

YValidator: a flexible tool for fetching and validating Indicators of Compromise

CA' FOSCARI UNIVERSITY OF VENICE

Department of Environmental Sciences, Informatics and Statistics



Computer Science Master's Thesis

Year 2020-2021

Graduand Davide Narder

Supervisor Riccardo Focardi

Contents

1	Introduction	1
2	Malware Information Sharing Platform	3
2.1	Data model	4
3	YValidator - Design and Implementation	7
3.1	Sources	9
3.2	Validation	12
3.2.1	Pre-Validation	14
3.2.2	External Validation	16
3.3	QRadar Validation	24
3.4	MISP Feed Generation	28
4	Validation Statistic and Testing	31
5	Conclusions and Future implementations	36

Chapter 1

Introduction

In this document we will describe the design and development of YValidator, a tool built in collaboration with Yarix S.r.l. This project is an attempt to automate and speed up the collection and verification of Indicators of Compromise (IOC) and to limit manual intervention. It is designed to regularly fetch data from selected sources that share IOCs such as Github repositories, Twitter profiles, security blogs, etc. The first chapter will briefly describe MISP (Malware Information Sharing Platform), a platform to organize and share malware information, with a focus on indicators of compromise as the main output of the program is a MISP type feed. A feed can be seen as a stream of new indicators grouped and divided in events to be added in a MISP instance.

Indicators are fetched and parsed with regular expressions and then have to go through a series of verification steps where a partial score and weight is assigned at every step and at the end a decision is made on the validity and maliciousness of the indicators. Then, as a final check, the last step consists of searching the alleged malicious indicators in various QRadar servers to gather information on the practical utility, feasibility and technical problems that can arise by the inclusion of indicators in the MISP instance. QRadar is a Security information and event management (SIEM) with many features but the relevant aspect to the tool developed is that it will gather logs from all the devices of an organization providing a complete view of application usage and network traffic. This step is useful because all the QRadar servers will raise an alert when an indicator in the MISP instance is found in any activity the SIEM is monitoring and by performing a search on the most recent logs before adding the events in MISP we can get

an idea on the number of alerts raised based on the frequency or absence of the searched IOC. This information is used to make a decision on whether to include specific indicators in the feed or to directly send a notification to the company managing the QRadar server when the search suggests a potential malware infection to avoid automatically raising unnecessary alerts on scenarios where manual intervention is preferred.

Finally events are generated grouping indicators based on source, they are tagged accordingly based on the tool configuration and are written in a MISP feed for easy ingestion in MISP instances. The initial and only output designed to be implemented was the MISP feed, but during the project creation other uses and useful interaction were added. Such as the possibility of using to program as a local indicator validation tool by excluding the first and last phase of online source fetching and event generation and to output a detailed report with all the information and related weight used to compute a final scoring. Additionally the report can be used in situation where the scoring system becomes for any reason ineffective and weight and score rebalancing has to be done. The forth chapter contains a report of a series of test for the accuracy of the scoring system of the validation phase together with the main problems and limitation the tool. Consideration are made regarding all the possible scenarios of failure of the tool, such as insertion of false positives and deletions of true positives with all the logic in place to prevent those results.

Chapter 2

Malware Information Sharing Platform

Malware Information Sharing Platform (MISP) is an open source threat intelligence platform. The project was started by an individual malware analyst and then picked up by NATO to become not only a malware Indicator of Compromise platform but also including fraud and vulnerability information.

MISP is used today by many companies to gather, share and correlate indicators of compromise regarding targeted attacks, threat intelligence, financial fraud information and vulnerability information. MISP offers many useful functionalities for an efficient malware analysis such as the ease of sharing data with other MISP instances and automated exports for Intrusion Detection System (IDS) or Security Information and Event Management (SIEM). The MISP philosophy is centered around sharing and automation. It is designed and meant to be used to enhance and simplify the sharing and flow of information to avoid situation in which analysts are working on something that has already been studied and allows user to see a more complete view and bigger picture by linking together related malware or related malware campaigns as it is quite often the case that similar organisations are targeted by the same threat actor or by the same malware, in addition to the ability to create a discussion per event to enable conversations between multiple analysts. Information sharing also comes in the form of integration with popular IDS and SIEM by generating IDS rules, text or csv exports to allow automatic import of indicators in other security systems resulting in better and faster detection of intrusions.

2.1 Data model

The core data model of MISP is the Event. Each event has a series of general information such as event description, date, threat level and can have zero or more attributes. Attributes contain all the indicator of compromise related to the event they belong, but they can also contain non technical data, such as PDF reports, text comments, externals links, etc. Events are usually created and added manually by malware analyst, researchers or forensic teams. When studying malware MISP allows you to store all the analysis data as attributes and provides automatic correlation to find relationships between attributes of other events with the option to disable correlation on desired attributes, usually keeping it enabled by default only for technical data. Together with simple attributes more complex MISP objects can be used to express more complex relationships, events and connected elements. [4]

Figure 2.1: Example of event with attributes in the MISP web interface

OSINT - New Cyber Operation Targets Italy: Digging Into the Netwire Attack Chain

Event ID	1180
UUID	5ede1810-6cfc-4a01-adb0-470902de0b81
Creator org	CIRCL
Owner org	ORGRNAME
Creator user	admin@admin.test
Tags	type:OSINT x osint:lifetime="perpetual" x osint:certainty="50" x tlp:white x misp-galaxy:rat="Netwire" x
Date	2020-06-08
Threat Level	Low
Analysis	Completed
Distribution	All communities
Info	OSINT - New Cyber Operation Targets Italy: Digging Into the Netwire Attack Chain
Published	Yes (2021-03-23 09:06:01)
#Attributes	27 (6 Objects)
First recorded change	2020-06-08 10:51:11
Last change	2020-06-08 10:59:18
Modification map	
Sightings	0 (0) - restricted to own organisation only

Date	Org	Category	Type	Value	Tags	Galaxies	Comment
2020-06-08		Persistence mechanism	regkey	HKCU\Software\NetWire			
2020-06-08		Network activity	ip-dst	185.140.53.48			C2
2020-06-08		Network activity	domain	cloudservices-archive.best			dropsite

MISP has support by default for about two hundred attribute types and sixteen type categories. Every type can make sense in one or more categories. For example ambiguous attribute types such as "link" can be classified as "Payload Delivery" or simply "Internal reference". For more complex data structure MISP also provides

the use of Objects to organize data in events. Objects are specific data structures composed by a dictionary of key-attribute pair and are based on real cyber security use-cases and existing practices in information sharing such as object describing bank accounts, credit cards, credential information but also social media interactions and posts. As a mean of organization and grouping together similar events MISP Taxonomy defines a common classification system to tag and organise events. Taxonomy allows to express the same vocabulary among a distributed set of users and organisations and is composed by a namespace (Required), a predicate (Required) and an optional value, for example a finance related event will be classified with a tag "circl:topic=finance". Users can also create and improve the MISP taxonomy by adding additional custom classifications. MISP galaxies on the other hand are a method to express a large object called cluster that can be attached to MISP events or attributes. A cluster can be composed of one or more elements and elements in the galaxy are expressed as key-values. An example of MISP galaxy could be a simple dictionary with country information or a dictionary containing a description of the most popular Android malware. There are a lot of default vocabularies available in MISP and all of them can be freely overwritten, replaced or updated. This exhaustive list of types, categories and objects can be used to express with detail the vast majority of malware analysis scenarios with complex relationships and detailed tags.

