About the Robustness and Looseness of Yara Rules

Gerardo Canfora, Mimmo Carapella, Andrea Del Vecchio, Laura Nardi, Antonio Pirozzi, Corrado Aaron Visaggio

▶ **To cite this version:**

Gerardo Canfora, Mimmo Carapella, Andrea Del Vecchio, Laura Nardi, Antonio Pirozzi, et al.. About the Robustness and Looseness of Yara Rules. 32th IFIP International Conference on Testing Software and Systems (ICTSS), Dec 2020, Naples, Italy. pp.104-120, 10.1007/978-3-030-64881-7_7 . hal-03239822

HAL Id: hal-03239822
https://inria.hal.science/hal-03239822v1

Submitted on 27 May 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

About the Robustness and Looseness of Yara Rules

Gerardo Canfora[1], Mimmo Carapella[1], Andrea Del Vecchio[1], Laura Nardi[1], Antonio Pirozzi[1], and Corrado Aaron Visaggio[1]

DEPARTMENT of ENGINEERING
UNIVERSITY of SANNIO
Benevento, Italy
{canfora,pirozzi,visaggio}@unisannio.it
{m.carapella1,a.delvecchio2,l.nardi}@studenti.unisannio.it

Abstract. The tremendous and fast growth of malware circulating in the wild urges the community of malware analysts to rapidly and effectively share knowledge about the arising threats. Among the other solutions, Yara is establishing as a de facto standard for describing and exchanging Indicators of Compromise (IOCs). Unfortunately, the community of malware analysts did not agree on a set of guidelines for writing Yara rules: a plethora of very different styles for formalizing IOCs can be observed, indeed. Our thesis is that different styles of Yara rule writing could affect the quality of IOCs. With this paper we provide: (i) the definition of two dimensions of Yara rules quality, namely *Robustness* and *Looseness*; (ii) a taxonomy for describing the kinds of IOCs that can be formalized with the Yara grammar, and (iii) a suite of metrics for measuring the quality of an IOC. Finally, we carried out a study on 32,311 Yara rules for examining the different existing styles and to investigate the relationship between the writing styles and the quality of IOCs.

Keywords: Yara, malware classification, threat intelligence, threat hunting.

1 Introduction

The information necessary to recognize and classify malware is captured through the indicators of compromise (IOCs in the remainder of the paper), which are generally divided into three categories [9]: Network, Host-based indicators and Email indicators. An indicator of compromise for a malware is a set of properties regarding the structure or the behavior of a malicious software, which are able to identify a specific malicious software. An IOC is written by an analyst who intercepted and dissected the malware, likely in the early stages of its diffusion in the network. Thus the analyst propagates this IOC through informative channels. So far, many formats for writing IOCs have been proposed and used [11, 3, 6]. Even if none of them has yet been chosen as an official standard, without any doubt we can state that Yara is establishing as the de facto standard in the malware analysts' community. Yara allows analysts to define IOC in the form of

a program, namely a Yara Rule. Different papers in literature mention the quality of an IOC as a relevant issue to face; authors in [10] point two dimensions of an IOC's data quality which need to be evaluated: verifiability and believaibility. Tounsi and Rais [12] observe that a strong limit of IOC is to keep up to date with respect to the high variability of malware. Currently, there is no a defined process for writing a Yara Rule which could take into account these aspects of an IOC quality. Therefore analysts still rely on their own experience and ability. Such a lack of a shared discipline for writing a Yara Rule determines a high variety of styles among the authors. It may be reasonable to hypothesize that the style can have a relationship with the *quality* of the Yara Rules, as it happens in the programming languages. Measuring rule's quality would mean being able to choose the best use case in which to use them. This point is paramount for the implementation of an effective Threat Intelligence Toolchain for classification or hunting tasks and to speed up the whole malware triaging process. There is not a definition of *quality* for a Yara Rule: at the best knowledge of the authors this is the first work that attempts to investigate the aspects determining the *quality* of a Yara Rule. With *quality* of a Yara Rule we mean the ability of the Rule to be accurate in detecting the intended malware. We propose a model for defining and measuring the *quality* of a Yara Rule, namely the $\mathcal{R}$-$\mathcal{L}$ Model. In our model, we consider two aspects of the *quality* of a Yara Rule. The first aspect concerns the *robustness* of the Rule, i.e. richness and diversity of IOC types used to detect the malware. The second aspect regards the *looseness* of the Yara Rule, i.e. the ability to recognize some behaviors or features of a certain malware occurring in other malware. The concept of *looseness* is balanced by the opposite concept of *tightness*. A tight rule is not able to recognize a malware family or a malware behavior, but only an exact malware instance. Finally, we carried out a study on 32,311 Yara Rules, investigating the relationships between the indicators identified with our model and the quality of a Yara Rule. This paper provides for a triple contribution:

- a definition of *quality* of a Yara Rule, declined in the two concepts of *Robustness* and *Looseness*;
- a set of metrics for measuring the *Robustness* and *Looseness*;
- a study that correlates our measurement model with the actual *detection performances* of a Yara Rule.

The paper continues as follows: section 2 discusses the novelty brought by this work with respect to the state of the art. Section 3 explains in depth the definitions of Robustness and Looseness, while section 4 describes how the study was designed and carried out. Finally, results are analyzed in section 5 and the main findings along with the evolution of this work are summarized in section 6.

