Fachhochschul-Masterstudiengang
Data Science und Engineering
4232 Hagenberg, Austria

Measuring Data Quality in STIX-based SOAR Platforms

MASTERARBEIT

zur Erlangung des akademischen Grades
Master of Science in Engineering

Eingereicht von

Konstantin Papesh, BSc

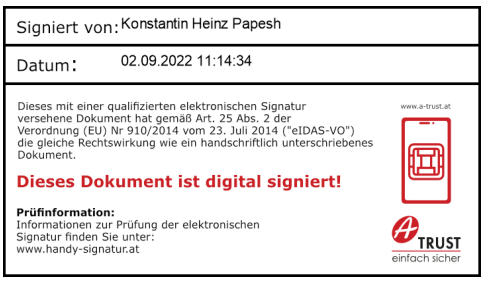
Betreuung: FH-Assistenzprof. Dr.techn. Emmanuel Helm, MSc

Hagenberg, September 2022

Declaration

I hereby declare that the thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere. This printed thesis is identical with the electronic version submitted.

Hagenberg, 2nd September, 2022



Contents

Preface **vii**

Abstract **ix**

Kurzfassung **xi**

1 Introduction **1**
1.1 Landscape 1
1.2 Goal 2
1.3 Involved Stakeholder 2
1.4 Structure 3

2 Background **5**
2.1 Cyber Threat Information and Intelligence 5
2.2 Structured Threat Information Expression 5
2.3 Security Operations Center 7
2.4 Security Orchestration, Automation, and Response 7
2.5 Data Orchestrator 8
2.6 Playbook 8
2.7 Related Work 11

3 Possible Approaches **13**
3.1 Requirements 13
3.2 Possible STIX Metrics 14
3.3 Possible API Metrics 24
3.4 Discussion 25
3.5 Final Metrics 30

4 Methodology **31**
4.1 Overview 31
4.2 Architecture 31
4.3 Playbook Execution in Detail 32
4.4 Hook 33
4.5 Scraper 34

4.6 Analyzer 37
4.7 API 46
4.8 Frontend 46

5 Results **49**
5.1 Setup 49
5.2 Report Example 50
5.3 E-Mail File 51
5.4 Malicious Hash 55

6 Discussion & Outlook **59**
6.1 Discussion 59
6.2 Outlook 60

A Technical Details **63**

List of Figures **69**

List of Tables **71**

List of Algorithms **73**

Glossary **75**

Acronyms **77**

Bibliography **79**

Preface

> Do not act as if you were going to live ten thousand years. Death hangs over you. While you live, while it is in your power, be good.
>
> Marcus Aurelius

Before starting this work, I was not particularly familiar with security systems or Cyber Threat Intelligence in general. Nevertheless, it was interesting to delve into the workings of today's cyber threat defence and to contribute just a little bit to its progress. This thesis was fun to implement and write. I hope it is also bearable to read.

Thanks to Emmanuel Helm for advising and guiding me throughout this thesis and gratitude to Christoph Praschl and Sebastian Pritz for proofreading, and the AIST staff in general; I enjoyed writing this thesis in your company. I would also like to thank Nextpart which allowed their platform Guardia to be used within this thesis.

Thanks to everyone for sticking to me. Thanks to all my friends for enduring me in general, but especially while writing my master thesis. I would also like to thank Dallmayr, J.Hornig and Bio-Kaffeerösterei Kurt Traxl for supplying dearly needed energy for my studies. Furthermore, I would like to thank the HSD-Kuchl, Starkenberger, whoever installed the air-conditioning in our office, Heaven Shall Burn, Kasger, the publicity department of Turkish Airways, the Österreichische Bundesbahnen and its main competitor, Westbahn, MPreis and last but not least, IKEA.

And finally, a big thank you to Viktoria and the whole team at the university pub. I will miss you.

Abstract

Security analysts use Security Orchestration, Automation, and Response (SOAR) platforms to dissect and analyse possible malicious data. These platforms rely on external services to enrich the given data. However, the data quality of new services is often unknown. To mitigate this issue, metrics can be established to assess different parameters in connection with data quality.

This thesis analyses Cyber Threat Intelligence (CTI) metrics currently proposed in the literature on their viability within Structured Threat Information Expression (STIX)-based SOAR platforms and implements the first version of such a measuring framework. Multiple metrics are compared against a list of requirements set by the nature of SOAR platforms. After viable metrics are identified, they are implemented within a framework which hooks into an existing SOAR platform. Finally, the framework is tested, and the calculated metrics are discussed.

The conclusion is that there are metrics available that can be altered to work with SOAR platforms. However, some metrics rely on parameters not readily accessible from SOAR platforms. That means the design of these platforms also needs to consider the requirements for data quality frameworks.

Kurzfassung

Sicherheitsanalysten verwenden Security Orchestration, Automation, and Response (SOAR) Plattformen, um Schadsoftware zu sezieren und analysieren. Diese Plattformen benötigen externe Dienste um die vorhandenen Daten anzureichern. Oft ist die Qualität neuer Dienste jedoch unklar. Um dieses Problem zu beheben können Metrik aufgestellt werden, welche verschiedene Parameter in Bezug auf Datenqualität erheben.

Diese Arbeit analysiert Cyber Threat Intelligence (CTI) Metriken, welche in der aktuellen Literatur vorgeschlagen werden, auf ihre Verwendbarkeit innerhalb Structured Threat Information Expression (STIX)-basierten SOAR Plattformen und implementiert eine erste Version eines solchen Frameworks. Mehrere Metriken werden mit einer Liste an Anforderungen von SOAR Plattformen verglichen. Nachdem verwendbare Metriken identifiziert sind, werden diese innerhalb eines Frameworks implementiert, welches in eine existierende SOAR Plattform eingebunden wird. Schlussendlich wird dieses Framework getestet und die berechneten Metriken diskutiert.

Als Fazit kann gesagt werden, dass bereits Metriken existieren welche abgeändert in SOAR Plattformen eingesetzt werden können. Jedoch verlangen manche Metriken Parameter welche nicht einfach von SOAR Plattformen zur Verfügung gestellt werden. Das bedeutet, dass auch SOAR Plattformen auf die Anforderungen eines Datenqualitätsframeworks eingehen müssen.

CHAPTER 1

Introduction

The cyber threat landscape is vast and ever-changing, with organizations progressively integrating Cyber Threat Intelligence (CTI) into their defences against cyber attacks [1]–[5].

This thesis presents an architecture for analyzing different sources providing CTI within a Security Orchestration, Automation, and Response (SOAR) platform. Each source analyses suspicious data and returns intelligence. This intelligence is analyzed and weighted against other sources, providing insight into how much intelligence a source offers and how much impact that information has.

This chapter describes the motivation behind an automated assessment of CTI sources, followed by the goal of the thesis. Lastly, the structure of the work is given.

1.1 Landscape

With more devices connected to the internet, cyber security gains importance. Web-enabled devices also mean a potential attack vector for adversaries [6]–[8]. And once attackers have successfully attacked an organization, there is a possibility of substantial losses to productivity, capital or customers [9].

As attacks become more sophisticated, companies can no longer depend on simple firewalls and virus scanners to keep threats at bay. Instead, CTI is used by a Security Operations Center (SOC) to gain enough insight into the current threat landscape to respond to emerging threats appropriately. This CTI cannot be created by a single company alone, so CTI sharing has become commonplace with communities and organizations exchanging threat information on marketplaces [10]. Several standards have been proposed for CTI sharing, such as Open Indicators of Compromise (OpenIOC), Cyber Observable Expression (CybOX) or Vocabulary for Event Recording and Incident Sharing (VERIS), with Structured Threat Information Expression (STIX) being the most common one [11].

These marketplaces can be open for anyone to contribute, only available to participating organizations or accessible only by payment.

With open-source marketplaces often being un-curated and lacking detailed reports, organizations rely on paid, curated CTI services to satisfy their cyber security requirements, with 44% of companies choosing to do so [12].

These feeds can also be used for forensic purposes. Suspicious data may be captured by security analysts, which then needs to be analyzed. For this, SOAR platforms can be used, which present analysts with an interface to upload, annotate and review files [13]. By uploading suspicious data, a case is opened where the uploaded file is automatically analyzed and enriched by multiple background services. The resulting data of the services are then merged into an overview which can be viewed by the analyst and annotated.

What is unclear is the quality produced by these services. Services rarely self-evaluate themselves, and quality is highly dependent on how the service is used and in what context. Some services provide intelligence on multitudes of topics where it is possible that one subject is covered in great length while other topics are served better by other providers. Studies within this field are getting quickly outdated, are using a generic approach not applicable for the often specific domains of cyber security or are anonymizing the feeds to preserve privacy but leading to no business value as quality services cannot be discerned [11], [14], [15].

1.2 Goal

This work aims to find metrics for CTI quality measurement and implement them within a framework able to evaluate services and their results. The calculated scores for each service should be able to be compared against each other and should not rely on long-term observation of the source. Several approaches to quality scoring are evaluated; viable metrics are chosen, implemented and augmented to satisfy the requirements set by the nature of SOAR platforms, namely case-based analysis of threats. The following research question will be the focus of this work: *How can data quality be measured within Security Orchestration, Automation, and Response platforms?* To ease answering this question, the following sub-question have been formulated:

1. What CTI measurements do already exist?
2. How can these be altered to work with CTI data of a SOAR platform?
3. How can a CTI rating framework be incorporated into a SOAR platform?

1.3 Involved Stakeholder

The company Nextpart, located in Linz, Upper Austria, specializes in cyber security and offers consulting as well as tools to avoid cyber threats. Its main product, Guardia,

also called *SOC-Toolkit*, specializes in the enrichment of security incidents by offering automated playbooks for security analysts. These playbooks enrich incidents by aggregating information about the incident from multiple CTI sources. As a stakeholder in this thesis, Nextpart provides feedback on the implemented algorithm.

1.4 Structure

The work is structured as follows: Chapter 2 provides background information and related work about the topic. The basics about CTI and its tools are explained. Chapter 3 compares different methodologies for assessing CTI sources. Multiple approaches from literature are introduced, and their effectiveness in SOARs are discussed. The following Chapter 4 describes the implementation of the proposed architecture, with all components used within the framework explained in detail. The framework is then tested in Chapter 5, where multiple STIX bundles are analyzed with the framework. The results are visualized and analyzed in the following Section 6.1, where the work concludes with a summary and outlook.

CHAPTER 2

Background

This chapter will start with the definition of cyber threat information and intelligence. Then, the used Cyber Threat Intelligence format Structured Threat Information Expression is described. This format is then used by Security Operations Center and, more specifically, Security Orchestration, Automation, and Response platforms, both of which are described. Then, the necessary data orchestrators are described, which are needed to run playbooks used by SOAR platforms. Finally, a literature review is done.

2.1 Cyber Threat Information and Intelligence

Cyber threats, as defined by the Computer Security Resource Center of the National Institute of Standards and Technology, are circumstances or events with potentially adverse effects on organisations, their operations and assets or individuals via unauthorised access, destruction, disclosure, modification of information or denial of service [16]. Cyber threat information can be used to mitigate, prevent and analyse attacks on computer infrastructure. It provides information about how systems can be attacked, what files may be malicious or what zero-day exploits are currently threatening the integrity of servers. Cyber threat information can be converted into CTI by transforming and enriching it to provide a context for decision-making [17]. CTI can be created from an organisation's internal systems or be received from public or paid web services which specialise in CTI aggregation.

2.2 Structured Threat Information Expression

STIX can be used to exchange CTI [18]. It allows organisations to share threat intelligence using a clear and defined standard. Based on the Java Script Object Notation (JSON), version 2 minimizes its memory footprint compared to version 1's Extensible Markup Language (XML) representation while maintaining its human readability [19]. An example

of a STIX JSON can be seen in Listing 2.1. Currently, most services provide and support version 2 of STIX. From this point onwards, STIX refers to STIX 2.1, unless mentioned otherwise.

```
{
  "created": "2022-07-23T13:02:04.599135Z",
  "modified": "2022-07-23T13:02:04.599135Z",
  "spec_version": "2.1",
  "type": "email-message",
  "id": "email-message--b3bf4d69-660e-550d-b6b9-76e81346b3ef",
  "date": "2022-03-09T11:56:04.000000Z",
  "content_type": "text/html",
  "from_ref": "email-addr--6f9b26ba-98c3-5dba-ac4c-5db89ec7bef2",
  "to_refs": [
    "email-addr--81122ca3-e9bc-5ab4-83ff-73e46c618359"
  ],
  "subject": "How's the master thesis going?"
}
```

Listing 2.1: Example of a STIX 2.1 JSON file representing an email object.

A STIX bundle acts as a container for arbitrary STIX objects. These objects are not required to be connected or have anything in common. STIX objects are further divided into two groups, STIX Core Objects, which are used to represent CTIs, real-world instances and relationships between them and STIX Meta Objects (SMO), which are used to supplement some STIX objects with metadata for user and system workflows, see Table 2.1.