MISP also come with a Python library to interact with the platform via API to fetch, add, update events or attributes, handle malware samples or search for attributes. A use case of a MISP is to have an instance to store malware information and to have multiple IDS and SIEM to log and detect intrusions. IDS and SIEM instances can import malware information and indicators of compromise from MISP periodically.

MISP Feeds

As an alternative to manual inserting MISP support feeds, an automatic way of importing events. A feed can be seen as a stream of event, usually managed by a trusted third party, that can be added to a MISP instance to allow periodic and automatic fetch of events that are added over time in the feed. They can also be used as a source of correlations for all events and attributes already present in the

instance without the need to import them directly into your system. The MISP feed system allows for fast correlation but also a for quick comparisons of the feeds against one another via a overlap matrix showing the percentages of indicator found in feed pairs. There are many default feeds that are already configured in any MISP instance and just need to be activated in the MISP feed web interface. A schedule can be set for refreshing feeds to have an automatic stream of events from a trusted source. The preferred feed format is the MISP standardized format as it can benefit from all the MISP functionalities, but also CSV and freetext is allowed, for example a simple text file containing a list of IPs can be imported as a feed and tagged as a blocklist. Also when adding a feed via the MISP web interface additional control is available such as feed tagging and filter rules to be able to select or skip specific events. When the files are local and present in the same machine the event files can be automatically deleted after ingestion. The main goal of the tool developed and described later is the creation of a feed in the MISP standardized format.

Chapter 3

YValidator - Design and Implementation

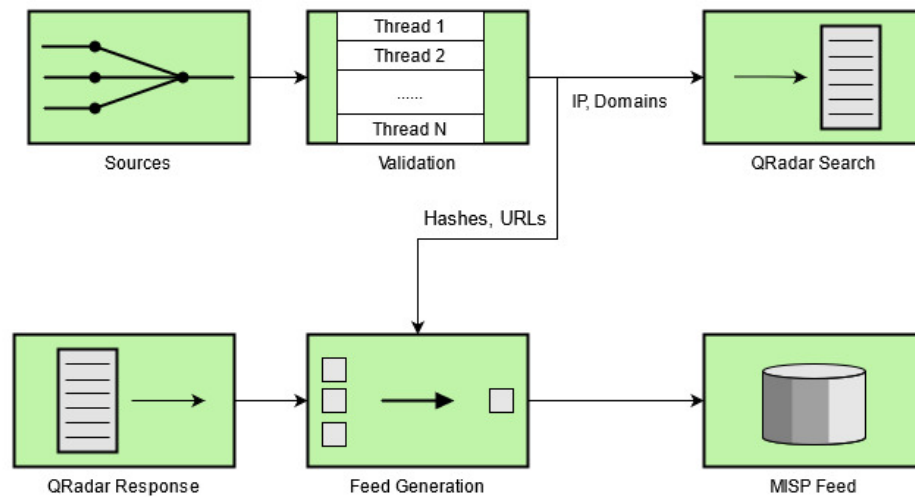
The idea behind the creation of this tool is to have an automatic system for fetching and validating indicators of compromise from publicly available sources and inserting them in a organized manner in a MISP instance. The point is to use sources that are not in any other MISP feed because of their being highly targeted nature, that are updated fairly often or simply because they are not published by a large and popular organizations but by smaller research groups or by individual malware analyst so that we can have automations for as much data that is being shared as possible.

The language used to develop the script is Python3 given that the company already had utility libraries for parsing indicators of compromise and also because there are Python API libraries available for all the services that we are going to need to interact with. The main goal of the program is to output a feed of events with information fetched from interesting sources in a MISP format for easy integration with the destination MISP instance. The secondary output of the tool is the validation report with information for all the validation steps for every indicator parsed during the execution. This report can be used to understand the reasons why an IOC was considered malicious or harmless (kept or discarded) and can be used to fine tune the weights and scoring system for better future validation decision.

The program is logically divided in four steps: fetching, validation, QRadar search and event generation. Each step has at least one thread assigned to it to make

sure that as soon as an event is finished with a step it is put in to a queue for the next task, if there is a thread available it can immediately continue the execution otherwise it will wait. More specifically when the program is run with multiple sources each source has its own thread, validation can be configured to run with one or more threads and for last two steps one thread each is enough, considering that the the QRadar validation just needs to send queries the QRadar server and the last thread waits for the results then generates and writes the MISP events to disk. Generally the script is network bound and indicator parsing will rarely be computationally expensive. During testing files with up to a couple thousand indicators are still processed in reasonable time and will briefly halt source fetching for event with the same type given that fetching and parsing share a single thread per source. There is the possibility to assign a thread specifically for the parsing task but the number of threads already used and the average size of the data are discouraging this choice. All the steps from fetching and parsing to validation and generation are logged in a rotating log file. The log can obviously be used for debugging purposes but also to gather additional information when a manual review is required, an expected process described later in section 3.3.

Figure 3.1: Diagram of all the phases



The tool interacts with many external APIs including Twitter, GitHub, URLScan.io, Virustotal, IBM-XForce and all the development and testing was done using the public and free version of these services with the relative limits applied. In a production environment these limits are too strict and services like Virustotal and

XForce prohibit the use of public APIs in commercial products or services and business workflows.

Every execution of the script requires as input a configuration file in the YAML format. This file contains logging information for debugging purposes and all the information regarding sources to fetch, validation steps to execute and some general information regarding the MISP feed, such as output folder, name and id of the creating organization, threat level and status of the analysis. These information will be added to all the event generated by the script.

The program is supposed to be run multiple times a day at a set interval with a subset of sources. There are five available command line arguments:

- -c, Configuration file: required argument with the file name of the configuration file.
- -f, Fetch options: list of online sources to fetch, the source name must be one in the list of the implemented ones.
- -i, Input file option: Alternatively to the fetch command, the script supports a text file as input with a list of indicators or a single indicator. This option is useful for running the program manually as a validator.
- -o, Output report name: For every execution a CSV report file is written with all the information gathered by the validators. A name and location can be specified for this file, otherwise the default one will used.
- -s, Skip options: For every execution step a skip flag can be set. Mainly used for debug purposes or when using only being interested in the validation results skipping the feed generation phase and the QRadar searches.

3.1 Sources

For every source all the detailed information regarding the specific fetch options are written in the configuration file and when running the script the fetch argument allows you to decide which subset of all the sources to look at. This is done to be able to check some sources more frequently than others while setting up the configuration for as many options as we want. Most sources create multiple events

in an asynchronous way, meaning that while the fetching thread is still loading and parsing data, event already parsed are being validated. All the classes used to fetch events have the same common interface consisting of a fetch method that will start the requests to load the data and a method for saving to file the date and time of the last successful fetching. Additionally all the different source types have a default set time frame or fixed maximum amount of events to load when sources are loaded from a cold start in order to avoid getting events too much far back in time. The event object is created at the fetching phase and it is used to group together indicators from the same specific source and is purposely similar to a MISP event with most of the required information needed to be added to a feed set to default values. Every event will contain a dictionary with all the indicators grouped by type and another dictionary for all the removed indicators to also keep track of all the removed values that failed validation. Additionally the raw text from the source that has been parsed is saved for later use.