2 State of the Art

So far, not many efforts have been spent for investigating Yara Rules. However, none till now has directly addressed the problem of quality assessment of Yara

Rules.
At the best of authors' knowledge, the only paper that explores the relationship between Yara Rules and malware analysis is [7], that describes a mechanism to boost the triaging performances of Yara Rules by adding references to *fuzzy hashing* of a specific sample. Scientific research in this area focused mainly on automatizing the process of rules and patterns generation and, consequently, on the automatic generation of IOCs, in order to ease and accelerate the work of malware analysts. Biondi et al. in [1] define a tool with a modular *client/server* architecture which generates new Yara Rules with dynamic and static analysis. Similarly, authors in [5] describe a IDA-Pro plug-in which can identify imported cryptography functions, used by a malware specimen, and generate a Yara hexadecimal signature for each of the captured functions.
Other researchers, instead, in order to achieve quality and resilience of detection, focused on the difference between *Observable Patterns* and *Observable Instances*, as described by [2]. Basically, the difference is that an *Observable Pattern*, also defined as Pattern Based IOC, describes a specific class of IOCs, while an *Observable Instance* is an actual instance of a class. To achieve this goal, the already cited [2] and [4] propose respectively a method and a tool to automatically produce pattern based IOCs, in order to make the detection as reliable as possible, ensuring accuracy, elasticity and interpretability. The proposed study aims to address the quality assessment in a holistic way investigating the key properties of a Yara Rule and how these properties affect the reliability of the Rule in different circumstances.

3 The $\mathcal{R}$-$\mathcal{L}$ model

The $\mathcal{R}$-$\mathcal{L}$ model is aimed at evalutaing the quality of a Yara Rule. It describes two properties of a Yara Rule: the ***Robustness*** ($\mathcal{R}$) and the ***Looseness*** ($\mathcal{L}$).
The ***Robustness*** of a Yara Rule is the capability of matching the intended malware, the intended malware families or the intended malicious behavior. The term "*Robustness*" refers to the desirable property of a Yara Rule to keep valid even when some IOCs of the malware change over time. This may happen in several cases; typical cases include but are not limited to: change in IP and domain of C&C, dropped files with different names, different names for mutexes, different signatures.
The ***Looseness*** of a Yara Rule is the capability of matching some behaviors or characteristics of the intended malware or malware families also in other malware or malware families.
This property allows a Yara Rule to identify malware variants of the intended malware, or malware families the intended malware belongs to. An inherent peculiarity of the malware evolution is the proliferation of malware variants. A malware variant is a modified version of an existing malware. It is important to recognize that a malware is a variant, because it may accelerate the triage stages, increasing the probability of an early intervention. Moreover, the reuse of (parts of) existing malware or the replication of some features in new malware is a

common practice among the malware authors, as every analyst may experience, and that is often documented by the malware analysis reports. If a Yara Rule is able to identify specific behaviors, like a certain evasion technique, a boot survival mechanism, a system for realizing a lateral movement, it could help analysts profile more quickly the malware. Similarly, finding behaviors of known malware families in new malware through a Yara Rule can allow analysts to obtain a more accurate malware phylogeny, i.e. all the malware families the malware derives from. The Robustness of a Yara Rule concerns the *reliability* of its accuracy, i.e. it remains valid even when malware changes. For this reason, we could state that Robustness is a *vertical* capability of a Yara Rule. The Looseness of a Yara Rule concerns the *versatility* of its accuracy, i.e. it can be used to catch features of one malware within other instances. For this reason, we could state that Looseness is a *horizontal* capability of a Yara Rule. Yara syntax offers a malware analyst the opportunity to define many types of IOCs for classifying a malware. Of course, some of them could be *malware-redundant*, i.e. adequate for many instances, besides the malware to match. Others could be *malware-specific*, i.e. they are verified uniquely by the malware to match. In order to evaluate the Robustness, we drew up the catalogue of all the types of IOCs that can be expressed with the Yara language. For each IOC, table 1 and table 2 show the regular expressions which help localize the respective IOCs in the Yara Rule, respectively in the *strings* section and in the *conditions* section.

```
rule APT17_Sample_FXSST_DLL
{

    meta:
        description = "Detects Samples related to APT17 activity - file FXSST.DLL"
        author = "Florian Roth"
        reference = "https://goo.gl/ZiJyQv"
        date = "2015-05-14"
        hash = "52f1add5ad28dc30f68afda5d41b354533d8bce3"

    strings:
        $x1 = "Microsoft? Windows? Operating System" fullword wide
        $x2 = "fxsst.dll" fullword ascii
        $y1 = "DllRegisterServer" fullword ascii
        $y2 = ".cSV" fullword ascii
        $s1 = "GetLastActivePopup"
        $s2 = "Sleep"
        $s3 = "GetModuleFileName"
        $s4 = "VirtualProtect"
        $s5 = "HeapAlloc"
        $s6 = "GetProcessHeap"
        $s7 = "GetCommandLine"

   condition:
        uint16(0) == 0x5a4d and filesize < 800KB and ( 1 of ($x*)
                                        or all of ($y*) ) and all of ($s*)
}
```

Fig. 1. An exemplar Yara Rule.

The $\mathcal{R}$-$\mathcal{L}$ model relies on the assumption that the Robustness and the Looseness of a Yara Rule may depend on: (i) the number of IOCs, (ii) the types of IOCs, (iii) the values of IOCs, and (iv) the conditions used in the Rule.

Given a Rule r, we can define the following metrics:

- the *amount* A_r, which counts the overall number of IOCs used in the Rule,
- the *diversity* D_r, which counts the different types of IOCs used, and
- the *distribution* N_r, which counts the density of each IOC in the rule.