STIX Core Objects are further broken down into three classes:

- STIX Domain Object (SDO)
- STIX Cyber-observable Object (SCO)
- STIX Relationship Object (SRO)

<table>
<tr><th colspan="6">STIX Objects</th><th rowspan="3">STIX Bundle Object</th></tr>
<tr><th colspan="3">STIX Core Objects</th><th colspan="3">STIX Meta Objects (SMO)</th></tr>
<tr><th>STIX Domain Objects (SDO)</th><th>STIX Cyber-observable Objects (SCO)</th><th>STIX Relationship Objects (SRO)</th><th>Extension Definition Objects</th><th>Language Content Objects</th><th>Marking Definition Objects</th></tr>
</table>

Table 2.1: The taxonomy of STIX. Source [18].

STIX Domain Object (SDO) objects are some of the most common STIX objects encountered. They represent concepts of CTI such as infrastructure, identities, locations

or incidents. These domain objects convey information about specific entities relevant to the current case. For example, an *Identity* object must include a `name` property, but may also possess an optional `contact_information` property telling security analysts how to contact said identity.

STIX Cyber-observable Object (SCO) objects are used to enrich SDOs by offering information about host-based or network-based occurrences. For example, an application running or specific network packages being sent may be described in an SCO and then connected to an SDO via an SRO.

STIX Relationship Object (SRO) objects indicate relationships between two SDOs or SCOs. These relationships can link an SDO or SCO to another SDO or SCO with the `type` property specifying the kind of relationship, i.e. using the type `exploits` to link a malware object to a vulnerability. SCOs are used to show the interaction between different objects in the graph. Some relationships are embedded within the objects themselves, most importantly `created_by_ref` which connects an object to its creator identity.

2.3 Security Operations Center

SOC are used by security analysts to overview, enter and analyse cases within their domain of responsibility [20]. SOCs can act as an interface between the human security expert and the underlying architecture of threat identification. SOCs can be used to enter possible indicators of attacks or unknown files, which are then automatically analysed, and the results are shown. These cases can then be visualised for easier consumption and annotated for further investigation or reference if another similar case occurs.

2.4 Security Orchestration, Automation, and Response

SOAR platforms are used to collect and view data from various security software services [21]. These platforms try to solve the problem of having too many manual processes while analysing security incidents, as manual processes may increase the probability of missing alerts, loss of critical response time via manual data enrichment, and having non-standardised workflows [21]. According to Lawson and Price [13], SOAR solutions have grown out of three different technologies, namely Security incident response platform (SIRP), Security orchestration and automation (SOA) and Threat intelligence platform (TIP), as Figure 2.1 shows.

SOAR platforms are using automated processes to handle security incidents faster by using predefined *playbooks* [21]. These playbooks are composed of different services based on the incoming incident type.

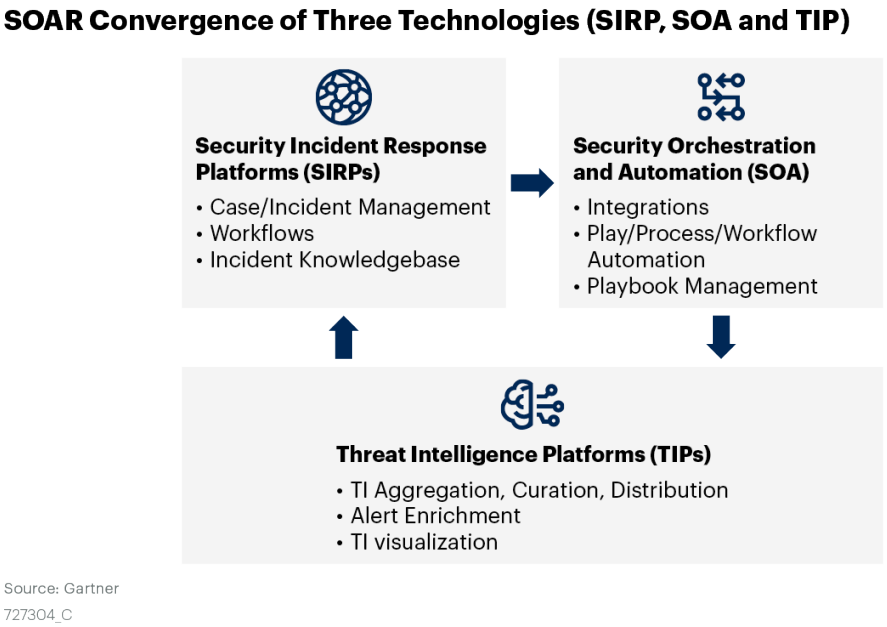


Figure 2.1: Components of a SOAR platform. Source [13].

2.5 Data Orchestrator

A data orchestrator is used to transform input data via multiple different job stages into an output with the help of an orchestration graph [22]. Input and output, as well as the different job stages, called *Op*s, are defined by the software architect, and their functionality depends on the given data orchestrator. Ops are atomic pieces of code which can be reused in multiple jobs, reducing code duplication.

2.6 Playbook

SOAR platforms rely on playbooks to function. These playbooks are predefined by security analysts based on services available and previous experiences with the same incident type. To visualise the workflow a security analyst will take, following Section 2.6.1 highlights an example usage of a playbook within the Guardia software.

2.6.1 Example Use Case

A security analyst gets forwarded a suspicious email by an employee who is unsure if it is real, spam or even malicious. To get an overview of possible threats contained within the email, a tool like a SOAR platform can be used to analyse said email.

First, the security analyst downloads the email and saves it as a `.eml` or `.msg` file. Then the web interface of the SOAR platform can be accessed. The welcome screen allows the security analyst to upload the email via drag-and-drop or via a file selection window. Uploading a file selects a playbook based on the file type and file properties. In Figure 2.2, the email file is being uploaded, so the playbook *Direct Email Playbook* is selected. This playbook is further customisable by adding tools to the investigation run. These tools can look up WHOIS entries for domains contained within the email body or try to see if the sender's email is contained within a blocklist. When the security analyst is satisfied with the services selected, the playbook can be started by clicking the *Investigate* button on the bottom of the interface.

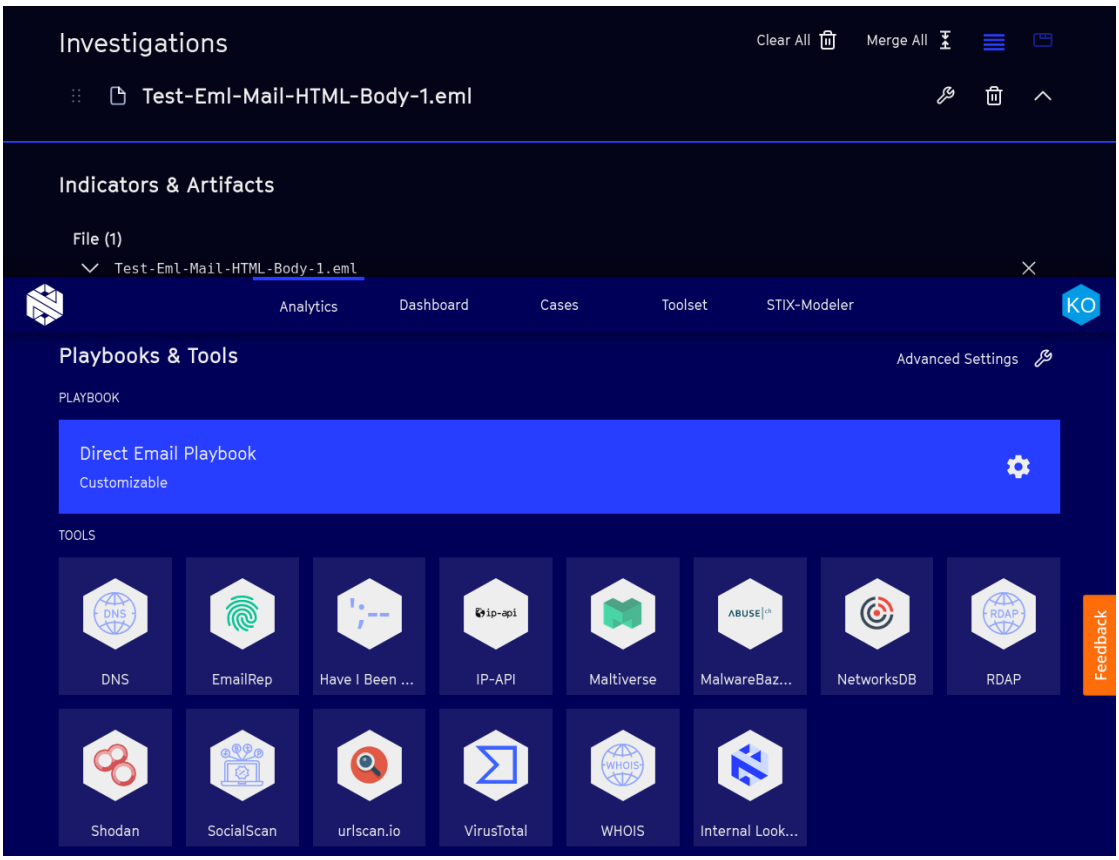


Figure 2.2: An email file being uploaded to Guardia.

After the playbook has been launched, Guardia will automatically execute all selected services and aggregate the results to a STIX graph. For a detailed technical explanation about playbook execution refer to Section 4.3.2. Depending on the number of services chosen, the playbook will run for a few minutes.

After the playbook is finished, the security analyst is redirected to a page showing the finished playbook from which a decision can be made if the email is a threat or not. A possible view of the case is presented in Figure 2.3. As can be seen, the graphs can be extensive.

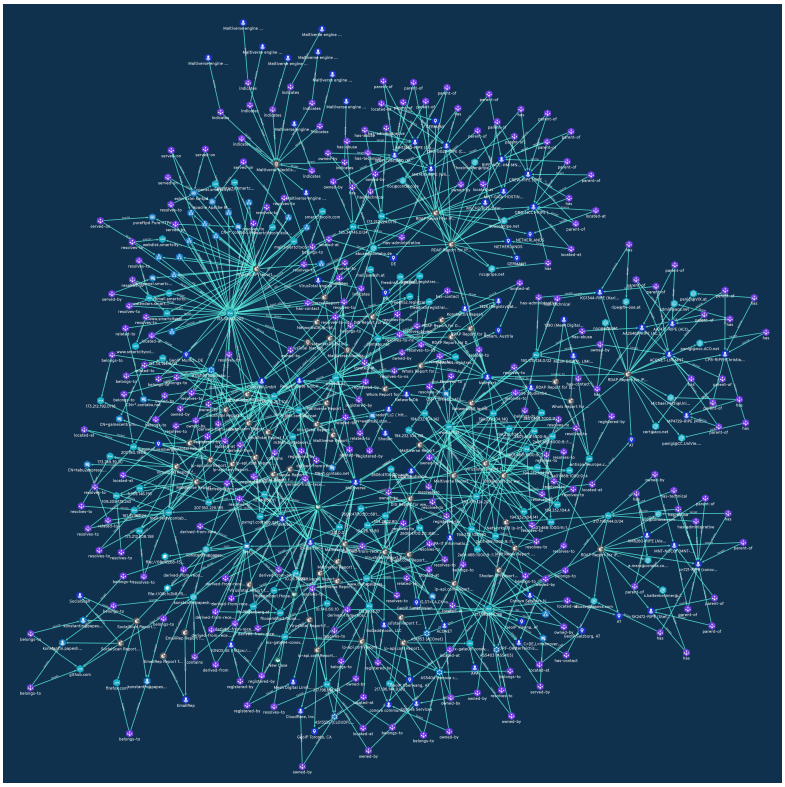


Figure 2.3: The resulting STIX graph as displayed by Guardia.

As can be seen from the already long list of integrations in Figure 2.2, the correct choice of integration is important. Selecting all possible integrations is a naive and unviable approach; with growing integrations, the run time and expenses will increase with every service selected. For every additional service selected, computation time is added by sending requests and parsing responses. Additionally, some services require a fee for usage; this fee is always applied, even when the service returns no to little intelligence.

Rating received CTI may benefit security analysts, software architects and platform consultants alike. For security analysts, the knowledge of which service provides more structured and relevant CTI supports them in deciding which services can be used within SOAR playbooks. Software architects on the other hand can use the metrics to verify transformation implementations as some services do not return STIX; with these services, transformers need to be used to convert the structured response into a STIX bundle. Having the ability to check these bundles for possible missing properties or loss of quality may be a welcome addition to the development process. Additionally, persisting the

metrics for each service after a playbook run may show quality trends of services; some services may improve their service quality while some may have a decrease in quality. A sudden drop in quality may indicate a change in data format; when a transformer is used and the format changes, properties may not be mapped properly anymore.