All the additional information set in the configuration file such as tags, organization, threat level and others will be added later by the feed generation step, the last phase of the script. The parsing of text is done via regular expressions that match domains, URLs, IP addresses version 4, md5 and sha256 hashes. The parsing process also does some additional checks on domains and IPs such as top level domain verification and IP ranges checks.

Twitter

When using Twitter as a source there is the option to load tweets from a list of specified users or search for tweets with a particular hashtag. Getting a user timeline is fairly straightforward using the twitter-python library and by saving the tweet id of the most recent fetch we can avoid loading already seen data from previous executions. Unfortunately the Twitter search is less easy to work with as it doesn't support the last seen tweet id and it also limits the date range interval to single days, so for example it is not possible to search for tweets in a two hour range. To solve this problem for every search it is needed to save the timestamp and skip the tweets created before this time if we want to check a specific hashtag multiple times a day. Additionally for every tweet returned by the search it is useful to also look at the entire Twitter thread looking for the replies written

by same user to himself, because unlike the user timeline API call these are not included.

Before passing the full tweet text to the parser, uninteresting links, such as links to images and articles, are removed and white-listed links are opened and added to text to parse. It is common due to the character limit to post a link to Pastebin or GitHub containing all the indicators and the whitelist is added to avoid following URLs that may be completely unrelated and that could add a lot of noise and useless data to parse and validate. Twitter is the only source that waits for all tweets to be loaded because there can be overlap between the two types of searches as a Tweet from a followed user could be also found in a hashtag search.

GitHub

GitHub Sources consist of a list of repositories, and target filenames to fetch. For every repository, commit information is accessed using the GitHub API. With all the commits data for the desired date range all the files found as modified or added that are also present in the target list of files to fetch will be saved and only the most recent commit version of these files will be accessed. With the list of the most recent files a request is made for the raw text version.

URLScan.io

URLScan is a popular URL scanning service that also provides interrogation APIs to search for past and completed scans meaning it can be used as a source by fetching the most recent scan results. Given the the very high volume of result produced each hour, about six thousand to seven thousand on average, and of number of harmless URLs submitted to the service, all the results with a unknown or non malicious verdicts are filtered and ignored, only the results with a malicious verdict will proceed to the validation step.

Blogs

Generic web pages can be used as a source by adding the page URL to the list of manual sources, but when we want for example to load recent articles from a blog or website we need an ad hoc solution for every case. With a library like Beautiful Soup the job becomes definitely easier but given the different structure of different

websites this is by far the less generic type of information source to use, in fact during development only one specific website publishing IOCs was added to the tool.

MISP

One of the goals of this project was to write a program that could validate events and attributes already present in a MISP instance, so that when new events are added automatically for example using the sharing functionality from other external MISP instances, they can also be fetched by the tool for additional validation. Using a MISP instance as a source can cause problems especially when the destination MISP instance and the source are the same. Events created by the tool are not directly added with the MISP API but are instead written to files which are then added as MISP feed to the desired instance. This means that before deciding what attributes to keep or remove we need to make sure to only allow duplicate events and attributes for events added automatically, to avoid validating events added manually by a team internal to the company.

Manual Sources

Lastly there is the possibility of providing a list of URLs to fetch, the script will simply make an HTTP Get request for every URL and give the parser the entire text response. Useful when wanting to add manually found URLs like Pastebin and similar services. The difference between this type of source and file in a GitHub repository is that using GitHub APIs to ask for commit information, we can be more efficient in fetching and parsing only when there are changes to the files we are interested in while for this type of source it is expected to be more static and rarely change. Also using the `-input` argument a file name can be provided and, if it exists it's read as text and parsed, otherwise the string provided is considered as a single indicator and parsed accordingly.

3.2 Validation

The validation phase is an attempt to simulate and automate a manual validation process consisting of analyzing all the information available for a specific indicator

Threat Intelligence – Dridex Malware Latest IOCs

By Balaganesh - April 20, 2021

0

Indicators of Compromise



THREAT IDENTIFICATION: DRIDEX

SENDER EMAILS

quickbooks@notification [.]jintuit [.]com

SUBJECTS

Reminder: Invoice 714873

MALDOC FILE HASHES

714873 [.]xls
5c3a1b785f532a889980751123e3ffce

PAYLOAD DOWNLOAD URLS

https://vegasvulkangermany [.]veronafoodbd [.]com/nteqdu5 [.]rar
https://sydwaltcrmfrontend [.]khholdings [.]co [.]za/d5mvar80 [.]zip

PAYLOAD FILE HASHES

nteqdu5 [.]rar
340994098deb6bf6fa91f73350af7c15

Figure 3.2: Example of Sources: Tweet and Blog article

to make a classification decision while also providing a high level of configurability through categories weights and scoring parameters.

A MISP extension already exists, called "MISP-IOC-Validator" published on the MISP Project website [7], for validating indicators in a MISP instance. The checks that are done by the extension are mostly on format, such as usage of invalid characters for domains or checks on private IP addresses, with the option to set a list of known false positives to remove. The limitations of this extension are that only values already present in a MISP instance are used as a source and the short list of basic checks done during validation. Another tool extending MISP functionality, called BTG [8], will simply search information online when provided input with a single or list of indicators while letting the user make the decisions based on all the information provided. This tool was designed to speed up the validation process but it is not completely automated as it will need user interaction with the advantage of producing a more desirable classification result. The validation approach of this tool can be seen as the sum of the two described MISP extensions: it will search for information online and then perform a series of checks to automatically determine the validity of the indicators processed. This phase can be considered the core of the program, it is the most configurable phase and the one that outputs the most amount of information. Every indicator will have a score assigned at the end of this step and decision will be made depending on the score value and indicator type. The final score is the result of the sum of partial scores, each with its own weight. Validation is divided pre-validation

where local information is used to keep or remove indicators and external validation where the information is gathered from Virustotal and IBM XForce.

Scoring for indicator x is computed by summing all the partial scoring p_i as follows:

$$S(x) = \sum_i w_i p_i(x) \quad (3.1)$$

The weights w_i for each partial score are positive values and can be seen as the amount of trust given to a certain type of information and are fixed for every execution. Scoring on the other hand can be positive, negative or zero: a positive score suggest that the indicator is malicious, a negative score indicates an harmless indicator and a zero score means a decision can't be made for reasons like the absence of information or partial lack of. Together with the final score a simple indication of confidence is provided based on resulting score and the number of partial score collected to compute the final score: when only two or less non zero partial score are used to produce the final score or when the resulting score is very close to zero then the validation confidence is considered low. The confidence value is computed as additional information but it is also used by the last validation step, during the validation with QRadar.

3.2.1 Pre-Validation

The first steps in the validation are used to answer some basic questions about an indicator and to prevent validation in cases when is not needed, keeping in mind the API quotas of external validation services. The pre-validation phase is used to check:

- If the indicator is already present in the destination MISP instance.
- If the indicators has already been validated by the tool recently
- If the indicator is present on a local blacklist or whitelist.