Given the *diversity* of a Yara Rule D_r, the *amount* A_r is measured as:

$$\mathrm{A_r} = \sum_{k=1}^{D_\mathrm{r}} n_k$$

where n_k is the number of IOCs of k type. We define n_k as the *cardinality* of the k-th IOC. The *distribution* of r, N_r, is measured as:

$$N_\mathrm{r} = \frac{D_\mathrm{r}}{A_\mathrm{r}}$$

It must be noticed that:

$$N_\mathrm{r} \in]0, 1]$$

where

$$N_\mathrm{r} = 1 \iff A_\mathrm{r} = D_\mathrm{r}$$

i.e. all the IOCs of the Rule belong to different types.

$$N_\mathrm{r} \neq 0$$

$\forall r$, since it can not exist a Rule without any IOC.

The *Robustness* of r is:

$$\mathcal{R}_\mathrm{r}=(\mathrm{A_r},\, \mathrm{D_r},\, \mathrm{N_r})$$

The *Looseness* of r, $\mathcal{L}_\mathrm{r}$, is:

$$\mathcal{L}_\mathrm{r} = (\mathcal{F}_\mathrm{r},\, \mathcal{M}_\mathrm{r})$$

where $\mathcal{M}_\mathrm{r}$ is the number of malware that match r, while $\mathcal{F}_\mathrm{r}$ is the number of malware families that match r.

As an example, let's consider the Rule r taken from a public repository[1] showed in figure 1. The ***Robustness*** parameters are computed as follows:

[1] https://github.com/Yara-Rules/rules/blob/master/malware/APT_APT17.yar, accessed on 12th March

- $A_r = 11$
- $D_r = 2$
- $N_r = 0.18$

D_r is 2 because $x2 belongs to the category *Dynamic-Link libraries and their functions*, while $x1, $y1-2, $s1-7 belong to the category *plain text strings*. Finally, we can conclude that:

$$\mathcal{R}_{\mathrm{r}}=(11,\ 2,\ 0.18).$$

We can tell that r uses only 2 types of IOCs, with a strong density: as a matter of fact, the amount of IOCs is 11; the high value of A_r suggests that the Rule is reliable, even if the low distribution may weaken the Robustness of r, since variants of new malware could be matched with other types of IOCs. The *Looseness* measures how many correct matches a Rule has, and may be calculated with regards to a reference repository of malware. By submitting r to Hybrid Analysis (one of the two platforms used for the study discussed later), we obtain the following ***Looseness*** parameters:

- $\mathcal{M}_{\mathrm{r}} = 10$
- $\mathcal{F}_{\mathrm{r}} = 36$

Finally, we can conclude that:

$$\mathcal{L}_{\mathrm{r}}=\ (10,\ 36)$$

The Rule r shows a good versatility, as it allows to match 10 different malware and 36 different malware families.

4 The Study Design

The goal of this study is to investigate the diversity of the styles adopted by the malware analysts for writing Yara Rules and the impact of the different styles on the quality of Yara Rules. The quality of Yara Rules will be evaluated with the $\mathcal{R}$-$\mathcal{L}$ model. The study is structured in three Research Questions:

- **RQ$_1$: which are the styles adopted by the authors of Yara Rules?** This RQ is aimed at profiling the main approaches used for writing Yara Rule in order to identify good and bad practices.

- **RQ$_2$: which IOCs are mostly used in versatile Rules?** This RQ is aimed at investigating which styles can be found in rules with high values of $\mathcal{M}_{\mathrm{r}}$ and $\mathcal{F}_{\mathrm{r}}$.

- **RQ$_3$: is Robustness correlated with the Looseness of a Yara Rule?** This RQ is aimed at understanding whether, given a Yara Rule r, A_r, N_r, and D_r are related to the $\mathcal{M}_{\mathrm{r}}$ and $\mathcal{F}_{\mathrm{r}}$.

Table 1. List of the IOCs used for evaluating the Robustness of a Yara Rule.

IOC	Description	Regular Expression
Uniform Resource Locators	Captures Yara Rule's strings referring to URL.	`([A-Za-z]+://)([-\w]+(?:\.\w[-\w]*)+)(:\d+)?(/[^.!,?"<>\[\]{}\s\x7F-\xFF]*(?:[.!,?]+[^.!,?"<>\[\]{}\s\x7F-\xFF]+)*)?`
Dynamic-Link Libraries and their functions	Captures Yara Rule's strings referring to dll and their functions.	`.*\.dll\s*`
Win32 Library	Captures Yara Rule's strings referring to dll Win32 functions.	`^[A-Z]\w{2,10}([A-Z](\w){3,10}){1,6}$`
IP Addresses (IPv4)	Captures Yara Rule's strings referring to IP addresses v4.	`(?:(?:\d|[01]?\d\d|2[0-4]\d|25[0-5])\.){3}(?:25[0-5]|2[0-4]\d|[01]?\d\d|\d)(?:\/\d{1,2})?`
IP Addresses (IPv6)	Captures Yara Rule's strings referring to Ip addresses v6	`((?=.*::)(?!.*::.+::)(::)?([\dA-Fa-f]{1,4}:(:|\b)|){5}|([\dA-Fa-f]{1,4}:){6})(([\dA-Fa-f]{1,4}((?!\3)::|:\b|(?![\dA-Fa-f])))|(?!\2\3)){2}|(((2[0-4]|1\d|[1-9])?\d|25[0-5])\.?\b){4})`
Domains	Captures Yara Rule's strings referring to Domain Names.	`\b((?=[a-z0-9-]{1,63}\.)(xn--)?[a-z0-9]+(-[a-z0-9]+)*\.)+[a-z^(exe)^(dll)]{2,63}(?<!exe)$(?<!dll)$(?<!png)$(?<!html)$(?<!htm)$(?<!ini)$(?<!bin)$(?<!php)$(?<!pdb)$(?<!dat)$(?<!sys)$(?<!txt)$(?<!log)$(?<!asp)$(?<!tmp)$(?<!xml)$(?<!js)$(?<!sh)$(?<!gui)$(?<!bmp)$(?<!dbg)$(?<!db)$(?<!apk)$(?<!asp)$(?<!pdf)$(?<!jpg)$(?<!dex)$(?<!qrk)$(?<!cgi)$(?<!manifest)$(?<!wc)$(?<!rtf)$(?<!gif)$(?<!lz)$(?<!nse)$(?<!plist)$(?<!sh)$(?<!ocx)$(?<!key)$(?<!gho)$\b`
Unix file Paths	Captures Yara Rule's strings referring to Unix systems file paths.	`(?:/[A-Za-z\d.][A-Za-z\d\-.]{0,61})+`
Windows file Paths	Captures Yara Rule's strings referring to Windows systems file paths.	`([A-Za-z]{1}:\\{1,2})?(\\{1,}[A-Za-z0-9]+)(\\{0,}[A-Za-z0-9]+\\{0,})+\\{0,}`
Email addresses	Captures Yara Rule's strings referring to email addresses.	`\b(\w[-.\w]*)@([-\w]+(?:\.[-\w]+)*)\.([A-Za-z]{2,4})\b`
Generic files	Captures Yara Rule's strings referring to files.	`.*\.[a-z]{3,4}\s*`
Portable Executable files	Captures Yara Rule's strings referring to *pe* files.	`[A-Za-z0-9_-]*\.exe\s*`