Finally, having a numeric metric allows security consultants to recommend services for customers to use in their playbooks based on the customers' field of technology. For example, for a new customer in the banking sector, the security consultant can review the metrics of already existing customers in the same sector. Knowing which services return the most suiting results may increase customer satisfaction from the start of the partnership.

2.7 Related Work

While SOCs are getting more common currently, automated aggregation of CTI is uncommon. Even more so is automated evaluation and scoring of said aggregated CTI.

Systems dealing with automated STIX generation and automated evaluation focus on unstructured intelligence [23], which is not applicable for already structured STIX data. These approaches often use artificial intelligence to classify data for importance. With already structured data, this approach may be less accurate and more wasteful of resources than simple metrics.

One paper [24] introduces four metrics for CTI feeds; *Timeliness*, *Sensitivity*, *Originality* and *Impact*. These metrics however are not based on live measurements, they incorporate data from a specific time frame, i.e. 7 months and need additional, worldview data to calculate a score for sources. This additional worldview data is also not available or needs extensive crawling and collecting for some cases of Guardia. Another work [25] also includes some metrics, *Volume*, *Differential contribution*, *Exclusive contributions*, *Latency*, *Accuracy* and *Coverage*. Again, these metrics are based on a long-term view of the feeds and are not applicable for case-based scores. *ETIP* [26] has a similar approach to calculating metrics for given data input, however, they use more complex heuristic features taking into account every possible type of STIX object. Each property of every type needs to be assigned a weighting score manually, leading to overhead for security analysts. Additionally, the metrics are again more relevant for worldview type of application than for single-case application, i.e. the weight criteria *Relevance* calculates if the received CTI is mentioning assets in your domain; with case-based analysis, with services just returning information for a single asset, this metric is of little importance. Ramsdale *et al.* [27] give a broad overview of sources, formats and languages. While mostly concerned about the general usage of different CTI formats, minor analytics are applied to STIX data, however, no metrics are composed or applied broadly to data. Wagner *et al.* [14] analyze different CTI platforms based on their original sources, vetting processes and trust mechanism. It focuses mainly on establishing trust and a total of 30 services are compared. A trust taxonomy is proposed and compared against other trust taxonomies. Menges and Pernul [28] give insight into multiple CTI exchange formats

by defining some evaluation criteria and then comparing standards with them. Four standards and two different versions of standards are compared. The work of Sauerwein *et al.* [11] focused on multiple CTI sharing platforms by performing a systematic study. 22 CTI platforms were compared based on functionalities and capabilities. The research of Serrano *et al.* [29] gives an overview of the major challenges of CTI sharing and proposes some solutions for them. Wagner *et al.* [2] discuss CTI sharing in general, visiting a few key points such as automated sharing, risks, trust and privacy. It includes an extensive review of literature related to CTI sharing, including a chapter defining and reviewing relevancy in CTI.

All discussed works only analyze the received data itself without taking the process of receiving the data into account.

Most implementations aggregate received STIX into a combined worldview, while Guardia takes suspicious files, emails or any other type of possible malicious content and tries to analyze and enrich it. Thus, the metrics and scores calculated from this approach must also be calculated differently compared to most other approaches available in the literature.

CHAPTER 3

Possible Approaches

This chapter introduces different approaches to quantify quality and information in CTI. First, the general requirements are specified based on the nature of SOAR platforms and their case-based approach. Then, various metrics from the literature are briefly explained and compared. Finally, an approach is proposed, which is then implemented in Chapter 4.

3.1 Requirements

The chosen metric should be able to reflect the quality and information of all participating services of a playbook run by providing one or more metrics which a security analyst can examine. These metrics should give a clear indication of what the source does best and in what areas it is lacking. For example, a service may provide detailed malware reports, but without any references to support them.

Important to note is that the calculation of metrics should happen with every playbook run. The metrics should also consider metrics that result from this case-based approach, i.e. the time until the service responds or how much one call to the service costs. Another important addition is also that the data to be analyzed is in the STIX data format, so the metrics should be able to handle graph data.

So based on these requirements, the following points have been noted for metrics approaches to test against:

- Able to rate CTI in the data format STIX
- Able to compare different sources between each other
- Able to be used on a snapshot

- Extensible
- Without human input
- Have a continuous value

Snapshot in this context means it should be possible to rate a single STIX bundle without any additional historical data.

3.2 Possible STIX Metrics

The following section highlights approaches found in literature to rate CTI data.

3.2.1 FeedRank by Meier *et al.*

FeedRank by Meier *et al.* [30] is a feed ranking approach developed in 2018 at ETH Zürich. It is based on the PageRank [31] algorithm, made famous by its usage by the search engine *Google*. It assumes that when a source is more relevant, other feeds will incorporate its contents within their feed.

The calculation of the FeedRank works in three stages, as seen in Figure 3.1.

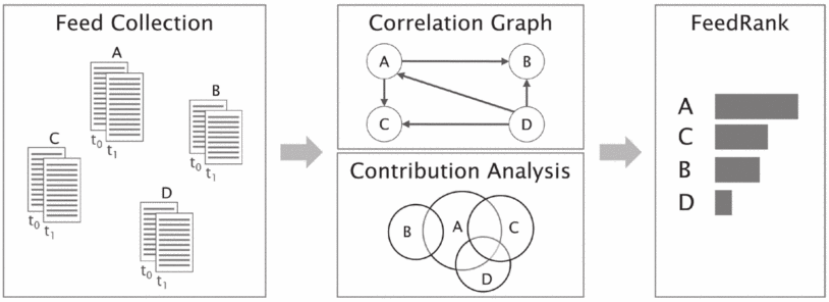


Figure 3.1: The three steps of FeedRank: Feed Collection, Correlation Graph & Contribution Analysis and Feed Rating. Source [30].

The correlation graph works by taking at least two snapshots and comparing entries between them. If source A provided entry X at time t and source B provided X at time $t+1$, source B implicitly confirms source A. Weighting is done based on how many entries are from another source divided by the total sum of entries of a source. Additionally, a dampening factor d is calculated by taking the average path length l, based on the average number of feeds that list an entry and dividing one by it, see Equation (3.1).

$$d = P(\text{continue}) = 1 - P(\text{stop}) = 1 - \frac{1}{l} \tag{3.1}$$

The contribution analysis yields a metric measuring how much of the worldview is supplied by a single CTI source. This measurement is calculated by determining which

source listed an object first. Meier *et al.* [30] do not describe the FeedRank equation mathematically. But given the PageRank equation taken from Brin and Page [32], see Equation (3.2), with the addition of N, which is omitted from the formula in the paper but required for the statement „Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one." (Brin and Page [32]) to hold true [33], Equation (3.3) can be formulated for the FeedRank.

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)} \tag{3.2}$$

$$FR(t_i) = \frac{1-d}{N} + d \sum_{t_j \in in(t_i)} \frac{FR(t_j)}{|out(t_j)|} \tag{3.3}$$

where t_i is a source within the worldview, d is the dampening factor, N is the number of sources observed, $in(t_i)$ are the sources linking to t_i and $|out(t_j)|$ is the number of sources referenced by t_j.

3.2.2 Trust and Quality Tool by Mavzer *et al.*

The Trust and Quality Tool (TQM) by Mavzer *et al.* [34] is a plugin for the CTI sharing software ECHO - Early Warning System (E-EWS), designed at Bournemouth University in cooperation with VisionSpace Technologies and Telefonica Global Services. It is designed to measure the trust and quality of CTIs within the E-EWS system, and in their work, the authors defined multiple metrics to measure trust and quality metrics.

The work differentiates between two metrics, *trust* and *quality*, which are broken down into multiple metrics. The *Trustworthiness* metric contains the metrics *Partner Sharing Activity* (p.s.a.), *Sector of Activity (s.c.)*, *Certified Cybersecurity* (c.c.), *Previous Ticket Ratings* (p.t.r.), *Privacy* (p.) and *Partner Sector* (p.s.). The metric *Quality* is defined by *Completeness* (t.c.), *Freshness* (t.f.), *Timeliness* (t.t.), *Extensiveness* (t.e.) and *Relevance* (t.r.).

Unfortunately, the underlying metrics of the trustworthiness parameter are not further defined; while there is a textual description of how they are determined, no formulas are given to implement said metrics programmatically. For example, the description for *Privacy* says, „The metric is defined by an external tool to check if the content includes personally identifiable information" (Mavzer *et al.* [34]), this leaves much of the definition and implementation to the programmer.

However, the metrics for the quality parameter are more defined and can be implemented as is. The authors also list two kinds of computations; one initial computation and one formulated after their research; only the improved version will be described in the following pages. The improvements are based on feedback from experts and try to mimic their rating of certain metrics.

Completeness

Completeness reflects the number of filled-in mandatory parameters. The computation happens based on fixed values; a base score of +40 is assigned to every object. If the object contains at least one attachment, a further +20 is added. One or more references yield an additional +15, one or more facets +25.

Extensiveness

The extensiveness metric is based on the number of attachments, references or facets connected to the object. It is not based on a continuous function; instead, a table is used to assign a score to the object, see Table 3.1. For example, an object containing one attachment, four references and two facets will receive the score $10 + 20 + 25 = \mathbf{55}$.

	Count				
Type	1	2	3	4	5
Attachments	+10	+25	+35	+35	+35
References	+10	+15	+20	+20	+30
Facets	+15	+25	+35	+35	+35

Table 3.1: The extensiveness computation [34].

Freshness

Freshness is the currency of the information presented in the object. Like the extensiveness metric, it uses a table-based scoring, with more recent objects receiving a higher score than older objects, see Table 3.2.

Days	Score
0-3	100
3-7	90
7-15	80
15-30	70
30-90	60
90-120	50
120-150	40
150-180	30
180-270	20
270-365	10
365+	0

Table 3.2: The freshness computation [34].

Aggregation of metrics

The two base metrics, *Trustworthiness* and *Quality* are calculated using a weighted sum of their sub-metrics. Equation (3.4) shows the calculation of the trustworthiness, Equation (3.5) the calculation of quality, where $w(x)$ is a function returning the weight for metric x. Finally, both metrics are combined into the CTI rating *CTI.R.* by simply adding both metrics together, as seen in Equation (3.6).

$$T = w(p.s.a.) * p.s.a. + w(s.c.) * s.c. + w(c.c.) * c.c. + w(p.t.r) * p.t.r. + w(p.s.) * p.s. \quad (3.4)$$

$$Q = w(t.c.) * t.c. + w(t.f.) * t.f. + w(t.t.) * t.t. + w(t.e.) * t.e. + w(t.r.) * t.r. \quad (3.5)$$

$$CTI.R. = T + Q \quad (3.6)$$

3.2.3 Measuring and visualizing cyber threat intelligence quality by Schlette *et al.*

In 2020, Schlette *et al.* [35] introduced data quality (DQ) metrics based on STIX. The authors propose three different levels of DQ dimensions, *Report*, *Object* and *Attribute*, see Figure 3.2. Before assessing any properties, the authors defined a ground truth to base their calculations on, see Equations (3.7) to (3.11).

Two sets of attributes are defined, required attributes A_r, see Equation (3.7), and optional attributes A_o, see Equation (3.8). Furthermore, STIX Domain Objects and STIX Relationship Objects are a specific subset consisting of required and optional attributes, see Equation (3.9). All objects contained in the worldview are described by O and may be of type SDO or SRO, see Equation (3.10). Finally, a report R consists of a subset of all objects contained in O, see Equation (3.11).

$$A_r = \{a_r | a_r \text{ required by STIX 2}\} \quad (3.7)$$

$$A_o = \{a_o | a_o \text{ optional in STIX 2}\} \quad (3.8)$$

$$SDO, SRO \subseteq (A_r \cup A_o) \quad (3.9)$$

$$O = \{o | o \in (SDO \cup SRO)\} \quad (3.10)$$

$$R = \{r \subseteq O\} \quad (3.11)$$

Concise representation

Concise representation tries to quantify duplicate entries in the data. It does so by comparing one object with all the other objects via semantic text similarity. The authors propose the *Simhash* [36] algorithm as an example. A threshold t is set, if the similarity of an object to any other object o_2 stays below it, it is classified as *unique*, else it is classified as *duplicate*, see Equation (3.12).

$$CR(o) = \begin{cases} 1 & \text{if } similarity(o_1, o_2) < t \\ 0 & \text{else} \end{cases} \quad (3.12)$$

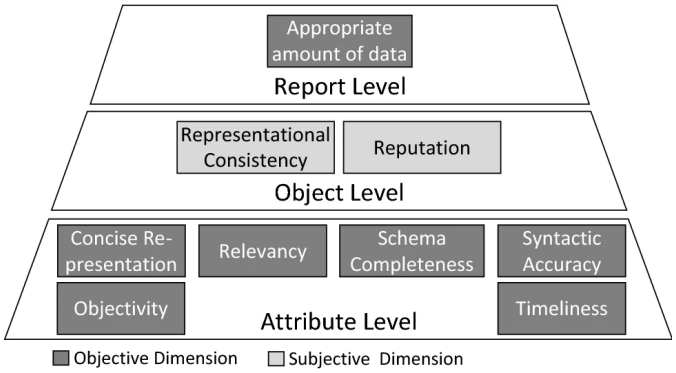


Figure 3.2: Three data quality levels. Source [35].