Essentially the pre-validation uses all the local information available to make a decision regarding the discarding of a indicator before proceeding to querying external validation services, to avoid making superfluous requests to Virustotal and IBM XForce with daily or monthly quotas.

Blacklisting and Whitelisting

The very first step is to look in the local blacklist and whitelist. There are four files containing respectively whitelisted domains, whitelisted IP addresses, blacklisted domains and blacklisted IP addresses. If an indicator is found in a whitelist it is instantly removed from the event, if it's found in a blacklist the indicator is kept and will continue the validation but with an added negative score of significant weight. Support for regular expression is not present but will be implemented at a later time.

Indicator History

The second pre-validation step is to look at the recent history of seen indicators. This means that every unique indicator fetched will be saved in a file for at least seven days with information about the source of the data, the type of source, the final validation score, whether it was added or not to the feed and the fetching timestamp. All of this information can be used to make a decision of what to do with indicators that are found multiple times especially from different sources. For example when an indicator is found in the history, meaning it's not the first time it is seen, and it was not added because of the resulting negative score of the validation it is checked again. This is done because it is possible that a fresh indicator may not be instantly found on any external validation services and therefore may be considered harmless at first but if checked again a couple of days later return a different verdict.

These are all the current logical rules of this step:

- a) If an indicator is very recent (around 24 hours old) it is not validated again
- b) If it is saved as already added it is not validated again.
- c) If it is found in history and comes from the same exact source it is not validated again.
- d) When all of the above are not true it is validated again only if the score is lower than a threshold.

At the start of each execution when the indicator history is loaded all the entries with an insertion timestamp older than seven days are removed, to prevent the

history to grow too big, and all the indicator not found in the history will be treated as new.

MISP Search

The last and final step of pre-validation is to search all the remaining indicators in the attributes of the destination MISP instance. This search will not simply look at all the events but at events with at least one of a list of specific tags. There are essentially two types of events: event created and added by the company operating the MISP instance, and events added automatically through the sharing process or through other feeds. In the configuration file a list of tags to be searched can be specified and if the searched attribute is found in an event with one of those tags then it is considered already in the MISP instance otherwise it will be kept and will continue with the rest of the execution. Simply, the purpose of this search is to avoid checking attributes that were already manually reviewed and added by the operating company. The ordering of the checks has to goal to make as few requests as possible to the MISP instance even if these kinds of searches are not particularly expensive.

3.2.2 External Validation

External validation refers to the use of online services that share threat information along with additional data when provided an indicator. The external validation services used in this tool are Virustotal [9] and IBM XForce [1] and all the data provided will produce a series of partial score that summed together will result in a final score used make decisions on the validity and maliciousness of the input. The scoring categories used by the tool are composed of two main categories and eight secondary categories. The difference between the two is that the main categories produce a score from information that can be considered a summary of all the data provided by a service and, in the absence of errors, there is very low probability that this information is missing. As a consequence the main categories have a higher default weight than the others. The main categories are the Virustotal Statistics and the XForce Threat Score and are present for all indicator while the secondary categories vary depending on indicator type.

The rest of section 3.2.2 will describe all the categories information used to produce

a partial score and how to compute it. The formulas described were created with the idea that every information category can be the deciding factor on the classification decision and the formulas scoring try to reflect the amount of trust and importance given to each information.

Virustotal Analysis Statistics

Virustotal is a popular malware analysis service that also provides access to scan results through any programming language using the HTTP-based public API. The service supports URLs, domains, IPs and hashes. Every indicator scanned by Virustotal will be analyzed by a large number of security vendors and every response will include a detailed list of all the individual results as well as statistics regarding the last analysis run. The statistics are simply the cumulative number of security vendors that returned a certain verdict.

Listing 3.1: Example of Virustotal Analysis Stats

Last Analysis Stats:

- harmless: 0,
- malicious: 4,
- suspicious: 1,
- timeout: 0,
- type-unsupported: 8,
- undetected: 54

These statistics are use to compute the first score of the validation step and more specifically the number of harmless, malicious, suspicious and the total number of analysis are used.

$$S_b(stat) = \begin{cases} b + \frac{stat[m]}{N} + \frac{1}{2} * \frac{stat[s]}{N} & \text{if } stat[m] > 0 \text{ OR } stat[s] > 0 \\ -\frac{stat[h]}{N} * \frac{1}{2} & \text{Otherwise} \end{cases} \quad (3.2)$$

The expressions in 3.2 include b , the baseline parameter set by 0.5 by default and N the total number of analysis run by Virustotal. This scoring ranges from -0.5 when 100% of the result are harmless to +1.5 when 100% of the results are

malicious. The only parameter that can be configured is the baseline and can be described as the amount of positive bias included when at least one of the analysis concludes that the input is malicious or suspicious. Realistically and during testing the scoring ranged from -0.5 to 1.2 and scores greater than 1.2 are very rare. As an improvement is reasonable to exclude the results returning an error removing "type unsupported" and "timeouts" from the total. On specific and often popular items (example: "cmd.exe" hash), there is the possibility of the inclusion together with all the other stats an additional object called "trusted result", a manually produced report including a final verdict (Malware of Goodware). When this specific field is encountered all the other analysis field are ignored and the successive checks and scoring produced by Virustotal is skipped and a significant high or low score is added depending on the verdict.

Popularity Ranks

Popularity ranks are another tool that can be used to gather some information towards the harmlessness of a domain, it is not reasonable to use them as whitelist but ranking can certainly be used to tell the difference between a potential false positive and a true negative. Fortunately when querying Virustotal with a domain popularity ranks are also returned. There are many companies that create and manage these ranking like Cisco Umbrella, Alexa, Majestic, Statvoo, Quantcast, etc and Virustotal will return a list of all the ranking where the input domain is present and the respective positions. The ranking score is computed by first getting the average rank position of all the ranking and then is score is assigned based on tiers. If the domains is not in any top 1 million rank is it assigned a small positive score.

The tier proposed are:

1. rank greater than 100'000
2. rank between than 100'000 and 10'000
3. rank lower than 10'000
4. Not in any ranking

With the first three cases having a negative score inversely proportional to rank position and the last case having a fixed small positive score.

Country

Country is a field returned on indicators of type IP in the ISO 3166 format, for this information a simple blacklist and whitelist is used with a default small weight. This validation category can be configured to give negative scores to a set of countries and give a positive score to another set of countries.

Autonomous System Numbers

It is possible to set a whitelist of Autonomous System Numbers (ASN) to be considered harmless to avoid adding in the feed very frequently used IP addresses by giving a significant negative score to indicators with specific ASN.