XML/HTML tags	Captures Yara Rule's strings referring to XML and Html tags, distinguishing among generic tags and open tags.	For open tag:
		`^\<[^\>]*\>[A-Za-z\d-:]*$`
		For generic tag:
		`<[^<='\"\\0-9>](\s|\w)+>?(</(\s|\w)+>?)?`

Registry keys	Captures Yara Rushowed in figurele's strings referring to registry keys usage.	`((HKEY_LOCAL_MACHINE|HKLM|HKCU)\\.*)`
SQL code fragments	Captures Yara Rule's strings referring to SQL commands and code fragments.	`(?:INSERT\sINTO|SELECT|DELETE\sFROM|CREATE\s(TABLE|DATABASE)|DROP\s(TABLE|DATABASE)|FOREIGN\sKEY|GROUP\sBY|IF\sEXISTS|UNION)\s+(\s|[^;])+;*$`

Hash	Captures Yara Rule's strings referring to hash, in the form of MD5, SHA1 and SHA256.	MD5:
		`^(0[x|X])?[A-Fa-f0-9]{32}$`
		SHA1:
		`^(0[x|X])?[A-Fa-f0-9]{40}$`
		SHA256:
		`^(0[x|X])?[A-Fa-f0-9]{64}$`

Universally unique identifiers	Captures Yara Rule's strings referring to UUID (format: 8-4-4-4-12).	`[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}`
MAC Addresses	Captures Yara Rule's strings referring to MAC addresses.	`(([0-9A-F]{2}|[\%0-9a-z]{2,4})[:-]){5}([0-9A-F]{2}|[\%0-9a-z]{2,4})`

Http headers	Captures Yara Rule's strings referring to generic Http headers, including the first Http request string and the user-agent header field.	Generic:
		`^((\w{1}-\w{3,})|(\w{2,})|(\w[\w-]+)):\s+\w(\s|\w)+`
		Regex for first http request string and for user-agent header field:
		`(POST)|(GET)|(PUT)|(.?HTTP/\d\.\d)|((User-Agent:)?(Mozilla)|([A-Za-z0-9]*)/\d\.\d)`

Http payload	Captures Yara Rule's strings referring to Http payloads.	`^&?(([A-Za-z0-9\%./\[\]]{2,}=[^\s;,<>\"\\/=]*)&?)+`
Generic code fragments	Capture Yara Rule's strings refferring to code fragments.	`[^(<>&\*\%+\\/!\?)]+\((.*)\)?`
Strings connected to keylogging	Captures Yara Rule's strings referring to keylogging.	`^\[\w[^:,();\[\]!'.\\\"{}=_]*\]$`
Shell commands or references	Captures Yara Rule's strings related to shell commands or references to them.	`(CMD.EXE|POWERSHELL|SHELL)`
Environment Variables	Captures Yara Rule's strings referring to environment variables.	`\%[A-Za-z_]{2,}\%`
Hexadecimal Strings	No regular expression was considered to retrieve information about this field, thanks to the support of Plyara, wich is able to directly identify hexadecimal strings.	-
Regular expressions	No regular expression was considered to retrieve information about this field, thanks to the support of Plyara, wich is able to directly identify strings in the form of regular expression.	-

YARA modules used	Captures references to Yara modules' used, with a special focus on *pe* module and its functions.	For generic module:
		`(\w+\.{1}\w+(\.{1}\w+)*((\([\w*.+/\\s\"=,.:?|]*\))|(\s?(==)\s?[\w*.+/\\s\"=,.:?]*)))`
		For *pe* module:
		`(\w+\.{1}\w+(\[[\w*.+/\\s\"=,()]*\])?\s?(\.{1,}\w+)*(\([\w\\s\"=,.]*\))?\s?((?!(and)|(or))\w+)?\s?((==)?\s?((\"[\w\\s\=,:/.]*\")|([\w\=,:/.]*)))?)`

Plain text strings	If a certain string did not match any of the described regular expressions, it was considered as a plain text string.	-
Base64 Strings	Captures Yara Rule's strings written in Base64 encoding scheme.	`^([A-Za-z0-9+\/]{4})*([A-Za-z0-9+\/]{2}={1,2}$|[A-Za-z0-9+\/]{3}=$)`

Table 2. List of the Conditions used for evaluating the Robustness of a Yara Rule.