Objectivity

Objectivity tries to classify STIX objects with free-text fields into objective or subjective categories. The authors suggest a thorough investigation before a sentiment algorithm is chosen. The classification is applied to every attribute of the object, see Equation (3.13), which are then aggregated to calculate an objectivity metric for the whole object, see Equation (3.14).

$$OB(a) = \begin{cases} 1 & \text{if } v(a) \text{ classified as } \textit{objective} \\ 0 & \text{if } v(a) \text{ classified as } \textit{subjective} \end{cases} \tag{3.13}$$

$$OB(o) = \frac{\sum_{a \in o} OB(a)}{|o \cap (A_r \cup A_o)|} \tag{3.14}$$

Relevancy

Relevancy quantifies the value of the object to the security analyst. It does so by matching attribute values of certain STIX objects to respective relevancies set by the security analyst. For example, a STIX *Intrusion Set* may contain the property `goals` set to "leak bank information of celebrities". This increases the relevancy of the object for banking institutions. Equation (3.15) shows the calculation of the relevancy with PV_c being the property values of the customer, PV_p of the publisher and PV_o of all relevant STIX objects.

$$RE(o) = \frac{|PV_c \cap (PV_p \cup PV_o)|}{|PV_p \cup PV_o|} \tag{3.15}$$

Schema completeness

Schema completeness reflects how many optional attributes of an object are missing. These attributes are not required by the STIX standard but may offer additional information for security analysts.

To calculate the schema completeness, all optional attributes have to be counted, see Equation (3.16), and then the sum of all optional attributes is divided by the amount of possible optional attributes, see Equation (3.17).

$$SC(a_o) = \begin{cases} 1 & \text{if } v(a_o) \neq \texttt{NULL} \\ 0 & \text{else} \end{cases} \tag{3.16}$$

$$RE(o) = \frac{\sum_{a_o \in (o \cap A_o)} SC(a_o)}{|o \cap A_o|} \tag{3.17}$$

Syntactic accuracy

Syntactic accuracy validates the received objects against the STIX 2.1 JSON schema provided by OASIS[1]. The metric expressed in Equation (3.19) only counts an attribute if its data type and value are valid within the schema D. To get the syntactic accuracy for the whole object, all attributes are validated and divided by all filled-out attributes of the object, as seen in Equation (3.19).

$$SA(a_o) = \begin{cases} 1 & \text{if } v(a) \in D \\ 0 & \text{else} \end{cases} \tag{3.18}$$

$$SA(o) = \frac{\sum_{a \in o} SA(a)}{|o \cap (A_r \cup A_o)|} \tag{3.19}$$

Timeliness

Timeliness shows the currency of the object. The authors propose three different approaches for timeliness. $TI_{Basic}(o)$, Equation (3.20), uses two metrics; *Volatility* and *Currency*. *Volatility* describes the number of changes applied to the object while *Currency* describes the time since the last change.

$$TI_{Basic}(o) = \frac{1}{(Currency(o) \times Volatility(o)) + 1} \tag{3.20}$$

$TI_{Statistical}(o)$, Equation (3.21), takes the approach of an exponential decline. It suggests that CTI slowly loses its value with time passing, as adversarial network addresses get reassigned, and malware falls out of use. Falloff depends on the type of object. For example, the identity information of attackers stays mostly the same, so no decrease in timeliness is observed.

$$TI_{Statistical}(o) = \exp(-Decline(o) \times Currency(o)) \tag{3.21}$$

[1] `https://github.com/oasis-open/cti-stix2-json-schemas`(Accessed 2022/07/29)

$TI_{Assisted}(o)$, Equation (3.22), includes subjective parameters defined by a security analyst. If the object has been modified prior to a set threshold, it is classified as outdated; else, it is still timely.

$$TI_{Assisted}(o) = \begin{cases} 1 & \text{if } t_{current} - t_{last} < \text{threshold} \\ 0 & \text{else} \end{cases} \tag{3.22}$$

Representational consistency

Representational consistency validates objects based on the content of attributes. It checks if the property values do not break any logical rules; for example, a *Relationship* object can not be created before the two objects it links to are created. It also cannot link to non-existent objects. An object may have multiple conditions, see Equation (3.23), which are then aggregated per object, see Equation (3.24).

$$c_j(o) = \begin{cases} 1 & \text{if } o \text{ fulfils condition } c_j \\ 0 & \text{else} \end{cases} \tag{3.23}$$

$$RC(o) = \prod_{j=1}^{|C|} c_j(o) \tag{3.24}$$

Reputation

Reputation is a mostly subjective metric for objects. It is designed around experts' rating sources or data sets based on reputation. For example, a professional CTI provider may receive a higher rating than a random company sharing their intrusions. Reputation is calculated on a per-source base, see Equation (3.25).

$$RS(p) = \{s \mid 1 \leq s \leq 5 \wedge s \in \mathbb{N}\} \tag{3.25}$$

Appropriate amount of data

This metric measures if the source provides enough data for the user to accurately act upon it. Schlette *et al.* [35] suggest that this metric needs more research as there is no clear definition of the appropriate amount of data yet. The proposed metric weights the connectedness of the report by comparing the existing relationships with the amount of maximum possible number of relationships, see Equation (3.26).

$$AD(r) = \frac{|sro \in r|}{\frac{|sdo \in r|(|sdo \in r|-1)}{2}} \tag{3.26}$$

Aggregation of metrics

Finally, to aggregate all collected metrics, the authors propose a weighted average summation of all scores, see Equation (3.27). The security analyst can set the weights or

leave them empty for a default score of 1. Alternatively, the DQ can also be measured on a report level; see Equation (3.28).

$$DQ(o) = \frac{\sum_{d \in D} d_i \cdot w_i}{\sum_{d \in D} w_i} \tag{3.27}$$

$$DQ(r) = \frac{(\sum_{o \in r} DQ(o)) + AD(r) \cdot w}{|r| + w} \tag{3.28}$$

3.2.4 Quantitative Evaluation of Trust in the Quality of Cyber Threat Intelligence Sources by Schaberreiter *et al.*

The paper of Schaberreiter *et al.* [37] proposes a scoring based on STIX. Ten different quantitative evaluation parameters are introduced, from which a total CTI source score, the so-called *Trust Indicator*, is calculated.

Extensiveness

Extensiveness quantifies the amount of information a source compiles into a single object. As mentioned in Section 3.2.3, the standard defines two property types for objects; required and optional. The extensiveness parameter shows how many optional parameters are filled out compared to the maximum amount of optional parameters; see Equation (3.29).

$$p_1 = \frac{1}{z} \sum_{i=1}^{z} \left(\frac{o_i}{\max y_i} \right) \tag{3.29}$$

where o_i is the number of properties set in object i and $\max y_i$ is the maximum number of properties supported by the type of i. z represents the number of objects supplied by the current source.

Maintenance

The Maintenance parameter indicates how often a source updates its CTI. Additionally, the update count is weighted by the average update count of all sources within the worldview, see Equation (3.30).

$$p_2 = \left\| \frac{\frac{1}{z} \sum_{i=1}^{z} u_i}{\text{avg}(p_2 s_1, \ldots, p_2 s_n)} \right\| \tag{3.30}$$

where u_i is the number of updated objects of the current source.

False Positives

The STIX standard allows for revocation of information by setting the parameter `revoke` to `true`. The authors argue that simply counting the number of invalidated objects will not result in a useful metric, as trustworthy sources will revoke outdated objects while

some lesser-maintained ones will not. Equation (3.31) will reflect this fact by weighting the number of revoked objects of a source with the total amount of revoked objects in the worldview.

$$p_3 = 1 - \left(\frac{F_{s_x}}{\sum_{i=1}^{n} F_{s_i}} \right) \tag{3.31}$$

where F_{s_x} is the number of false positives of the current source and F_{s_i} is the number of false positives of source i.

Verifiability

Objects can reference its information sources via the property `external_reference`. By counting these references and comparing them to the average number of references in the worldview, see Equation (3.32), it can be deducted if a source's references are above average.

$$p_4 = \left\| \frac{\left\| \frac{1}{z} \sum_{i=1}^{z} r_i \right\|}{\text{avg}(p_4 s_1, \ldots, p_4 s_n)} \right\| \tag{3.32}$$

where r_i is the number of references provided by the current source.

Intelligence

According to Schaberreiter *et al.* [37], an important aspect in CTI is how many links a threat intelligence parameter has. To calculate the intelligence metric, a count of relationships attributed to the source is done as seen in Equation (3.33).

$$p_5 = \left\| \frac{\left\| \frac{1}{z} \sum_{i=1}^{z} l_i \right\|}{\text{avg}(p_5 s_1, \ldots, p_5 s_n)} \right\| \tag{3.33}$$

where l_i is the number of links from and to object i.

Interoperability

The interoperability score reflects the format used by the source. While Guardia uses STIX as the format for all metrics, sources may use a different standard to convey information, such as OpenIOC, or even no standard at all, such as Comma-seperated Values (CSV). In most cases, the data can be transformed into STIX; this means additional workload and may result in loss of information. Therefore, sources with different formats should be rated lower than the ones already providing the correct format. While this parameter could also be set empirically by a security expert rating the different standards, the authors propose Equation (3.34) to score sources, where b_i is the number of sources using the same format as the source to be rated divided by the total number of sources.

$$p_6 = \sum_{i=1}^{n} \left(\frac{b_i}{n} \right) \tag{3.34}$$

Compliance

Compliance checks if the source violates any restrictions set by the standard. This can be facilitated by using a validator against the official schema of the standard. All objects from a source are checked for compliance, and only when they are fully compliant are they counted in c_i, see Equation (3.35).

$$p_7 = \left\| \frac{\left\| \frac{1}{z} \sum_{i=1}^{z} c_i \right\|}{\text{avg}(p_7 s_1, \ldots, p_7 s_n)} \right\| \tag{3.35}$$

Similiarity

The similarity parameter shows how much the source contributes to the overall worldview. It does so by calculating the overlap between the sources' objects and the objects of the worldview, see Equation (3.36). For objects to be considered similar or equivalent, an additional algorithm has to be chosen to find two objects' similarity parameter. Also, the threshold to consider two objects equivalent must be set empirically by a security expert as no two objects from different sources will be 100% the same. The authors propose the two text similarity algorithms Jaccard [38] and Cosine Similarity.

$$p_8 = \frac{1}{z} \sum_{i=1}^{z} (y_i) \tag{3.36}$$

Timeliness

Timeliness indicates how fast the source publishes CTI compared to the other sources in the worldview. Using the similarity metric from above, it can be determined if another source has already published the intelligence provided by the source. If it was, Equation (3.37) describes the calculation of the timeliness parameter, with $\min t_i$ being a timestamp of the CTI first being spotted and $(t_s)_i$ being a timestamp of the source providing the CTI.

$$p_9 = \frac{1}{z} \sum_{i=1}^{z} \frac{\min t_i}{(t_s)_i} \tag{3.37}$$

Completeness

The completeness parameter indicates the contribution of the source. It shows how much of the worldview is covered by the source. Equation (3.38) details the calculation, with $|B|$ being the total number of objects in the worldview and $|A|$ being the number of objects found with different ids and not similar to the objects of the source.

$$p_{10} = \frac{|B| - |A|}{|B|} \tag{3.38}$$

Aggregation of metrics

Finally, the collected metrics must be aggregated into a single, meaningful score. This score is called *trust indicator* by the authors. Besides a weighted aggregation of current parameters, the equation also includes an ageing factor D using previous trust indicators of the source. In Equation (3.39), p_n represents each parameter of source s_x at time t. The weighing factor ω weighs parameters differently, based on set weights by the security analyst. The calculation of the trust indicator should be done in regular intervals, with t being dependant on the use case, i.e. how often sources are updated.

$$TI_{s_x}(t) = \frac{D * TI_{s_x}(t-1) + \sum_{n=1}^{m} \omega_n * (p_n)_{s_x}(t)}{D + \sum_{n=1}^{m} \omega_n} \tag{3.39}$$

3.3 Possible API Metrics

Besides the discussed STIX metrics, two additional parameters are collected; time taken by a service to respond, named *Duration* and cost per call, aptly named *Cost*. Both of these metrics are not included in the STIX bundle. Instead, they are generated by the Application Programming Interface (API) call to the services. Therefore, they need to be aggregated from the technology used to communicate with the CTI sources.