Referrer Files and Communicating Files

Virustotal automatically creates relationships with the data that is submitted for analysis, a simple relationship can be a domain with its resolving address or sub-domains. The two most interesting relationship available for domains and IPs consist of Referrer Files and Communicating Files. The meaning of the term referrer files is simply a file that has the related domain or IP found in its strings, on the other hand communicating files are files presenting any sort of traffic to the related domain or IP. This kind of information can sometime be very useful, especially with Command and Control servers because if the related malware has been analyzed by Virustotal the communicating file list for a C2 will have one or few entries with a malicious analysis result and it will be a strong sign against the harmlessness of the indicator. There are some particular cases to be considered, for example popular domains will have a very long list of both communicating and referrer files, some of those files will be malicious, considering that files in the Virustotal relationships are file uploaded and scanned by users so it is normal for a unusual number of those to be malicious. In conclusion we cannot say that the indicator is malicious just because the files in those relationships are. Since the files relationship endpoints always return a list of files and require the maximum number of files returned as a query parameter we can use these value to balance the score computed using this information. First for every file a score is computed using the formula 3.2, then if all the files in the relation are not malicious then a small negative score is returned. If at least one of the file is malicious then an

additional parameter called volume is computed. The purpose of this parameter is to reduce the final scoring when the file list size is close to the maximum number of file requested and to increase the final scoring when the file list is small and mostly containing malware. By doing so all the popular domains will have a much lower numeric value proportional to the file present in these relationships. The final score will simply be the average file score times the volume parameter.

$$avg = \frac{\sum_i stats(f_i)}{N} \quad (3.3)$$

$$v = \begin{cases} 1 - \frac{N}{MAX} * \frac{4}{5} & \text{if } N \geq MAX/2 \\ 1 + \frac{malicious}{N} * \frac{1}{2} & \text{if } N < MAX/2 \end{cases} \quad (3.4)$$

$$S = \begin{cases} -0.25 & \text{if } avg = 0 \\ avg * v & \text{Otherwise} \end{cases} \quad (3.5)$$

The volume parameter will range from 0.2 when the maximum number of file is returned, suggesting a popular indicator, to 1.5 when the amount of file returned is low and are mostly considered malicious. The scoring is used for both referrer files and communicating files and only these two endpoints are available in the public API. There exists a lot more relation information but is locked behind the enterprise version of the service and because development and testing was done using the former only these two types of relationships were included.

HTTPS Certificate Issuer

Virustotal returns a complete description of the HTTPS certificate including issuer, signature and subject. The only field that is considered is the Certificate Issuer and it's checked against a list of organizations that issue free HTTPS certificates like Let's Encrypt, ZeroSSL and others providing a small positive score.

Domain Resolution

An additional information that is computed and added to the final IOC report is if a certain domain will return a valid DNS record when trying to resolve it. This

success or failure of this step will not influence in any way the numeric final score as it has by default a weight of zero, but it is useful data to have when manually reviewing an execution result. The DNS hostnames to use for the resolution can be edited in the configuration file.

Domain Creation

For domains only, a field containing the domain registration date is checked and domains created in recent date will have a small positive score. The date range to be considered recent is editable in the tool configuration file and is by default thirty days. This score can be useful to spot, for example, phishing sites as they will mostly be newly registered domains.

IBM XForce

IBM XForce Exchange is a service by IBM providing a wide array of threat information from collection of reports published by the IBM XForce research team to anonymization services, botnets information, phishing sites, vulnerability information, etc. In the use case of validation we are only interested in information regarding IP addresses, domains, URLs, and file hashes. All this information is publicly available and free of charge through a REST API with a monthly quota of 5000 requests.

When queried with an indicator the API response will include a report containing, among other information, a score ranging from one to ten described as Threat Score. There is not much available documentation detailing the meaning of this number or how it is computed so an initial period of testing is needed to gain familiarity and to understand what scoring is assigned in different scenarios.

XForce IP, Domain and URLs Scoring

Reports for IP, domains and URL all have the same response structure and similar information gathered. For IP addressed IP Reputation is defined as the entire history of instances when the IP scoring was changed and updated. XForce lets you decide if you want the entire reputation history without their final threat scoring or just the recent reputation history with their current threat score and reasoning behind it. Specifically reasons are briefly described in the reason description field

and the most common are the following: Content analysis, Early Warning and Statistical Analysis of previous history. Other information included in the response is country, categories of the IP and a list of tags applied.

Domains and URLs are treated in the same way by single endpoint called URL report. Similar to the IP reports it contains a reputation history with a list of all the previous scoring and related reasons. The difference with IP addresses is the additional information regarding the possible company associated with the domain and more detailed descriptions on eventual categories and tags associated with the service provided. The partial scoring produced by the XForce reports for IPs, domains and URLs is only dependent on the Threat Score, no difference in made regarding the reasoning provided or the reputation history as it seemed reasonable to put trust in the final and most recent score computed by the XForce system.

$$S_t(ts) = \begin{cases} \frac{ts}{10} * 1.5 & \text{if } ts \geq t \\ -\frac{t - ts}{10} & \text{if } ts < t \end{cases} \quad (3.6)$$

The scoring formula is modifiable through a single parameter t called threshold. The threshold will determine the lowest possible score for the indicator to be considered malicious and all the Threat Scores below that number will have a negative score that is linearly related with its value. The default threshold value is equal to 3 to have a similar range to the scoring computed using the Virustotal Stats in (3.2) given that these are the two main scoring with equal weight, it makes sense to have the same or similar scoring value range as well.

XForce File Scoring

Files are treated in different way that the rest of the indicators as the response is much simpler. It always contains a risk field with values equals to "low", "medium" or "high" and it can contain some malware origins such as a list of download servers, email sender and CnC Servers. In most cases the only usable information is risk as all the other fields are very rarely present. File scoring is somewhat hard to define as it only depends from three possible values. The proposed scoring is, like the previous, biased towards higher numerical values, just the fact that there is

an entry in the XForce Database is considered a small sign of maliciousness. This is not a light assumption to make because there are harmless file hashes in the XForce database all assigned with a "low" risk score. These edge cases combined with the very basic and limited risk classification can cause some incorrect scoring. For example it occurred many times during testing that hashes of a popular false positive files was reported by XForce as malware with "low" risk.

In conclusion XForce is very useful service for confirming indicators as malicious while not being very useful for proving the contrary. It's less intuitive and takes some time to understand the meaning of the scoring and is overall less reliable, for example it is not uncommon to receive a valid response to an indicator search with all the response fields empty or "Null" meaning that the queried data is in the XForce database but has no scoring assigned yet. It is also very important to have as a parallel source of information to Virustotal, even if in relation to the latter does not have comparable indicator database size as it will respond about 20% of the time with a error indicating nothing was found.

Validation Results

As a result of the validation phase each indicator of every event will have a list of partial scores assigned to it. For every partial score the relative fixed weight will be the one present in the configuration file before execution and a short text comment will briefly describe the information and the reason behind the produced score. All the indicators removed during the pre-validation phase will not have any validation data and will not be present in the indicators report, as they are considered already validated or there is a certain confidence that the result will not change if the validation is run again. At this point for indicators of type hash or URL the validation is over and a decision is made based on the final score: any negative value is considered harmless and discarded, positive or zero values are instead kept as attributes for the last step of event generation. For indicators of type domain or IP there is an additional check and event having at least one value of these types are passed to the last validation step of the program, the QRadar Validation. More specifically given that IPs and domains are the most important in terms of impact, all indicators of these types that have a validation result with a low confidence flag, meaning that the final score was the result of the sum of

two or less categories of information or the verdict is negative but still significantly close to zero, will also proceed to the QRadar validation to hopefully get a more clear result.