Condition	Regular Expression
and conditions	`.+?\sand\s.+?`
or conditions	`.+?\sor\s.+?`
not conditions	`.*?not\s.+?`
filesize conditions	`.+filesize.?`
all of ... conditions (e.g. all of them or $string*)	`all\sof\s(them|\(\$.*?\))`
any of ... conditions (e.g. any of them or $string*)	`any\sof\s(them|\(\$.+?(,\$.+?)*?\))`
Subset of strings conditions (e.g. 2 of them)	`\d+\sof\s(them|\(\$.+?(,\$.+?)*?\))`
Entire set of strings conditions (e.g. $string*)	`\${1}\w*\*{1}`
Offset conditions (e.g. $string1 at 200):	`.+at\s(0(x|X)[0-9a-fA-F]+|\d+|\(.+?\))`
Range conditions (e.g. $string1 in (0..filesize))	`.+in\s\((\d+|.+?)\.{2}(\d+|.+?)\)`
Position conditions (e.g. uint16(0) == ...)	`u?int(8|16|32)(be)?`
Serial number of certificates	`pe\.signatures).*serial.*==.*`
Issuer/subject conditions	`(pe\.signatures).*(issuer|subject).*(contains).*`

4.1 The Dataset

The dataset used in the study contains 32,311 Rules collected from the following sources:

- a catalogue of Yara Rules written by McAfee ATR Team[2];
- the InQuest awesome Yara collection[3];
- a catalogue of Yara Rules publicly available on github repository[4]
- the VALHALLA collection[5];
- the YOROI ZLab YARA Rules collection[6].

Because of the many different versions of Yara used by the authors of the rules, not all the collected Rules were compliant with the process of gathering and analysis. Moreover, some Rules were affected by typos, which prevented them from being scanned. For this reason, 51 files[7] were discarded.

4.2 Data Collection

In order to evaluate how many malware and how many malware families were matched by each Rule in dataset, we submitted the Rules to two of the most

[2] https://github.com/advanced-threat-research/Yara-Rules, last access on 23rd march 2020
[3] https://github.com/InQuest/awesome-yara, last access on 23rd march 2020
[4] https://github.com/Yara-Rules/rules, last access on 23rd march 2020
[5] https://www.nextron-systems.com/yara-rule-feed/, last access on 23rd march 2020
[6] https://yoroi.company/research/, last access on 23rd march 2020
[7] It must be recalled here that a file may contain more than one rule.

popular platforms for threat intelligence: Hybrid Analysis[8] (HA from here on) and VirusTotal[9] (VT from here on). In particular, Hybrid Analysis provides a list of files containing the matched instances of malware, the malware families, and the indicators that caused the match. Moreover, for each submitted Rule, we identified:

- First Sample: submission date of the least recent report returned for a certain Rule
- Last Sample: submission date of the most recent report returned for a certain Rule.

We referred VirusTotal mainly as an oracle, in the sense that we gathered from this platform a variety of information to check the accuracy and the consistency of that collected from Hybrid Analysis.

4.3 Information Gathering process for the Looseness

The indicators contained in a Rule were extracted through a process organized in a sequence of tasks. First, every Yara Rule of the dataset was parsed by Plyara [8] in order to transform each Rule in the form of a Dictionary data structure, i.e. a sequence of key-value pairs, where the key is the name of the Rule and the value is a string containing the body of the Rule. For each Rule, a Retrohunting task on HA was performed, in order to find and collect samples that were matched by that Rule. The submission of Yara Rules on HA and the activation of a Retrohunting task were automated by a web scraper written with Python[10] by the authors.
For each sample identified in the previous step, VT was queried through its public REST API, by submitting the hash of the malware, in order to obtain the information of interest. Then, for each sample, classification signatures from the different AVs were retrieved.

4.4 Information Gathering process for evaluating the Robustness

The collection process for each Rule was based exclusively on Plyara and the script described in the previous section. A set of regular expressions was defined to identify the different kinds of indicators occurring in the Rule. By using the Plyara functionality, every file of the dataset was parsed and its rules were disassembled into their basic elements, i.e. `metadata`, `strings` and `conditions`. For each Rule, Plyara returned a dictionary which contained data for type, name and value of a string. This value was compared to each regular expression in order to find a specific pattern, thus allowing the classification of each string.

[8] https://www.hybrid-analysis.com/, last access on 13th March 2020
[9] https://www.virustotal.com/gui/, last access on 13th March 2020
[10] https://www.python.org/

5 Results

This section discusses how the study's results may help to reply to the three Research Questions.

5.1 Results for RQ_1: which are the styles adopted by the authors of Yara Rules?

To get the picture of the styles adopted by the analysts for writing the Yara Rules, we can examine the distributions of the *amount*, *diversity*, and *distribution* that we have characterized through the following parameters of descriptive statistics (see table 3): the maximum, the minimum, the mean, the median, the first and the third quartile, the mode, and the standard deviation.
The first observation is that the most Rules are very light, as the 75% of the dataset contained less than two indicators with a distribution of 1, and that this combination represents also the most common one, as the mode parameter shows; the 25% of the dataset has only one IOC. We should mention that there are some Rules that are exceptionally overloaded with IOCs, as the maximum value measured is 1361.