3.3.1 Duration

The duration is the amount of time a service takes to respond to a request, where a lower response time should yield a higher score. However, the response time metric should not be linear. For example, fluctuations in an API between 0.5 and 1.0 seconds should be barely noticeable in the score while higher durations, for example, 10 to 15 seconds, should be scored more severely; these long response times may be annoying for the user.

A metric to score durations was already introduced as *Timeliness* by multiple algorithms in Section 3.2. Although these metrics are focused on STIX, they can be easily modified to be also used with response times of API calls. Schlette *et al.* [35] introduce the metric $TI_{Statistical}(o)$ with an exponential falloff for objects. By changing the type of o from STIX object to API Call, the metric can also be used for the duration, as can be seen in Equation (3.40). The falloff function $Decline(o)$ can be mapped to different types of services, as some types may aggregate more data and naturally take longer. This duration factor for each kind of service is not known. Also, there exists no classification that separates CTI sources into clearly defined classes. Therefore, mapping guidelines would need to be established and then the sources in the architecture would need to be manually labelled. If no differentiation between sources is possible, a generic decline can be set.

$$DU(s) = \exp(-Decline(s) \times Currency(\overline{Req_s})) \tag{3.40}$$

where s represents a service and $\overline{Req_s}$ is the average duration of a call to the service for a specific playbook run.

3.3.2 Cost

Some APIs may not offer their data for free, instead offering pay-per-call payment plans. This numeric value is not inherently visible in the STIX bundle or in the API request. Therefore, it must be entered manually into a database and mapped to each request. As the monetary value itself is a good metric, recalculating it into a numeric metric may add abstraction to an otherwise clear metric. Additionally, a generic recalculation is not feasible as the reasonable amount of money spent on a service is highly dependent on its offered features; a simple whois service providing information about webpages has a lower reasonable price than a service offering detailed information about different vulnerabilities for a specific device.

Assigning the cost for subscription-based services poses a challenge which needs further research. An initial approach is to observe or estimate the usage of the service in the time span of the subscription cycle. Then, the cost of a single run can be approximated by dividing the total cost of a subscription cycle by the usage of the service within that subscription cycle.

A reasonable approach is to visualise the cost of an endpoint on a graphical scale. Visualizing the combined cost for one playbook run, combined with the additional information supplied by the STIX metrics, a security analyst is able to determine if a service provides adequate intelligence for its price.

3.4 Discussion

Knowing what approaches are currently proposed in the literature, it is possible to select metrics fulfilling the requirements postulated in Section 3.1, refer to Table 3.3. Additionally, some papers propose the same metrics; these can be merged.

3.4.1 FeedRank

FeedRank fails on multiple requirements. As it relies on multiple snapshots to measure what source is the original creator of CTI, it does not work with the single request setup used by Guardia. Additionally, it is also not extensible; while adding metrics beside it and doing a weighted sum is possible, there is no way to integrate them into the FeedRank itself.

3.4.2 Trust and Quality Tool

The three metrics proposed by TQM are all based on value mappings. For example, the *Extensiveness* metric maps a ranging number of attachments to a specific value. This drastically reduces comparability, as objects having more than three attachments are not scored higher than ones with exactly 3, leading to the possibility of having multiple sources receiving the same score. This would mean that additional metrics would have to be added to rank sources with a more fine-grained granularity.

Metric Name	Snapshot	Extensible	Without human	Continous
FeedRank	-	-	●	●
Completeness (TQM)	●	●	●	-
Extensiveness (TQM)	●	●	●	-
Freshness (TQM)	●	●	●	-
Concise Representation	●	●	●	-
Objectivity	●	●	●	-
Relevancy	●	●	-	●
Schema Completeness	●	●	●	●
Syntactic Accuracy	●	●	●	●
Timeliness	-	●	●	◐
Representational consistency	●	●	●	●
Reputation	●	●	-	◐
Appropriate amount of data	●	●	●	●
Extensiveness	●	●	●	●
Maintenance	-	●	●	●
False Positives	-	●	●	●
Verifiability	●	●	●	●
Intelligence	●	●	●	●
Interoperability	●	●	-	●
Compliance	●	●	●	●
Similarity	●	●	●	●
Timeliness	-	●	●	●
Completeness	●	●	●	●

● = provides property; ◐ = partially provides property; - = does not provide property; All metrics can rate STIX and are comparable. For brevity, these columns have been omitted.

Table 3.3: Overview of metrics and their fulfilment of requirements.

3.4.3 Approach of Schlette *et al.*

The approach of Schlette *et al.* [35] is further discussed in this section including the explanation of why some metrics can not be reasonably implemented for use with SOAR platforms.

Concise representation

Concise representation uses a binary classification for objects, as they are either *unique* or *duplicate*. On a single object, this violates our requirement for a continuous metric. The metric could possibly be aggregated for a source by assigning *unique* and *duplicate* a numeric value and averaging all objects from the source. However, this mirrors the approach *Similarity* suggested by Schaberreiter *et al.* [37] which uses more elaborate scoring. Therefore, the metric does not add additional information when added to the implemented framework.

Objectivity

While the objectivity measurement could in theory be continuous as it is applied to multiple properties of an object, it is argued that with STIX it will still result in binary

classification. The reason for this is that STIX objects rarely offer more than one free-text field within a single object type. Most STIX objects only contain a single property with free-text, `description`. Thus, with only a single property scoreable, objects would not receive a continuous score.

Relevancy

The relevancy metric proposed is tied closely to manual input from security analysts due to the fact that for each customer, their relevant sectors of work must be defined. Then these sectors are matched against the sectors mentioned in the received STIX bundles, resulting in a metric value only relevant to the current customer. While this metric has its relevance and may be implemented separately from the analyzer, it was decided to not include it in the metrics for the architecture.

Timeliness

Three different timeliness metrics are proposed by the work [35]. $TI_{Basic}(o)$ uses *Volatility* within its formula; this variable is the number of changes to the object in a specific period. This violates our requirement of being able to handle data on a snapshot basis.

$TI_{Assisted}(o)$ sets a fixed threshold and assigns a score based on that threshold, which leads to the score being binary; either an object is above or below the value. This violates the continuous requirement.

$TI_{Statistical}(o)$ is the only timeliness metric that could potentially be used by the analyzer framework. However, as CTI is always fetched just-in-time for cases, and the STIX property `created` represents the time of object creation, the currency of objects is always *now*. One exception to this rule are identities, as these are persisted when they are created. However, identities have a $Decline(o)$ set to a constant, as identity information potentially never gets outdated.

Reputation

The reputation metric fails two requirements. Firstly, every source needs to be rated by a human based on subjective factors at a certain point in time. Thus, to stay relevant, reputation would need to be updated regularly to reflect changes in reputation for different services. Additionally, this may impose the subjective view of the rating security analyst onto another, different security analyst, even though they could have their own reputational view of the source.

3.4.4 Approach of Schaberreiter *et al.*

This section discusses the metrics proposed by Schaberreiter *et al.* [37] and why some may not be viable for integration within SOARs.

Maintenance

Maintenance takes the number of updates applied to the object and weighs them against the average number of updates in the worldview. However, this methodology can not be applied within the domain of SOAR platforms, as playbooks are run once and thus CTI is just gathered a single time, not leaving the possibility of updated versions.

False Positives

Similar to *Maintenance*, *False Positives* also rely on gathering CTI more than once with sources revoking obsolete intelligence. However, with SOAR platforms, the only possibility of receiving revoked objects would be if they are returned within the single request sent while the playbook is run. This is unlikely, and even if revoked content is received it can simply be discarded, resulting in fewer objects attributed to the source and thus receiving a worse rating by different metrics, this maintenance metric was omitted from the analyzer.

Interoperability

Interoperability requires a manual assignment of format to each source present in the worldview. Additionally, rules need to be defined to specify how different formats are to be described. For example, some endpoints may return structured, proprietary formats; however, they should not be grouped together into the same format. To define an extensive and clearly defined taxonomy and then assign all available services within the worldview a category is out of the scope for this work.

Timeliness

Timeliness ranks different sources based on their speed of providing relevant CTI data. The problem with this metric is once again the nature of the SOAR platforms' playbooks, in which all services are only queried once, all at the same time. Thus, no service can provide CTI faster than a different service. A similar metric can be gathered from the response time of different services; however, this metric can not be discerned from STIX data and must instead be calculated via different means; see Section 3.3.1.

3.4.5 Merging metrics

Filtering metrics that can not viably be used within the Guardia architecture leaves ten viable metrics. This number can be further reduced by merging similar or identical metrics of different approaches.

Schema Completeness - Extensiveness

Schlette *et al.* [35] and Schaberreiter *et al.* [37] both propose the use of a metric scoring the completeness of an object, i.e. how many optional properties are filled out. Both

calculate this metric by dividing the number of set optional properties by the maximum possible number of optional properties.

Syntactic Accuracy - Compliance

Both metrics score the objects' validity against the STIX standard. While Schlette *et al.* [35] score every object individually, Schaberreiter *et al.* [37] approach only considers the whole source. Based on the requirements, the second approach is more fitting for Guardias use-case.

Appropriate amount of data - Intelligence

Even though both metrics weigh part of the input differently, both try to accomplish the same goal with the same input. The connectedness of the report is quantified in both approaches. Schlette *et al.* [35] weigh the existing relationships with the maximum possible number of relationships based on the existing objects in the report while Schaberreiter *et al.* [37] calculate the average amount of connections per object into account. The approach of Schaberreiter *et al.* [37] yields higher values as the divisor is smaller than in the Schlette *et al.* [35] metric, so this approach was chosen to increase the influence of the metric in the final score.

3.4.6 Refining Metrics

Some metrics described contain the metric itself in the calculation. For example, Intelligence p_5 includes the average of all p_5 metrics. This would lead to an infinite loop; therefore, it is assumed that the formula can be split into two formulas, see Equation (3.41) and Equation (3.42).

$$p_5 s_i = \left\| \frac{1}{z} \sum_{i=1}^{z} l_i \right\| \tag{3.41}$$

$$p_5 = \left\| \frac{p_5 s_i}{\text{avg}(p_5 s_1, \ldots, p_5 s_n)} \right\| \tag{3.42}$$

Additionally, while some metrics are to be normalised, no formula is introduced to accomplish this task. As seen in Equation (3.42), the average of all sources is used to divide the metric of the current source. However, this will lead to a metric value over 1 when the metric of the current source is above the average. It is therefore proposed to switch $avg()$ with $max()$. This captures the value of the metric to between 0 and 1, as required in Schaberreiter *et al.* [37]. For example, the final formula for p_5 is expressed in Equation (3.43).

$$p_5 = \left\| \frac{p_5 s_i}{\max(p_5 s_1, \ldots, p_5 s_n)} \right\| \tag{3.43}$$

Furthermore, the ground truth of Schaberreiter *et al.* [37] must be altered to include SCOs as well, as SOAR platforms use them regularly. Therefore, Equations (3.9) and (3.10) are altered by adding SCOs to the formula, see Equations (3.44) and (3.45). SCOs also contain a specific subset of required and optional attributes, so no other alterations are needed.

$$SDO, SRO, SCO \subseteq (A_r \cup A_o) \tag{3.44}$$

$$O = \{o | o \in (SDO \cup SRO \cup SCO)\} \tag{3.45}$$

3.5 Final Metrics

Finally, after excluding all unviable metrics, merging duplicates and refining the remaining metrics from different approaches, Table 3.4 reflects the final metrics, which can be used within the framework to evaluate STIX bundles from different sources.

Metric	Formula
Extensiveness	$p_1 = \frac{1}{z} \sum_{i=1}^{z} \left(\frac{o_i}{\max y_i} \right)$
Compliance	$p_2 = \left\| \frac{p_2 s_i}{\max(p_2 s_1, \ldots, p_2 s_n)} \right\|$
Representation Consistency	$p_3 = \prod_{j=1}^{\lvert C \rvert} c_j(o)$
Verifiability	$p_4 = \left\| \frac{p_4 s_i}{\max(p_4 s_1, \ldots, p_4 s_n)} \right\|$
Intelligence	$p_5 = \left\| \frac{p_5 s_i}{\max(p_5 s_1, \ldots, p_5 s_n)} \right\|$
Similarity	$p_6 = \frac{1}{z} \sum_{i=1}^{z} (y_i)$
Completeness	$p_7 = \frac{\lvert B \rvert - \lvert A \rvert}{\lvert B \rvert}$
Duration	$p_8 = \exp(-Decline(s) \times Currency(\overline{Req_s}))$

Table 3.4: The final metrics and their corresponding formulas.

where $p_2 s_i = \left\| \frac{1}{z} \sum_{i=1}^{z} c_i \right\|$, $p_4 s_i = \left\| \frac{1}{z} \sum_{i=1}^{z} r_i \right\|$ and $p_5 s_i = \left\| \frac{1}{z} \sum_{i=1}^{z} l_i \right\|$.