3.3 QRadar Validation

IBM QRadar is a security information and event management or SIEM product designed for enterprises. The software collects events and network flow data from hosts operating systems, endpoints and applications. QRadar correlates all this different information and aggregates related events into single alerts to accelerates incident analysis. It is also used to perform analysis of the log data and the network flows in real-time so that malicious activities can be identified and stopped as soon as possible. Queries can be sent to search for data in the logs with a QL called Ariel [2]. In a production environment YValidator is expected to query around twenty or thirty QRadar instances but during testing only three to five different instances where used in order to avoid long responses waits. The purpose of searching in the QRadar logs is multiple: it can easily detect false positives that the validation phase wrongly classified, it can add evidence to the maliciousness of the indicator and can raise attention at potentially infected networks or machines.

There are two threads that handle QRadar searches and event generation. This is done for multiple reasons: the first one is to be able to write and save to disk an event as soon as it is done and the second one is that Ariel queries are by design asynchronous. Asynchronous means that once a query has been sent the response will only include the result of the parsing and successful start of the search, the status of the search and the results need to checked and retrieved later with another request. The tool will query all the QRadar servers with the same queries and wait for results. When all responses have arrived for all indicators of an event then the thread waiting for the replies will also execute the code for event creation. The QRadar search API reference can be found in [3]

Starting searches

To start a search the only parameter needed is an AQL query to be sent in the query string of a POST request to the Ariel searches endpoint. A query is formed

with a group of indicators and the same request will be sent to all of the QRadar servers. Given that events with indicators are popped from a queue, there is no information on the amount of events to validate or even the remaining amount and in order to make sure that most of the searches are balanced in terms of size and consequent waiting time, two search parameters are enforced: a minimum search size and a maximum search size. Considering an example event with very few indicator of type IP (lower than the minimum) and a lot of indicators of type domain (greater than maximum), the domains will generate multiple searches per server in order to split the indicators in smaller queries and the few indicator of type IP will be put in a waiting list to be integrated with other few indicators from other events, as soon as the waiting queue will have a cumulative number of indicators greater than the minimum allowed the search will be initiated.

The query format is the following:

Listing 3.2: Example of IP Query Search in AQL

```
select destinationIP , UNIQUECOUNT(sourceip) , count(*)  
  
from events
```

```
where destinationIP = ... GROUP BY destinationIP last 3 days
```

The goal of the search is to determine how many hosts of a certain monitored system are interacting with the searched indicator and in what volume. The more significant value is the unique count of the hosts found having any kind of traffic with it and it's the value that decisions are based on regarding the insertion in the feed of the related indicator, in part because the number of unique hosts is also equal to the number of alerts generated if the attribute is present in the MISP instance. The cumulative number of log entries containing the searched indicator is not used for any decision because it can be in most cases not relevant or not helpful and overall less significant than the unique count of hosts involved, it is saved anyway for a potential manual review of the search results.

Another possibility for searching a large amount of indicator is using reference sets. QRadar allows the managing of lists of values to be stored in the database and then used for searching. It is possible to create and populate a custom reference sets with all the indicator of all the events and then request one single search with

the created reference set as a target. This approach has the drawback of adding logical overhead to the validation and the need to create and delete the sets for every execution but simplifies the query creation and eliminates the need to split searches in multiple requests. In conclusion the use of reference sets was considered but ultimately not implemented to avoid adding additional logic to the code and to not having to manually deal with non deleted reference sets in the database in the case of any kind of software failure. Additionally one set for each server needs to be created, populated and deleted for every single QRadar instance adding a considerable amount of HTTP requests and additional points of failure.

Waiting Results

After every search start whether is was successful or it encountered errors the search is added to a list of pending searched because is going to be accessed by the second thread assigned with the task of checking for search statuses and for results retrieval. Specifically the thread will, after waiting for a fixed time period, look at the status of every pending search in the list and depending on the value will remove them from the pending list. If the search is completed but the results have not been fetched yet, the result request and parsing will be queued, if the results have already been fetched without errors or an error occurred in any stage the search is removed from the pending list, in any other case the search status is requested again to the server to update the status and current progress percentage. When a search is removed from the pending list in order for the events related to the search to be generated all the searches on all the indicator types on all instances need to completed with the results fetched, so the tool needs to keep track of all the searches started for every event.

There are many configurable parameters for QRadar searches:

- Minimum and maximum amount of indicators per query: event with a lower number than minimum will be put in a waiting list and grouped together, event with a greater number than maximum will be split in multiple searches
- Minimum waiting time for responses and of rate of status polling: a lower rate will result in a faster program execution with the downside of making more server requests

- Number of maximum concurrent ongoing queries per server
- Search timeout in minutes

As a brief note on performance, every QRadar instance can approximately handle 4000 indicator searches per hour on indexed fields. During testing the domain property on logs was not indexed and queries would take up to thirty minutes compared to the two minutes needed for IP addresses queries, an indexed property.

QRadar Verdict

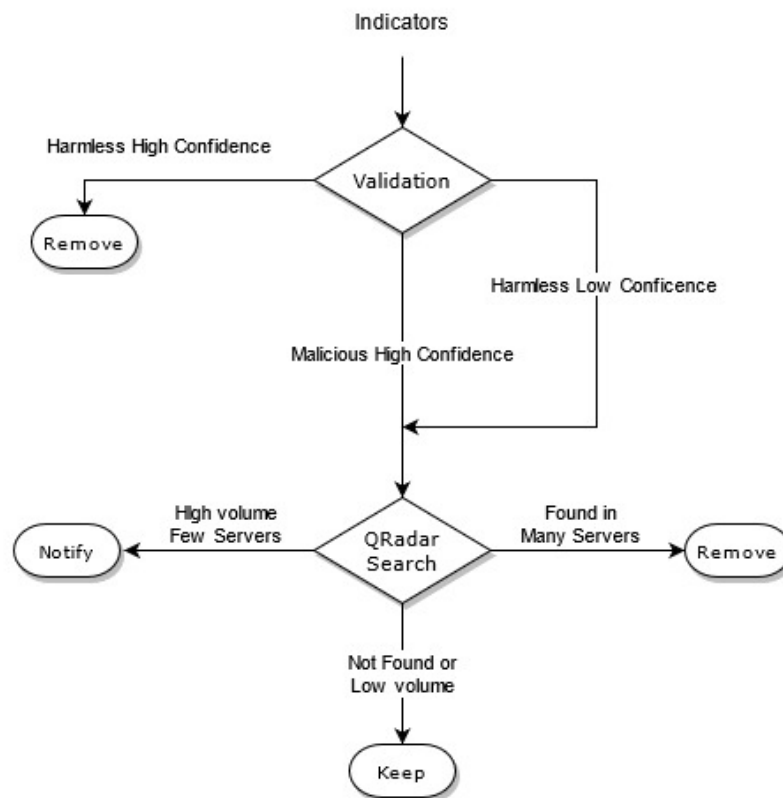
Decisions for every indicator made depending on the numbers found in the QRadar logs are the following:

- a) Remove it when the indicator is found in a number of different QRadar servers greater than a threshold
- b) Remove and add the indicator to notification list when the indicator is found in a lower number of instances (lower than case a) but in a significant number of hosts.
- c) Keep it when an indicator is found in a small number of servers and in low quantity of hosts

All the limits and parameters are configurable, default parameters are: at least found in 20% of instances to remove, 50% error limit to make a "Keep" decision and 100 unique sources to save the indicator for notification.