Table 3. The Results of RQ_1

Amount	Diversity	Distribution	Parameter
1361	11	1	max
0	0	0	min
2.929	1.826	0.866	mean
2	2	1	median
1	1	0.8	1qt
2	2	1	3qt
2	2	1	mode
11.24	0.89	0.25	st dev

However, the variability of the amount is not that huge ($\sigma_{amount} = 11.24$), but it is very low for the diversity ($\sigma_{diversity} = 0.89$), which confirms that malware analysts tend to use a very limited number of IOC types (one or two, as previously observed), and when the number of IOCs is greater than 2, it tends to keep small, except for very rare cases. We could distinguish between *light* Rules, i.e. the ones containing a few parameters (2 ± 1) and *overloaded* Rules, which contain many parameters (≥ 13). Two indicators mainly contribute to the maximum outliers, more precisely *domains* and *hexadecimal strings*, which means that overloaded Rules are due to an excessive number of domains and signatures. Mode, median and third quartile are zero for all the IOCs except for *hexadecimal string*, for whose the value of the three parameters is 1. This means that the distribution of IOC types is not uniform in the dataset, while the *hexadecimal string* is the most frequent IOC. Even if the mean is a tricky

parameter as it is influenced by extreme values, it can be used to confirm the previous interpretations: the *hexadecimal string* appears in pretty every Rule ($\mu > 0.9$), followed by the plain text stings *text* ($0.8 < \mu < 0.9$). Except for the *Module*, which appears in average in the half of Rules, and *Windows Path* (μ slightly greater than 0.1), all the other indicators have a frequency smaller than 0.1, thus they are used in (less than) 1 Rule out of ten. It is interesting to observe that the IOCs spanning from *HashMD5* to *IPV6 Address* are used a negligible number of times in the sample. Figure 2 shows that *domains* and *hexadecimal strings* have the highest variability, as expected by the very high values of the maximum for these indicators. Except for *Text*, *Module*, *PE File* and *Path Windows* which have a $1 < \sigma < 3$, all the other IOCs show a standard deviation much under 1, which means that their frequencies do not change a lot in the dataset.

RQ_1 Summary: *On average a Rule contains one or two types of indicators. The distribution of IOC types is not uniform among the Rules. The most widespread IOC is the hexadecimal string, while domain is the main cause for overloaded Rules.*

5.2 Results for RQ_2: which IOCs are mostly used in versatile Rules?

In order to understand which IOCs mainly contribute to the detection for a Yara Rule, we computed parameters of descriptive statistics[11] for those Rules which reported at least a true positive, i.e. a number of matches ≥ 1. 15,162 Rules out of 32,311 did not produce any match with existing malware, corresponding to the 47% of the dataset. This result may depend on several factors. First of all, many Rules are created and distributed by International Organizations such as national CERT, or by private companies that deal with Incident Response activities which provide descriptions, in the form of Yara Rules, about threats which were not necessarily been disclosed on public sandboxes. Another explanation may be that some Rules, regard malware specific for platforms such as MacOSX (i.e. *OSX_MacSpy.yar*) or Android (i.e. *Android_Dendroid_RAT.yar*) that are not very popular on these public sandboxes. Finally, there are also Rules which contain signatures for specific packers (i.e. *peid.yar*) that are not widely used. It is worth observing the high values of the maximum for *domains* (almost 1,400) and for *hexadecimal strings* (1,000), while for a not negligible subset it is over 30, however. Interestingly, *domains* are not that used among successful Rules.

[11] These parameters are: the maximum, the minimum, the mean, the 1st and the 3rd quartile, the median, the standard deviation, and the mode

Fig. 2. The values of the Standard Deviation for each IOC

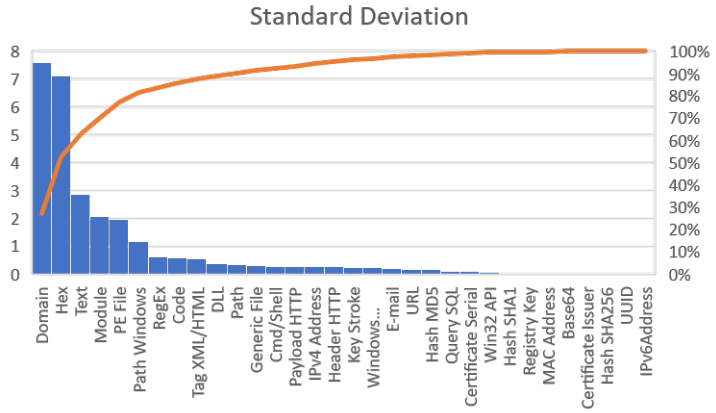


Figure 3 tells that *hexadecimal strings* are used more or less in all the Rules. With the only exception of *Module* and *text strings* with a $0.6 < \mu < 0.7$, all the other IOCs have values ≤ 0.2, that means they can be found, in average, in 1 successful Rule out of 5. It emerges that some IOCs are never used in Yara Rules, as well as: `Win32API`, `Hash SHA1`, `Base64`, `RegistryKey`, `HashSHA256`, `MAC Address`, `UUID`, `IPv6Address`, `Certificate Serial`, and `Certificate Issuer`.

The variability of *amount* in successful Rules remains under a threshold of 0.5 for most indicators, as figure 4 suggests. *Domains* and *hexadecimal strings* have the highest variability ($8 < \sigma < 11$), while a small subset has a moderate variability ($1.5 < \sigma < 2$). As shown in Figure 2, Rules with non-zero matches are heavier, as the strong differences in variability of IOCs suggest (see figure 2).

The mode is 0 for all the indicators, but *hexadecimal strings* (for which it is 1). This means that the most Rules have hexadecimal strings, while all the other indicators are very sparse in the dataset. The same happens for the median and the 3rd quartile. The minimum value and the 1st quartile are overlapped on zero.

