m**

- `msa`, 1

## Symbols

\_ProgressBar\_\_cur\_percentage()  
     (lib.progress\_bar.ProgressBar method), 3  
 \_ProgressBar\_\_cur\_progression()  
     (lib.progress\_bar.ProgressBar method), 3  
 \_ProgressBar\_\_display\_bar()  
     (lib.progress\_bar.ProgressBar method), 4  
 \_Vt\_\_malware\_md5() (lib.vt.Vt method), 2  
 \_Vt\_\_print\_ip\_rslt() (lib.vt.Vt method), 2  
 \_Vt\_\_send\_req() (lib.vt.Vt method), 2  
 \_\_init\_\_() (lib.allRules.cmd.Cmd method), 7  
 \_\_init\_\_() (lib.allRules.formatStr.FormatStr method), 7  
 \_\_init\_\_() (lib.allRules.id.Id method), 8  
 \_\_init\_\_() (lib.allRules.ipAddr.IpAddr method), 8  
 \_\_init\_\_() (lib.allRules.message.Msg method), 8  
 \_\_init\_\_() (lib.allRules.path.Path method), 9  
 \_\_init\_\_() (lib.allRules.section.Section method), 9  
 \_\_init\_\_() (lib.allRules.symbol.Symbol method), 9  
 \_\_init\_\_() (lib.allRules.undefined.Undefined method), 10  
 \_\_init\_\_() (lib.allRules.url.Url method), 10  
 \_\_init\_\_() (lib.core.Core method), 6  
 \_\_init\_\_() (lib.database.Database method), 5  
 \_\_init\_\_() (lib.progress\_bar.ProgressBar method), 4  
 \_\_init\_\_() (lib.rules.Rules method), 3  
 \_\_init\_\_() (lib.vt.Vt method), 2

## C

close() (lib.database.Database method), 6  
 close() (lib.progress\_bar.ProgressBar method), 4  
 Cmd (class in lib.allRules.cmd), 7  
 Core (class in lib.core), 6  
 createEntry() (lib.database.Database method), 6

## D

Database (class in lib.database), 5

## E

exit\_error() (in module msa), 1

## F

file\_analysis() (lib.vt.Vt method), 2  
 FormatStr (class in lib.allRules.formatStr), 7

## G

getCmd() (lib.parser.Parser static method), 4  
 getFormatStr() (lib.parser.Parser static method), 4  
 getId() (lib.parser.Parser static method), 4  
 getIp() (lib.parser.Parser static method), 4  
 getIpAddresses() (lib.database.Database method), 6  
 getMessage() (lib.parser.Parser static method), 5  
 getopts() (in module msa), 1  
 getPath() (lib.parser.Parser static method), 5  
 getSection() (lib.parser.Parser static method), 5  
 getUrl() (lib.parser.Parser static method), 5  
 getUrls() (lib.database.Database method), 6

## I

Id (class in lib.allRules.id), 7  
 init() (lib.progress\_bar.ProgressBar method), 4  
 ip\_analysis() (lib.vt.Vt method), 3  
 IpAddr (class in lib.allRules.ipAddr), 8  
 isActivate() (lib.vt.Vt method), 3  
 isValidBin() (lib.parser.Parser static method), 5

## L

lib.allRules.cmd (module), 7  
 lib.allRules.formatStr (module), 7  
 lib.allRules.id (module), 7  
 lib.allRules.ipAddr (module), 8  
 lib.allRules.message (module), 8  
 lib.allRules.path (module), 9  
 lib.allRules.section (module), 9  
 lib.allRules.symbol (module), 9  
 lib.allRules.undefined (module), 10  
 lib.allRules.url (module), 10  
 lib.core (module), 6  
 lib.database (module), 5  
 lib.parser (module), 4  
 lib.progress\_bar (module), 3  
 lib.rules (module), 3  
 lib.vt (module), 2

## M

msa (module), 1  
 Msg (class in lib.allRules.message), 8

## P

Parser (class in lib.parser), 4  
 Path (class in lib.allRules.path), 9  
 ProgressBar (class in lib.progress\_bar), 3

## R

Rules (class in lib.rules), 3  
 run() (lib.core.Core method), 6  
 run\_analysis() (lib.allRules.cmd.Cmd method), 7  
 run\_analysis() (lib.allRules.formatStr.FormatStr method),  
 7  
 run\_analysis() (lib.allRules.id.Id method), 8  
 run\_analysis() (lib.allRules.ipAddr.IpAddr method), 8  
 run\_analysis() (lib.allRules.message.Msg method), 8  
 run\_analysis() (lib.allRules.path.Path method), 9  
 run\_analysis() (lib.allRules.section.Section method), 9  
 run\_analysis() (lib.allRules.symbol.Symbol method), 9  
 run\_analysis() (lib.allRules.undefined.Undefined  
 method), 10  
 run\_analysis() (lib.allRules.url.Url method), 10  
 run\_analysis() (lib.rules.Rules method), 3

## S

Section (class in lib.allRules.section), 9  
 start() (in module msa), 1  
 strings() (lib.parser.Parser static method), 5  
 Symbol (class in lib.allRules.symbol), 9

## U

Undefined (class in lib.allRules.undefined), 10  
 update() (lib.progress\_bar.ProgressBar method), 4  
 Url (class in lib.allRules.url), 10  
 url\_analysis() (lib.vt.Vt method), 3  
 usage() (in module msa), 1

## V

Vt (class in lib.vt), 2