The choice made for the implementation of the notifications of indicators where a decision can't be made on whether to add or remove it to the feed was to send uncertain indicators via email. The content of the email will include for every indicator a report on all the searches results for every QRadar instance with the number of errors or timeouts occurred. At this point manual intervention is needed to verify the situation, if it is the case of a false positive no further action is needed as the indicator is not in the feed, on the contrary if it is a case of malware infection the indicator in question will need to be added manually later. In this case the validation report and the execution log will contain all the information used by the tool from all the partial scores and data used to compute them and all the complete QRadar search responses.

Figure 3.3: Validation Decisions Flowchart



3.4 MISP Feed Generation

There are essentially two ways of creating and adding events to a MISP instance: the first one involves being authenticated to the MISP server and making requests using the MISP APIs to add events and attributes directly, the second one involves creating an event feed in the MISP format. The main advantage and difference between the two is the independence from the destination server, the tool will still produce events even if the MISP instance is down or for some reason unreachable while this is not true if events are added using the MISP API. Creating a feed just means writing events on files and when the feed is made reachable or publicly available then not just one but multiple instances could use it as it just needs to be added in the desired MISP web interface. A scenario of implementation is to have two different machines: one running MISP and the other being a web server with the tool running at scheduled times. A completely independent case as each machine can run without problems when the other fails. A second scenario could be having a single machine with both MISP and the tool: a web server is not

needed if we are only interested of inserting the feed in a single instance and this setup has the advantage of being a local feed, meaning that MISP will automatically delete events after insertion while the first scenario might need an additional simple cleanup script to prevent the feed to grow to an unnecessarily large number of event files.

Feed generation was done with the help of the PyMISP library[6]. The GitHub repository provides examples of a feed generation script creating daily events [5]. That example was modified to be able to create arbitrary events while keeping the functionality needed to create a feed in the MISP format. There three types of files are created in a feed: the manifest file, the attribute hashes file and event files. The manifest is a JSON file containing a dictionary with all the events UUID as keys and events basic information without attributes as data. The attributes hashes file is as CSV file containing a row for every attribute of every event with the pair of MD5 hash of the attribute value and UUID of the corresponding event, this file is used by the MISP instance to perform a quick attribute search in the entire feed. The last type files are event files, every event has its own JSON file named "*UUID*.JSON" that contains the MISP event dictionary. The event dictionary always contains the event info or description field, the event creation date, threat level, analysis phase, information about the organization creating the event and whether the event is in a published state. All of this information apart from the date and event description can be set at a general level in the configuration file. Another configurable parameters are tags: tags are used in MISP to categorize and aggregate similar events, in this tool tags are handled in a basic way. There is the option to set a list of general tags in order to classify events added and created by the tool and the option to set specific tags for every type of source.

The feed generation phase also handles the difference between sources of type MISP and other sources. When fetching events from a MISP instance the event data and attributes are already in the output format and the only action to take is the optional removal of attributes failing the validation phase, also all the other information regarding the event such as tags and organization must not be overwritten, the output event will be at most with a lower number of attributes and will have additional tags indicating the event was validated by the tool. MISP attribute types considered for validation are: hostname, domain, URL, link, md5,

sha, ip-dst and ip-src. An improvement over the current implementation would be to also validate more complex data such as MISP objects considering the feed generator class also support objects generation. Another simplification implemented that could be improved is the different handling of source IPs and destination IPs, as of now all IP addresses are added in MISP as "ip-dst" type attribute so it can only be used to block outgoing connections and is not in any way helping block incoming undesired traffic. For example it can't be used to import IPs related to email spam or the list of hosts belonging to a botnet as the tool is not context aware with regard to IP addresses.

Lastly all non MISP sources will have an additional attributes of type "comment" with the URL to the specific source used to create the event to be able to look at the original raw data to get all of the context that cannot be explained with just attributes when viewing the event in the MISP web interface.

Table 3.1: Mapping YValidator type to MISP type

Indicator Type	MISP Type
IPV4	ip-dst
Domain	domain
URL	url
Hash md5	md5
Hash sha256	sha256

Table 3.2: Mapping MISP type to YValidator type

MISP Type	Indicator Type
ip-dst	IPV4
ip-src	IPV4
link	URL
url	URL
domain	Domain
hostname	Domain
md5	Hash md5
sha256	Hash sha256

Chapter 4

Validation Statistic and Testing

In this chapter we will briefly describe the testing done to compute some measures on the accuracy of the validation scoring system. As a first step is it useful to understand the maximum amount of information used to compute the final score for every indicator as not all of types handled by the tool have access to the same amount of data, as it easier to find more information related to a domains or IP address than it is for file hashes and URLs. Luckily the latter types are also the one that are generally less impactful when a erroneous verdict is made. This information is available in table ??

The accuracy score were computed by executing the validation phase on about 1500 known indicators across all types and with around 25 % of indicators being harmless. The accuracy values are computed using a list known harmless indicator and a list of known malicious indicators and by inspecting the result and confidence columns in the validation report.

These test showed that even with a correct final resulting score the validation report showed that every single validation category can output a convincingly opposite partial score compared to the final sum and for this reason all the scoring with same or similar weight have also a similar range of output values to enable every single potentially wrong partial score to be outweighed by a single or multiple other scores of opposite sign. Testing was done on limited number of indicators also due the daily quotas of external validation services, as it was possible using the free versions of these services, to get information for about 120 indicators per day. The main problem and limit with the validation is the potential low amount or complete lack of information for a specific indicator. This problem can be

attributed to many reasons, ranging from all the generic or network error that will fail to gather validation data and will contribute with a zero score to more critical situations like the higher probability of not finding data on indicators published very recently that will also contribute with a zero score. For recent indicators a solution might be to re validate all the indicator with a negative score before the deletion from the history, by default after seven days, a step described in the pre-validation section 3.2.1. The lack of data can cause problems especially for non malicious false positive IPs and domains that can be added to the feed and could not cause problems in the immediate future but reemerge later after weeks or months. For example a rarely visited domain could be classified as malicious and also not found in any QRadar search resulting in the addition the feed and consequently in the MISP instance. This a probable and expected case as false positive are inevitable, and if any legitimate future traffic is generated, it will cause inconvenience and will eventually need to be manually reviewed and removed. The reasoning behind the decision to add indicators of type domain and IP even when the score is slightly lower or when the amount of information is low is that the cost of not adding a malicious indicator cannot really be evaluated and considering the necessary outcome of QRadar searches giving zero or few hits and the generally low impact of adding a rarely used false positive to MISP, this decision will outweigh in most cases the potential damage done by all the low confidence or low information true positives. Another limit of validation is that while it is true that weight are easily changed, editing weight cannot solve all the erroneous validation scores and verdicts. Modifying the scoring system ideally requires knowledge of the entire scoring variables and categories and requires altering the code of the script, which is not ideal, even considering all the scoring logic is not very complex and centralized in a small Python file. In the accuracy tables the notation used for the possible validation result is "Keep" or "Remove" and not "Malware" or "Harmless". The distinction is made to avoid adding more complexity to the tables by having to add all the combinations of positive and negative scoring with high or low confidence given that it is possible that an indicator with a harmless validation result is still kept as valid and would also only make sense for IPs and domains as all the other indicators simply rely on the numeric final score for decisions.