CHAPTER 4

Methodology

This chapter introduces the methodology used to implement the metrics formulated in Chapter 3. First, the general architecture of the implemented CTI rating framework called Enodo, latin for *analyze* or *unravel*, is explained. Then, the playbook execution via Dagster is described. Finally, the components are described in detail.

4.1 Overview

The current implementation of Guardia has services added at the discretion of software architects. The services must be manually reviewed for relevance to the pipeline. Possible charges for non-free APIs are only considered at their implementation which leads to the problem of developers not being familiar with these APIs and using them as disposable assets without considering their impact on cost, run time length or even necessity for the current pipeline workflow.

4.2 Architecture

Proposed is a system that hooks into the existing pipeline compositor, evaluating the results of executed services according to the metrics described in Section 3.4, persisting them and using an additional recommender engine offering easily digestible graphs to security analysts. Figure 4.1 shows an overview of the architecture, including its interfacing with external endpoints. Notably, the hook is the only component embedded within Dagster and is therefore written in Python, the other components are implemented in Go. Figure 4.2 shows the database schema which is automatically created by the Object–Relational Mapping (ORM) tool contained within the analyzer.

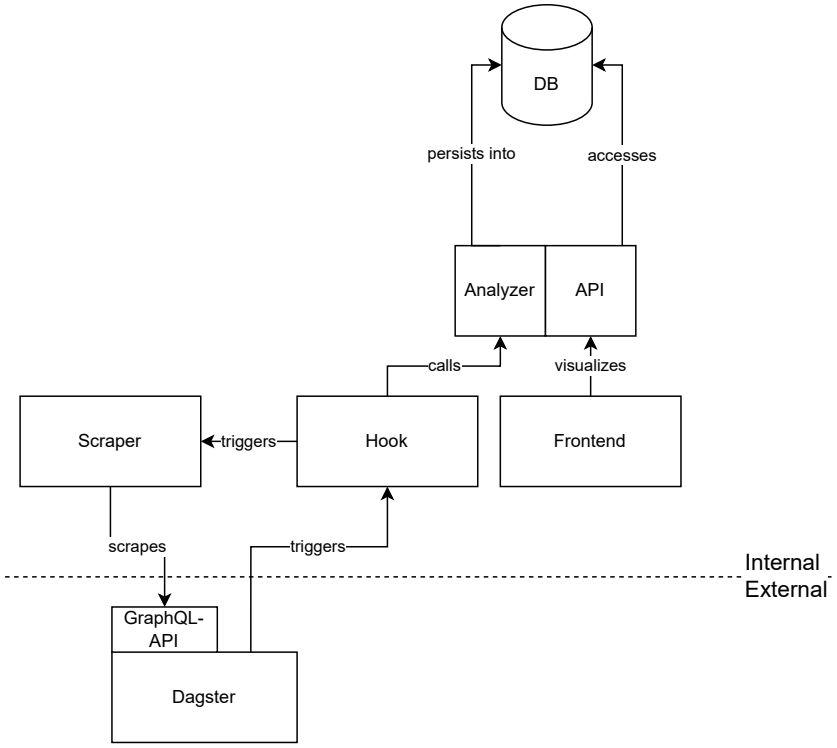


Figure 4.1: The Enodo architecture.

4.3 Playbook Execution in Detail

To understand where the STIX graph comes from and where other metrics are originating from, the example playbook run from Section 2.6 is explained from the technical backend view.

4.3.1 Pipeline Definition

For scheduling and executing playbooks, the Guardia architecture uses Dagster, a platform for data orchestration and workflow execution. Workflows can be defined in code via Python files. These files are read and validated by Dagster at initialisation time and if no errors in the definition are found, the playbooks in form of workflows can be executed by calling API endpoints. These workflows now not only contain the calls to the CTI services but also the parsing of input data, extraction of possible attachments, validation of content and packaging of results into STIX bundles again.

These workflows contain multiple building blocks, called *Ops* by Dagster. These blocks can be designed generic and thus be reused in multiple workflows. They also allow for closed inspection of data in- and output within a workflow and crucially, can be timed from begin of execution to the return of output.

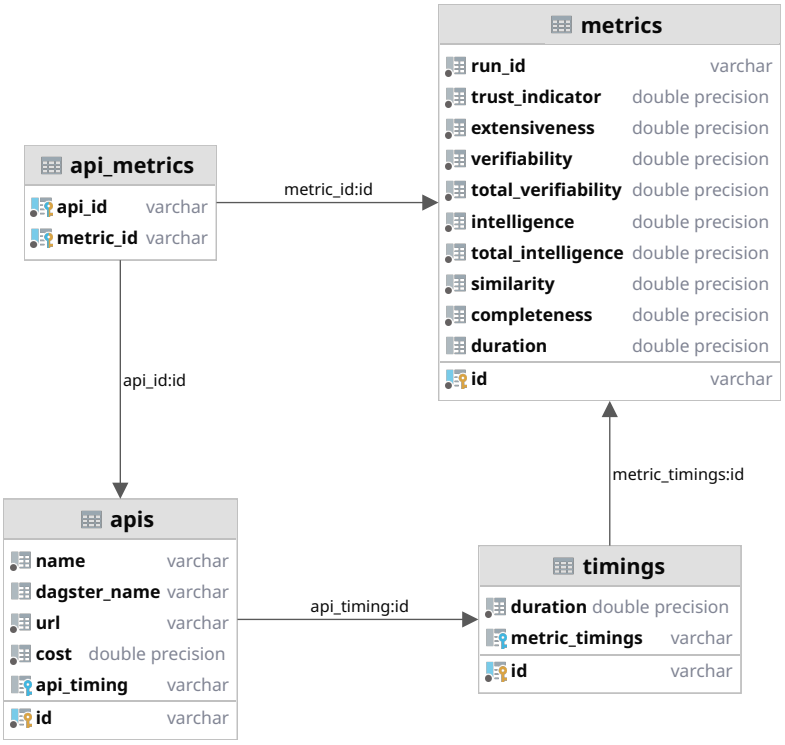


Figure 4.2: The Enodo database structure.

4.3.2 Pipeline Execution

For each playbook, Dagster builds a graph with all ops connected by input and output. Taking the example from Section 2.6, the playbook consists of multiple stages of extraction, enrichment and aggregation. From Python code files, Dagster creates workflows as seen in Figure 4.3. These workflows can then be triggered by a simple API call containing the required inputs defined within the source code of the respective playbooks. Dagster then automatically schedules and executes the Ops.

4.4 Hook

The hook component is the smallest part of the Enodo architecture. Dagster allows for hooks, also called *sensors* within Dagster, to be attached to jobs. These sensors are then automatically triggered whenever the pre-configured conditions are met. The requirements for the metric trigger are simple; whenever a playbook successfully finishes, Enodo should aggregate and analyze various metrics and return either *success* or *failure* to the Dagster frontend. As both scraper and analyzer accept API requests, the hook can call the scraper endpoint first and pipe the result into the analyzer endpoint. If

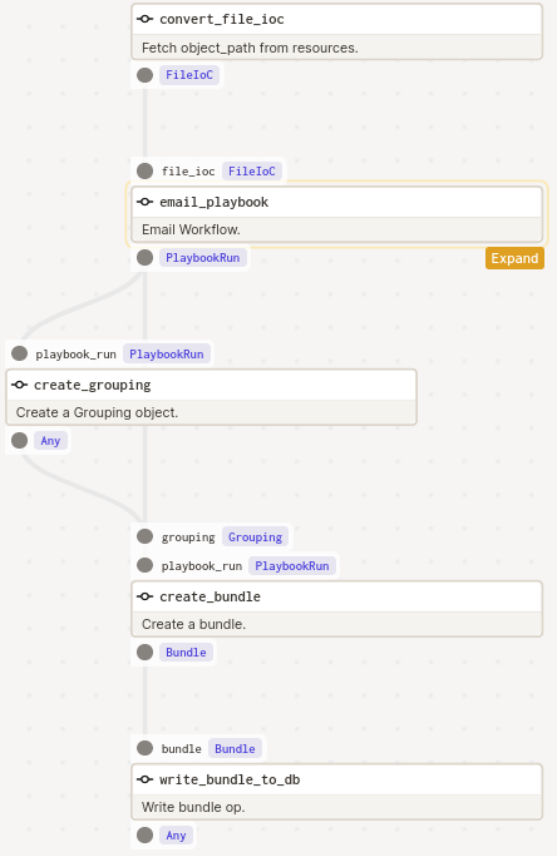


Figure 4.3: Dagster workflow as shown on the Dagster Graphical User Interface (GUI).

that action also succeeds, the analysis was successful and can be shown as such in the frontend, as seen in Figure 4.4.

4.5 Scraper

The scrapers' responsibility is to aggregate the input data needed by the analyzer. Following metrics shown in Table 4.1 need to be evaluated and are therefore the responsibility of the scraper to fetch, with its source listed beside it.

Item	Source
Case ID	Dagster API
Service Timings	Dagster API
STIX Bundle	Guardia API

Table 4.1: Items needed for the analyzer and their sources.

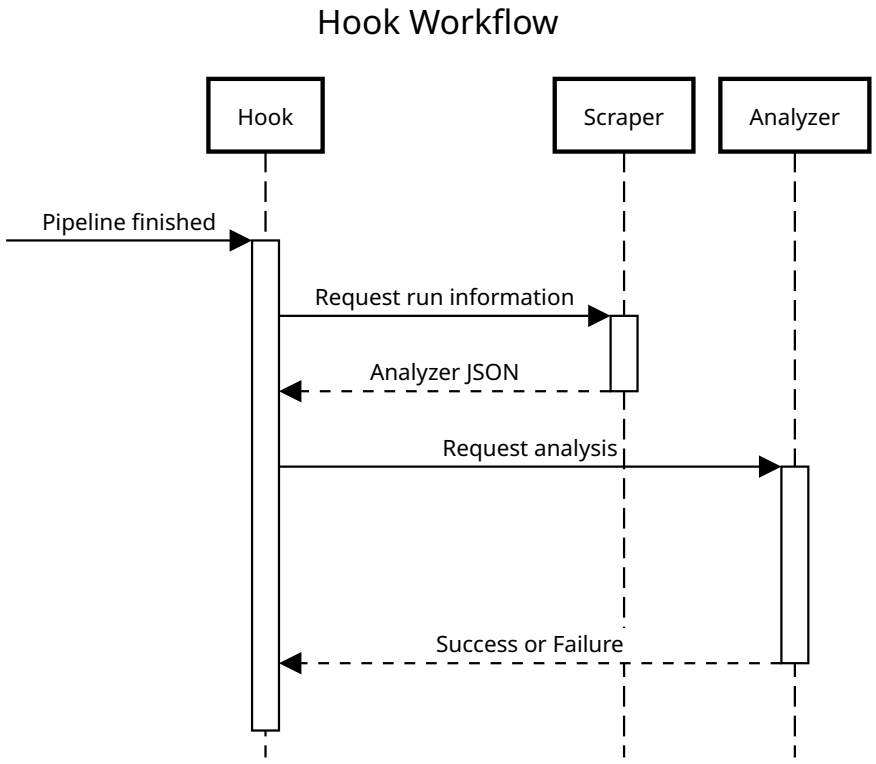


Figure 4.4: Sequence diagram of the hook workflow.

As the `caseId` is not known by the hook, the scraper needs to fetch the Dagster API first before requesting the STIX bundle from the Guardia API. The resulting workflow is shown in Figure 4.5.

4.5.1 Accessing the Scraper

The scraper can be activated by accessing it via the API. It offers a single endpoint at `/scrape` which accepts `POST` requests containing the `runId` of the Dagster workflow run, see Listing 4.1.

```
POST http://enodo-scraper/scrape
Accept: application/json
Content-Type: application/json

{
  "runId": "7fa62179-2182-4b7d-80cf-5b973887e727"
}
```

Listing 4.1: Example `POST` request for the scraper.