RQ_2 Summary: *In Rules with non-zero matches, the most widespread indicators are strings, both textual and hexadecimal, and modules. Non-zero matches Rules are heavier than zero matches rules, with a variability of amount for IOCs slightly over 3. Many indicators are never used, while domains, which have the highest number of occurrences in the entire dataset, resulted the most used IOC also in non-zero matches Yara Rules.*

Fig. 3. The values of the Mean for each IOC of the Rules with non-zero matches

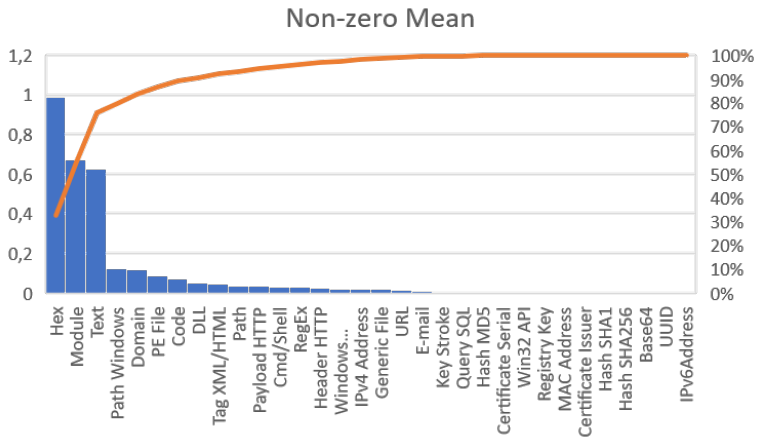
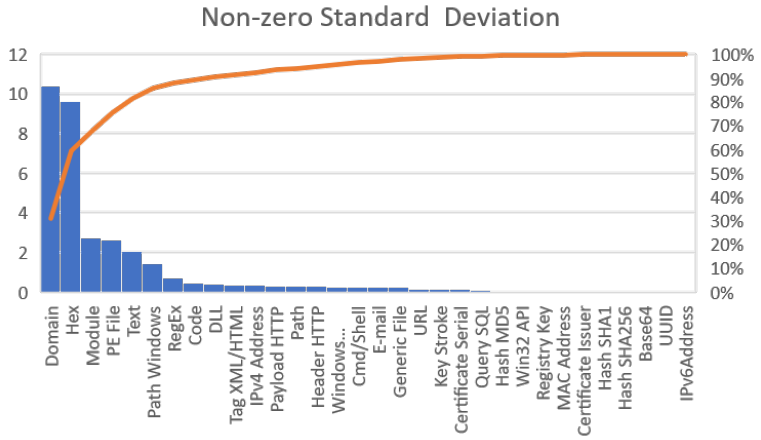


Fig. 4. The values of the Standard Deviation for each IOC of the Rules with non-zero matches



5.3 Results for RQ_3: is Robustness correlated with the Looseness of a Yara Rule?

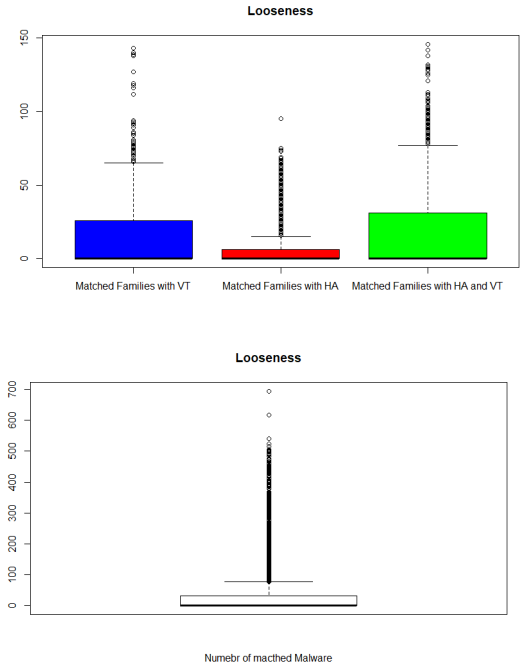
Data (table 4) tell us that half of the Rules have not matches with any malware (and, thus, with malware families). But, the population of the Rules under the 3rd quartile matches with 15 malware and 22 malware families. Having a look at the maximum values, we can find a huge Looseness of Rules, which are able to identify up to 695 malware instances, and 143 malware families.

The images of the corresponding boxplots (figures 5 can help to get the picture of how the values are distributed in the dataset. The most part of the

Table 4. The Parameters of the Distributions of the malware and families matched by Rules

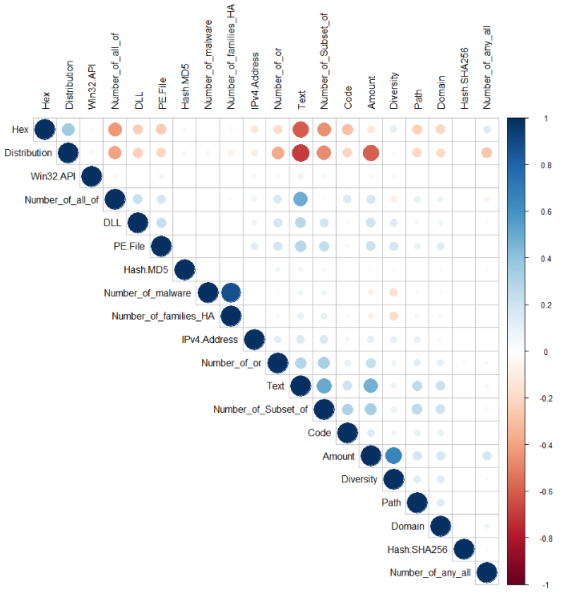
Parameter	Number of Malware	Number of Families
Mean	19.82	10.13
Standard Deviation	54.88	15.57
Min	0	0
1st Quartile	0	0
Median	0	0
Mode	0	0
3rd Quartile	15	22
Max	695	143

Fig. 5. The Box Plots of the number of matched families by Rule obtained on the different platforms, and of the number of matched malware by Rule.