Table 4.1: Accuracy for Domains

		Expected Result	
N = 647		Malware	Harmless
Decision	Keep	429	12
	Remove	37	169
		7.9% FN Rate	6.6% FP Rate 92.4% Accuracy

Table 4.2: Accuracy for IP addresses

		Expected Result	
N = 613		Malware	Harmless
Decision	Keep	467	17
	Remove	25	104
		5.1% FN Rate	14.0% FP Rate 93.1% Accuracy

Table 4.3: Accuracy for URLs

		Expected Result	
N = 116		Malware	Harmless
Decision	Keep	62	20
	Remove	4	30
		6.0% FN Rate	20.0% FP Rate 79.3% Accuracy

-

Table 4.4: Accuracy for Hashes

		Expected Result	
N = 118		Malware	Harmless
Decision	Keep	93	15
	Remove	0	10
		0 % FN Rate	6.6% FP Rate 87.2% Accuracy

Table 4.5: Combined Accuracy

		Expected Result	
N = 1494		Malware	Harmless
Decision	Keep	1051	64
	Remove	66	313
		5.9% FN Rate	16.9% FP Rate 91.2% Accuracy

Tests were particularly focused on domains and IPs given the amount of possible information used is greater than the rest as URLs and Hashes only rely on at most two partial scores to get the final score. The most important statistic to consider is the false negative rate. The scoring system is inherently biased towards classifying indicators as malware and therefore deciding to keep them. When a wrong negative result is produced the indicator is removed without additional checks and these cases are the ones that have the most potential negative impact in real threat monitoring scenarios. A solution to this problem, described previously, could be

to re-validate not added indicator before removing them from the history.

Accuracy for URL and Hash is acceptable, not a lot of indicators were tested due to the limited categories involved in the final scoring and no major inconsistency or mistakes were found. For IPs and domains the most problematic cases are indicators related to C2s as an high percentage of them will have a negative Virustotal and XForce scoring that can be outweighed by the very high positive scoring in the relating file category when present, showing a correct output of the scoring system even when multiple partial scoring are in disagreement. The overwhelming majority of false negatives for domains and IPs is the case when there is no information on related files and the numeric scoring was negative enough to be considered high confidence.

As a reminder these results are not the final decision for inserting the data in the MISP feed because IPs and domains also need to pass the checks on the QRadar servers a step that has the task of removing a portion of false positives produced. For this reason the false positives rates being high is not particularly a problem but consist of cases when a search in the SIEM is preferred. The test results should be used to adjust the scoring and weights of the scoring system but the quotas of the free version of external validation services prevented the changes from being tested and validated in a reasonable time.

Table 4.6: Score Ranges with default weights and parameters

	Virustotal Stats	XForce Scoring	Popularity Ranks	Domain Creation Date	Country	ASN	Certificate Issuer	Communicating Files	Referrer Files	Domain Resolv- able
Weight	1	1	0.5	0.1	0.1	1	0.1	0.5	0.5	0
IP Score Range	-0.5,+1.5	-0.5,+1.5	NA	NA	-1,+0	-1,+0	-0,+1	-0.25,+2.25	-0.25,+2.25	NA
Domain Score Range	-0.5,+1.5	-0.5,+1.5	-1.75,+0.25	-0,+1	NA	NA	-0,+1	-0.25,+2.25	-0.25,+2.25	0
URL Score Range	-0.5,+1.5	-0.5,+1.5	NA	NA	NA	NA	NA	NA	NA	NA
Hash Score Range	-0.5,+1.5	-0.5,+1.5	NA	NA	NA	NA	NA	NA	NA	NA

Chapter 5

Conclusions and Future implementations

After a couple of weeks in a testing environment the program did not present any major problems apart from the expected minor bugs and unaccounted for cases. Considering the script configurability, there are in total 32 parameters ranging from scoring weights, formulas parameters and general validation configuration that allow a wide range of modifications without accessing the code script. Additionally all the scoring logic and formulas are contained in a small Python file of less than 100 lines to facilitate even the more important and sizable changes but with a required knowledge of the validation phase much more complete than just the configuration file.

As far as sources implemented we believe we covered a wide amount of possible malware information data available online considering all the default feeds already present in every MISP instance. The advantages are that the tool can be configured by hand picking the sources with the more relevant and coherent information with the system that a company has to monitor and to ingest it in a daily or hourly updated MISP feed to also help prevent against smaller and more targeted malware campaigns given that all the more popular, critical and high threats events are already covered by other feeds. Additionally simply implementing a new type of source can be done in a reasonably short time. Performance wise the various tests show an average execution time of 50 to 70 minutes every 1000 indicators with the longest phase being the external validation considering it has to make up to

five HTTP requests per indicator. An acceptable result considering that the tool is intended to be run multiple times a day in order to fetch all the sources that are updated very frequently as soon as possible. Regarding the QRadar performance it is marginally dependent on the tool ability to merge as much indicators as possible in single query but mostly on the server configuration and hardware, testing showed a significant performance penalty when querying, for example, unindexed logs properties. On the contrary query that use indexes take at most a couple of minutes to complete a waiting time and always lower than execution time of pre-validation and validation combined.

The accuracy, false positive and false negative rates are reasonable and the overall system works fairly well. There are some known issues and types of indicator that have a higher probability of returning an erroneous results but in general these problem involve a small subset of all the data fetched. Testing gave a lot of insight on the causes of results errors but with a limited quota for verification it is not feasible in short time to apply and verify changes as the data took a couple of weeks to produce.

In conclusions the predisposed goals are accomplished but the long term use will possibly highlight the major concerns and limitation that tool has and a more flexible and easier to modify scoring system solution can easily be an area of great improvement. Another section that was largely simplified is the MISP event generation. All the event parameter apart from the description and attribute and tags are statically assigned and a more intelligent event tagging system, event groping or dynamic event parameter assignment can greatly improve the usefulness and usability of the events generated when they are imported and visualized in a MISP instance while also even possibly reducing the total number of events.

Bibliography

- [1] IBM. XForce API Reference. <https://api.xforce.ibmcloud.com/doc/>, .
- [2] IBM. QRadar AQL Reference. https://www.ibm.com/docs/en/SS42VS_7.3.2/com.ibm.qradar.doc/b_qradar_aql.pdf, .
- [3] IBM. QRadar Search API. <https://www.ibm.com/docs/en/qsip/7.3.2?topic=endpoints-post-arielsearches>, .
- [4] MISP Project. MISP Data Models. <https://www.misp-project.org/datamodels/>, .
- [5] MISP Project. Generic MISP Feed Generator. <https://github.com/MISP/PyMISP/tree/main/examples/feed-generator-from-redis>, .
- [6] MISP Project. PyMISP, official Python library for MISP Rest API. <https://github.com/MISP/PyMISP>, .
- [7] MISP Project. MISP-IOC-Validator Extension. <https://github.com/tom8941/MISP-IOC-Validator/>, .
- [8] Conix Security. BTG IOC Search Tool. <https://github.com/conix-security/BTG>.
- [9] Virustotal. Virustotal API Reference. <https://developers.virustotal.com/v3.0/reference/>.