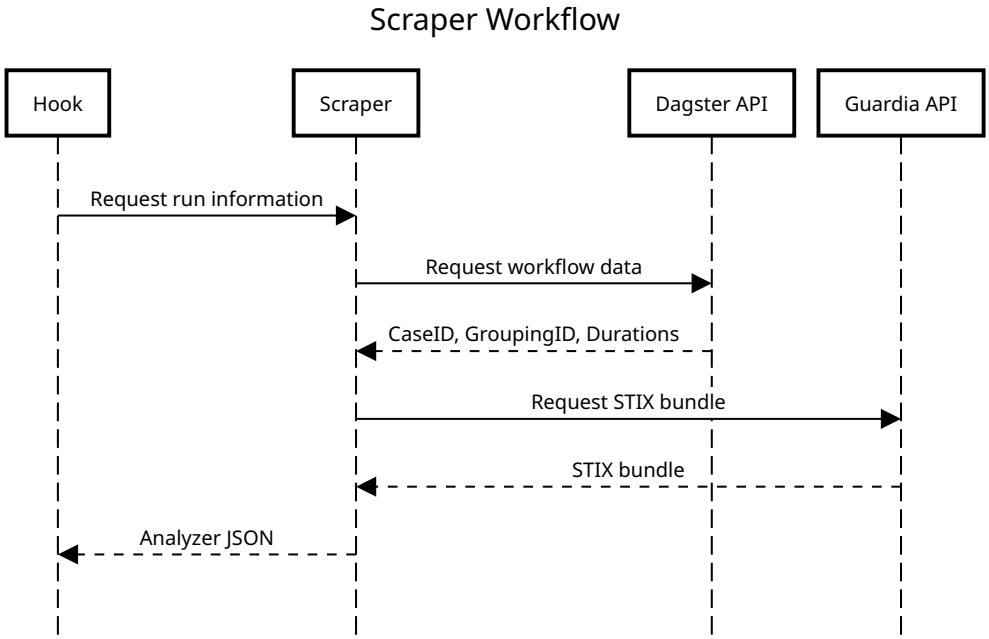


Figure 4.5: Sequence diagram of the scraper workflow.

4.5.2 Scraping the Workflow Metadata

Dagster employs a GraphQL API to serve information about workflows and runs. This allows for a single, selective query to fetch all information needed for the analyzer. One important aspect of the scraper is the matching of the Dagster workflow run with its corresponding `runId` to the resulting STIX bundle with its `caseId`. Additionally, due to the nature of how the STIX bundles are fetched by the Guardia API, the originating *grouping* node id has to be grabbed from the workflow run.

To satisfy these requirements, various features of Dagster need to be used besides its main functionality of executing workflows, namely tags and materializations.

Tags Dagster allows for tagging of workflow jobs. Tagging can happen while a workflow is executed and may be user-defined. Thus, for the scraper to successfully match `runId` with `caseId`, the underlying playbook code must tag the workflow runs with the correct case id used for the STIX bundle.

Materializations Materializations, also called `AssetMaterialization` in Dagster, are entities that are created within workflows but are external to Dagster, such as database insertions, push notifications or the creation of STIX bundles. Materializations consist of a fixed key, identifying the materialization throughout different runs, and a key-value pair map containing additional metatags. This map can then be used to convey information for the scraper by setting an entry with key `service` and value set to the

name of the API being called. This allows the scraper to more efficiently query services by only querying for `AssetMaterializations` within the run and then filtering out materializations without a `service` key set.

After facilitating these changes in the playbook code, tagging of workflow runs and materializing API responses, the scraper is able to fetch all the required information with a single GraphQL call to Dagster. The response then contains a list of all `AssetMaterializations` within the workflow run. If the `AssetMaterialization` contains the metadata entry `service`, the duration of the API response time is calculated by subtracting the start time of the `op` from the end time. As services may be called multiple times in a single workflow, it is necessary to save all duration into a list mapped to the name of the service.

4.5.3 Scraping the STIX Bundle

After the scraper has fetched the corresponding `caseId` from the Dagster endpoint, it can download the completed, deduplicated bundle from a Guardia API endpoint. The endpoint receives the `caseId`, `nodeId` and returns a STIX bundle. This bundle is then copied into the scrapers' response and returned to the hook to be forwarded to the analyser. An example response is demonstrated in Listing A.1.

4.6 Analyzer

Dagster allows for functions to be called after successful or unsuccessful pipeline runs, called hooks. The functional content of these hooks can be defined by the developer; thus they can also call external programs. This functionality can be used to connect to the analyzer. These hooks also allow for additional metadata to be passed such as the job name itself, possible thrown exceptions and the operation output itself [39]. The output is presented in STIX format, which can then be analysed according to the formulated metrics. These metrics are aggregated based on the source. In Guardia, each external source is represented by a report linking to all objects received from the source.

The analyzer implements the algorithms described in Section 3.5. Serving as the focal point of this work, it analyses received STIX bundles by calculating the aforementioned metrics. To reduce redundancy in source metadata and as the historic change of metric values is also important, the calculated metrics are also persisted in a database. The analyzer is also responsible for storing data not directly derived from the analyzer such as metadata from each source, i.e. endpoint Uniform Resource Locator (URL) or cost per call.

4.6.1 Accessing the Parser

The parser can be accessed similarly to how the scraper in Section 4.5.1 is accessed. Differing from the scraper is the input for the endpoint `/analyze`. The JSON response

from the scraper can be directly piped into the request body for the analyzer. An example of a request body can be seen in Listing A.1 in the appendix.

As the analyzer persists all calculated metrics into the database, nothing is returned in the response except a Hypertext Transfer Protocol (HTTP) status code. Possible status codes are 500 if an internal error has occurred, i.e. the database for persisting the metrics is not available, 400 if the request is not valid, i.e. does not conform to the expected JSON structure, or 200 if the request was successful.

4.6.2 Parsing

Parsing the request is done in two steps. First, the JSON body is parsed and split into the three components it contains; the `runId`, the STIX bundle and the service duration map. Secondly, the bundle itself is decoded. This happens with help from an external library called *libstix2* [40]. The library allows for easy parsing of JSON documents into STIX bundles. Some minor changes had to be made to the library to incorporate features required by Guardia. These changes are described in Section 4.6.7. After the parsing is completed, the bundle can then be fed into the analyzer.

4.6.3 STIX Metric Calculation

The analyzers' main purpose is to calculate the formulated metrics in Section 3.5 and persist them and the resulting *Trust Indicator* into the database.

Calculate Extensiveness

The extensiveness calculation for a source is straightforward, as for each object the amount of existing properties is divided by the amount of possible properties. This value is then added to a `sum` variable. This continues in a loop until all objects of a single source are evaluated, then the `sum` variable divided by the number of objects of that source is returned, see Algorithm 4.1.

Algorithm 4.1: Calculation of Extensiveness.

Input: All objects of a single source $\mathbf{O_S}$
Output: Extensiveness $\mathbf{p_1}$

```
for O ∈ O_S do
    ExistingProperties := GetExistingProperties(O);
    PossibleProperties := GetAllProperties(O);
    Sum += length(ExistingProperties) / length(PossibleProperties);
end
p_1 := Sum / length(O_S);
return p_1
```

Calculate Compliance

The compliance calculation was not implemented within this framework. This was due to Guardia currently transforming all API responses from different formats received from the services to STIX. In this process, a validator is used to confirm the implemented transformer is returning valid STIX. In turn, this means Enodo will never receive invalid STIX, as the validator used would be the same validator as in the tests of the transformer.

Calculate Representational Consistency

As with compliance, the transformer will validate any incoming data before persisting it within Guardia, meaning it will never reach Enodo as it only receives data already contained within the system. This limitation could be overcome by passing invalid STIX still on to Enodo. However, this change was out of scope for this thesis.

Calculate Verifiability

Verifiability can be calculated by simply measuring the length of the `External-References` list. The required `AverageReferences` are calculated in an additional function. As the average number of references does not change for a bundle, regardless of which source is currently being evaluated, it can be calculated once in the first call and then cached. When the verifiability is calculated for other sources, this average number value can then be accessed from the cache, reducing the number of calculations done and speeding up the runtime. Refer to Algorithm 4.2 for said implementation.

Algorithm 4.2: Calculation of Verifiability.

Input: All objects of a single source $\mathbf{O_S}$
Output: Verifiability $\mathbf{p_4}$

```
1  Function AverageExternalReferences():
2      if GlobalAverageExternalReferences = Undefined then
3          for O ∈ O_Bundle do
4              TotalReferences += length(O.ExternalReferences);
5          end
6          GlobalAverageExternalReferences := TotalReferences / length(O_Bundle);
7      end
8      return GlobalAverageExternalReferences;
9  ;
10 for O ∈ O_S do
11     Sum += length(O.ExternalReferences)
12 end
13 p_4 := Sum / length(AverageExternalReferences() * length(O_S));
14 return p_4
```

Calculate Intelligence

The metric Intelligence measures how many relationship objects a source provides. The more the provided objects are connected together, the higher the intelligence score. The metric is calculated once for the whole bundle; subsequent requests for a source simply access the saved value for each service.

The method itself iterates over all objects contained within the bundle. Depending on the type, all connections from that object to another object are counted. For example, the *Sighting* object in STIX contains following additional properties referring to other objects: *Sighting of Ref*, *Observed Data Refs*, *Where Sighted Refs*. *Observed Data Refs* and *Where Sighted Refs* are both reference lists; therefore the length of each list is used for the reference count. The final count of each object is then added to its sources' count. This is facilitated by a map, containing the service name as key and the current reference count as value.

After all objects are accounted for, the intelligence score for each source is calculated by taking the number of relationships of a source and dividing it by the number of STIX objects of the source, as seen in Algorithm 4.3.

Algorithm 4.3: Calculation of Intelligence.

```
   Input: Source S
   Output: Intelligence p5
 1 Function IntelligenceMap():
 2     for O ∈ O_Bundle do
 3         begin
 4             switch O.Type do
 5                 case Relationship do
 6                     Connections[O.SourceRef] = Connections[O.SourceRef] + 1;
 7                 end
 8                 ...
 9             end
10         end
11     end
12     for S, ConnectionSum ∈ Connections do
13         Intelligence[S] = ConnectionSum / length(ObjectsOfSource(S));
14     end
15     return Intelligence;
16 p5 := IntelligenceMap[S];
17 return p5
```

Calculate Similarity

Similarity measures how many objects from other sources are similar to an object of the current source. The metric is implemented differently than described in the paper [37] in that text similarity algorithms are not used on the whole object, but instead on each property where fitting. This allows for more fine-grained control of how properties of different objects are handled. For example, some properties may contain lists. When simply comparing two lists containing the same contents, but in a different order, the similarity index may vary drastically. However, the order of two lists should not decide their similarity, thus lists must be checked for similarity by checking if each item of one list is contained within the other list.

Calculating Similarity for all objects To calculate the similarity of a source, for each object, a similarity map is created containing the similarity of the current object to the object referenced by the key of the map. The sum of all similarities is then divided by the number of objects provided by the source as seen in Algorithm 4.4. This value represents the similarity metric, the average similarity of objects by the source compared to the worldview. The similarity map calculation is described in the following paragraph.

Algorithm 4.4: Calculation of Similarity.

Input: Source **S**, Weight Map **WeightMap**
Output: Similarity $\mathbf{p_6}$

```
1 for O ∈ O_S do
2 |   Similarities += Avg(GetSimilarityMap(O));
3 end
4 p_6 := Similarities/length(O_S);
5 return p_6
```

Calculating the similarity map Algorithm 4.5 compares the current Object A with all objects within the worldview. It first filters all objects which are not the same type as A, as they cannot be similar. It also only compares A against objects from other sources. If there are objects which are of the same type and not from the same source, they are compared by the function `GetObjectSimilarity()` and the result is saved into a map. This map is then returned as the similarity map.

Calculating Similarity between two objects Algorithm A.1 in the appendix compares two objects; object A, created by the current source, and object B, taken from the worldview and created by another source. When comparing two SDOs, the existing properties of both objects are discerned. Then, the properties of A are iterated over. First, B is checked if the property even exists there. If not, there is nothing to compare the similarity to and the next property of A can be checked.

Algorithm 4.5: Calculation of the similarity map.

```
1 Function GetSimilarityMap(O_This):
2     for O_Other ∈ O_Bundle do
3         if O_Other.Type ≠ O_This.Type||O_Other ∈ S then
4             Skip;
5         end
6         SimilarityMap[O_Other] = GetObjectSimilarity(O_This, O_Other);
7     end
8     return SimilarityMap;
```

If B also contains the property of A, the property key acts as a switch between different calculations of the similarity. As different keys have differently typed values contained in them, the key is used to discern which similarity calculation must be used on the current property. This key-based switch is mostly used for atypical keys; for example `External-References` is a list containing multiple `ExternalReference` entries, which are JSON structures in-memory. This is unique for this key and thus a dedicated function is implemented, comparing the values of both objects A and B.

However, most property values are lists of atomic types, numbers, strings or other atomic types. If the property is of atomic value, both values of A and B can simply be compared via an appropriate method. For string properties, the Jaccard [38] index is used, for numbers, Equation (4.1) is applied.

$$Similarity_{Number} = 1 - \frac{Max(A,B) - Min(A,B)}{Max(A,B)} \tag{4.1}$$

For slices, while they may contain a multitude of STIX types, they are all represented as `string` within the code. A notable exclusion is `kill_chain_phases`, this case is handled by the key-based switch mentioned above. For lists, each element contained within the list of object A is compared with every object in the list of the object B via the Jaccard index. Notably, for lists containing references to other objects, the id part is removed from the string. For example, `identity--2c4be6e6-2c99-443f-9ab1-24ecedb026d5` gets shortened to `identity`. This is due to the fact that different sources may reference different objects even though they contain the same information. By removing the unique part of the reference, two objects pointing to the same type but different objects will still yield a high similarity value.