Rules are extremely *tight* (half of the dataset have no matches at all), while a minority of the Rules are *loose*. In contrast, the *loose* Rules show a huge *Looseness*, considering the high number of outliers rising from the maximum bar in the boxplot. A similar observation can be made for the matching with the malware families, but with much smaller numbers, as expected. Let's recall here that the Looseness of a Rule is the ability of detecting a characteristic or a strain of a malware (family); a loose Yara Rule may help detect a new sample or track a new malicious campaign (threat hunting). Finally, Kendall non-parametric correlation has been computed among IOCs, conditions constraints and both Number of Malware and Number of Families. It has been decided to apply the Bonferroni method to adjust p-values in order to reduce the Type I error. The goal of the correlation analysis is to obtain statistical evidence of the correlation between the IOCs, the Robustness (*amount, diversity and distribution*) and the Looseness (*Numb of Malware, Number of Families*).

Fig. 6. Upper Triangular correlation matrix



Then, the resulting correlations, in the form of upper triangular matrix, are shown in figure 6. In order to estimate the effect size of each correlation, the Cohen standard has been adopted. About Looseness, there is statistical evidence for *Num of Malware* and *diversity* (small negative effect size) and *Number_of_families_HA* and *diversity* (small negative effect size) as shown in figure 6. Moreover, it is observed that there is not statistical evidence between Looseness and condition's constraints (*Number of OR, Number of AND, Number of Any Of, Number of All Of, Number of Subset Of*). Based on these observations,

it is possible to conclude that Looseness does not depends on the amount or distribution of the IOCs but weakly depends on the diversity. This basically means that, Yara Rules with a minor diversity are better and more suitable for threat hunting activities.

RQ$_3$ Summary: *The most part of the Rules are tight. The loose Rules, which are a small portion of the examined dataset, are exceptionally powerful, as they are able to match numerous instances of malware and malware families. Unfortunately, no correlation has been observed between IOCs and Looseness and between Looseness and condition's constraints, but only a weak correlation between Looseness and the diversity.*

6 Conclusion

In this paper we propose a model, namely the $\mathcal{R}$-$\mathcal{L}$ model for defining and measuring the *quality* of Yara Rules in two dimensions: the Robustness and Looseness. The Robustness of a Yara Rule is the capability to match the intended malware even if some IOCs change over time. The Looseness of a Yara Rule is the capability of matching variants of the intended malware and regards the versatility of its accuracy. To validate the proposed model we first collected a dataset composed of 32,311 Yara Rules from different public repositories and proved them on two threat intelligence platforms: Hybrid Analysis and VirusTotal. Our results demonstrate that:

(i) Most Yara Rules have a small diversity, but a limited subset is overloaded with many IOCs;
(ii) Looseness is a very widespread property of Yara Rules;
(iii) the correlation between Robustness and Looseness is weak.

Yara is an adopted standard in Threat Intelligence community but there are not accepted shared guidelines. These findings could be useful for integrating Yara in any process of Intelligence creation. It is possible to introduce these metrics in a recommendation tool for writing Yara signatures for specific use cases, like malware hunting, classification, detection, and sharing knowledge. A finding which can drive the writing of a Yara rule is that no correlation has been observed between Looseness and Robustness; this means that the adequacy of a rule for proactively finding new malware variant does not depend on the number or types of IoC's therefore probably from a semantic dimension of the rule.

As future work, we will also investigate more in-depth the usage of compound conditions within a Rule and how styles differ among the Rules for diverse platforms.

References

[1] Fabrizio Biondi, François Dechelle, and Axel Legay. "MASSE: Modular Automated Syntactic Signature Extraction". In: *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE. 2017, pp. 96–97.

[2] Christian Doll et al. "Automated Pattern Inference Based on Repeatedly Observed Malware Artifacts". In: *Proceedings of the 14th International Conference on Availability, Reliability and Security*. 2019, pp. 1–10.

[3] Panos Kampanakis. "Security automation and threat information-sharing options". In: *IEEE Security & Privacy* 12.5 (2014), pp. 42–51.

[4] Yuma Kurogome et al. "EIGER: Automated IOC Generation for Accurate and Interpretable Endpoint Malware Detection". In: *Proceedings of the 35th Annual Computer Security Applications Conference*. ACSAC '19. San Juan, Puerto Rico: Association for Computing Machinery, 2019, pp. 687–701. ISBN: 9781450376280. DOI: 10.1145/3359789.3359808. URL: https://doi.org/10.1145/3359789.3359808.

[5] Han Seong Lee and Hyung-Woo Lee. "Automatic Detection of Cryptographic Algorithms in Executable Binary Files using Advanced Code Chain". In: *2019 International Conference on Green and Human Information Technology (ICGHIT)*. IEEE. 2019, pp. 103–107.

[6] Robert A Martin. "Making security measurable and manageable". In: *MILCOM 2008-2008 IEEE Military Communications Conference*. IEEE. 2008, pp. 1–9.

[7] Nitin Naik et al. "Augmented YARA rules fused with fuzzy hashing in ransomware triaging". In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2019, pp. 625–632.

[8] *Plyara*. Available at https://github.com/plyara/plyara (04/04/2020).

[9] J Ray. *Understanding the Threat Landscape: Indicators of Compromise (IOCs)*. 2015.

[10] Christian Sillaber et al. "Data quality challenges and future research directions in threat intelligence sharing practice". In: *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*. 2016, pp. 65–70.

[11] J. Steinberger et al. "How to exchange security events? Overview and evaluation of formats and protocols". In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 2015, pp. 261–269.

[12] Wiem Tounsi and Helmi Rais. "A survey on technical threat intelligence in the age of sophisticated cyber attacks". In: *Computers & security* 72 (2018), pp. 212–233.