After all similarity values have been calculated, an optional weighting factor for each is applied; with some objects, a single property may indicate similarity much more than other properties of the same object. For example, for each object, the `modified` property is also compared; however, this value may differ between two sources, even though the other content matches to a high degree. If no weight is set, a default weight of 1 is applied.

Finally, the sum of all similarity values is calculated and divided by the number of properties shared between both objects, as can be seen in Algorithm 4.6.

Algorithm 4.6: Calculation of object similarity.

```
1 Function GetObjectSimilarity(O_This, O_Other):
2     SimilarityMap := CompareProperties(O_This, O_Other);
3     for Property ∈ SimilarityMap do
4         SimilarityMap[Property.Name] = SimilarityMap[Property.Name] *
           WeightMap[Property.Name];
5         Divisor += WeightMap[Property.Name];
6     end
7     Similarity := Sum(SimilarityMap.Values)/Divisor;
8     return Similarity;
```

Calculate Completeness

The completeness calculation, see Algorithm 4.7, uses the similarity map calculation described in Algorithm 4.5. For each object of a source, this similarity map is calculated. Then, the highest similarity within the map is compared with a previously set threshold; if the similarity is higher, it is assumed that both objects contain the same information, albeit from different sources. In the end, the number of similar objects is divided by the total number of objects originating from other sources.

Algorithm 4.7: Calculation of Completeness.

```
   Input: All objects of a single source O_S
   Output: Completeness p_7
1  for O ∈ O_S do
2      SimilarObjectsValues := GetSimilarityMap(O);
3      MostSimilarObjectValue := Max(SimilarObjects);
4      if MostSimilarObjectValue > SimilarityThreshold then
5          SimilarityCount = SimilarityCount + 1;
6      end
7  end
8  p_7 := SimilarityCount / (length(O_Bundle) - length(O_S));
9  return p_7
```

Calculate Indicators

As all metrics have the same input and output, a generic interface is introduced which metric implementations need to follow. This eases portability and reusability, as this interface allows for the simple addition of further metrics without changing the existing code much; the metric simply needs to be added to the list of metric functions and

have its weight added to the *Trust Indicator* calculation. Algorithm 4.8 describes the calculation of each metric value.

Algorithm 4.8: Calculation of an indicator.

Input: Source name **S**
Output: Map of metric values **I**
Data: *metrics* is a list of metric objects with name and the corresponding function set.

```
1 O_S := GetObjectsOfSource(S);
2 for metric ∈ metrics do
3 |  I[metric.name] = metric.function(O_S);
4 end
5 return I
```

Calculate Duration

The calculation of the duration metric is also done by the analyzer. The durations are not contained within the STIX bundle; instead, they are received from the scraper as a list of durations. As seen in Algorithm 4.9, the average of the list is calculated and inserted into the formula describing the duration metric.

Algorithm 4.9: Calculation of Duration.

Input: List of duration values **D**
Output: Duration $\mathbf{p_8}$

```
1 average := Avg(D);
2 p_8 := Exp(Decline * average);
3 return p_8
```

4.6.4 Trust Indicator

Depending on the customers' preferences, weights can be used to prioritize certain metrics when calculating the final *Trust Indicator* score. Each indicator is assigned a weight between 0 and 1, with the requirement that the sum of all weights equals 1. Finally, the sum of all weighted indicators is returned as the trust indicator.

The original formula proposed by Schaberreiter *et al.* [37] includes two variables which are unused in the calculation of the trust indicator within Guardia; the ageing factor D and the previous trust indicator $TI_{s_x}(t-1)$. Both of these variables take the historic trust indicator for the source into consideration. However, this is not applicable for the use case of Guardia, as the trust indicator is calculated for different inputs, thus the possibility of it varying greatly is given. For example, if a source lacks CTI for the banking sector, a following run concerning the aviation sector should not be affected by it. If the previous value of the trust indicator is used, the rating for the current run

may be affected in incomprehensible ways. Thus, the formula used to calculate the trust indicator within Guardia is described in Equation (4.2).

$$TI_{s_x}(t) = \frac{\sum_{n=1}^{m} \omega_n * (p_n)_{s_x}(t)}{\sum_{n=1}^{m} \omega_n} \tag{4.2}$$

4.6.5 Duration Mapping

As mentioned in Section 4.6.2, besides the STIX bundle and the run id, the request also contains a service duration map, containing the run times of all Dagster ops which contact external service APIs. A problem faced here is the inconsistency between the name of the service in Dagster and the created identity for a service within STIX. For example, in the Python code for Dagster, the ip-api service may be called `IpApi` while its identity in STIX is called `ip-api.com`. To be able to correctly attribute durations to services, an additional row within the `metrics` table is added with the corresponding name of the service within Dagster.

4.6.6 Database Persistence

After the metrics have been calculated, they are persisted in a database. This is accomplished by the *Analyzer* component with the help of an ORM tool.

For each service, it is checked if it already has an entry in the database. If it has, this entry will be referenced in the metrics to be inserted. Else, the service will be created in the database, with some rows left empty such as `cost` or `url`. These values are not contained within the STIX bundle and must be entered by the security analyst or software architect.

Finally, the durations and metrics are persisted by creating an ORM metric object and assigning it the values calculated by the analyzer.

4.6.7 Library Additions

The recommender extensively uses the Go library `freetaxi/libstix2` [40]. While it provides extensive base functionality, some areas vital for Guardia are lacking; custom object decoding and SCOs.

Custom object decoding in bundles is needed by Guardia as some new types of objects have been introduced by Nextpart which are not part of the default objects part of the standard. Initially, *libstix2* only decoded pre-defined objects when decoding a bundle. An additional method allowing the passing of custom object decoders was added.

SCO objects were lacking in the library; this is due to the fact that they are mostly unused in other STIX usages, such as Trusted Automated Exchange of Intelligence Information (TAXII) services [41]. However, SOAR platforms make extensive use of SCO objects, as most input data is described as SCOs, such as email messages or domain names.

Thus, a variety of SCO objects is introduced into the library to allow for the correct representation of input data within Guardia.

4.7 API

As the database table itself is not useful to either the user nor the pipeline architect, an additional service called *Recommender* is added. The API is implemented with GraphQL and included within the analyzer as it already has a connection to the database, although eventually to avoid bottlenecks, the API component may be implemented as an independent service. The used ORM tool furthermore provides support for GraphQL queries, thus the inclusion of an API is easily maintainable. The API includes a history of the calculated trust indicator but also includes additional metadata for each source such as API cost, company or endpoint. This allows the architect to further enhance their knowledge of the different APIs. Also, having an overview of the different sources allows for direct comparison between competing sources and recognition of up-and-coming sources.

4.8 Frontend

To ease development and to visualise the metrics in a comprehensible way, a frontend is implemented. This frontend accesses the endpoint provided by the analyzer. Thanks to a dropdown, runs can be filtered by their case id.

This frontend is written in Vite and Svelte, which is also used in the Guardia framework. Using the same technologies enables the possibility of easily merging the external Enodo frontend into the Guardia frontend for the customers to use. In its current iteration, the GUI is aimed at debugging and reviewing changes to the metric algorithms. It can also be used by security analysts to compare multiple services against each other. However, to be suitable for customers, the metrics must be integrated into the frontend and further explained for customers to make sense of them.

Figure 4.6 shows the frontend as seen from within a browser. Below the logo and title of the page a dropdown menu is visible. This menu can be used to select a case, in this example named *Example*. When a case is selected, one or more services will be listed below the dropdown. For each service, the name and id are presented. Furthermore, all metrics of the service are shown graphically via bars. Some metrics, such as Verifiability or Intelligence have a number beside that bar; this number represents the absolute value of that metric, while the bar represents the normalised value.

Figure 4.6: The Enodo frontend.

CHAPTER 5

Results

This chapter describes the test setup for the Enodo framework. All components and their setup are introduced and the test samples are described. The resulting metrics are shown and will be discussed in Section 6.1.

5.1 Setup

The integrated Enodo framework is tested while integrated into Guardia. This allows for a review closest to the real-world application of the framework. The system specifications of the system are shown in Table 5.1.

Operating System	Fedora 36
Container Management	Docker 20.10.17
Kernel Version	5.18.13
System Type	x64
Processor	Intel i7-8665U
RAM	32GB

Table 5.1: System specifications of the test system.

For the calculation of Similarity, some properties are assigned custom weights; these are empirically chosen and try to minimize the impact of the timestamps `created` and `modified` when comparing objects from two different sources. Additionally, `custom` properties are ignored as these need special cases handling them and no such comparator is implemented. The properties `integration` and `raw_report` also weighted lower than 1.0 as they might differ significantly between two objects from different sources while the content refers to the same concept. The property `classification` is weighted

with 2.0 as this property is an important aspect of the object type `report` in terms of similarity. The full weight map can be seen in Table 5.2.

Type	Weight
created	0.2
modified	0.2
custom	0.0
integration	0.5
raw_report	0.2
classification	2.0

Table 5.2: Weight map for the similarity calculation.

Furthermore, all tests were executed from the internal network of the University of Applied Sciences Upper Austria, Campus Hagenberg, located in Hagenberg im Mühlkreis. The provided network speed at the time of writing, August 2022, is 710Mbps Down– and Upload. Guardia is deployed via Docker and runs on the single instance noted in Table 5.1.

5.2 Report Example

The STIX 2.1 standard includes an example report [18], seen in Listing A.2 in the appendix. This example report contains just 5 objects; 1× `report`, 1× `identity`, 1× `indicator`, 1× `campaign` and 1× `relationship`. However, Enodo is still able to analyze this package, as seen in Figure 5.1.

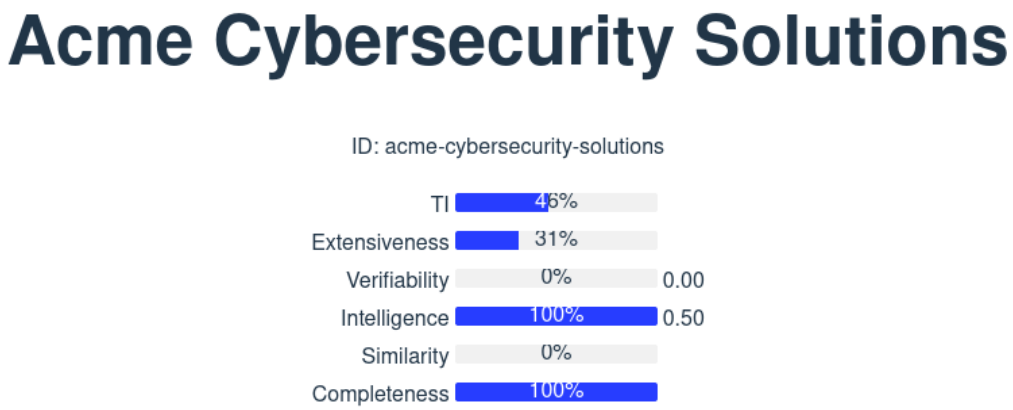


Figure 5.1: Metrics of the report bundle provided by the STIX standard.

Extensiveness The report shows low extensiveness with a value of 0.31.

The `indicator` object contains 4 type-specific properties: `name`, `indicator_types`, `pattern` and `valid_from`. 9 properties are possible for the `indicator` type, thus the extensiveness value of this object is 0.44. The `campaign` object has 1 type-specific property set: `name`, of possible 6 properties, resulting in a extensiveness value of 0.17.

As the other two objects within the bundle, `identity` and `report` are not contained within the report itself, they are not counted against the extensiveness. Thus, the extensiveness metric for the source is $\frac{0.44+0.17}{2} = 0.31$.

Verifiability No object contained within the bundle references any external sources, thus the verifiability metric for the source is 0.0.

Intelligence The value of the non-normalised intelligence metric is 0.5, as the report contains 3 objects; two SDOs and 1 SRO. Thus, dividing the number of relationships with the number of objects yields $\frac{1}{2} = 0.5$. Normalisation happens by taking the highest metric of a source and dividing by it; thus $\frac{0.5}{0.5} = 1.0$.

Similarity As no other sources are contained within the bundle, the similarity is 0.0 as no other objects are available to compare against.

Completeness The completeness takes into account how much of the world view is provided by the source. As no other sources contribute to the world view, the completeness of the source is rated at 1.0.

Duration As this example bundle was not executed within the Guardia framework, no duration is available to be measured, therefore it is omited from metric overview and also not considered in the trust indicator calculation.

5.3 E-Mail File

An email message with textual content, as seen in Listing 5.1, is uploaded as a file in the interface. The email itself was sent from *konstantin@papesh.at* to *konstantin.papesh@fh-hagenberg.at*.

```
www.orf.at
www.dietagespresse.com

Have fun!
```

Listing 5.1: Content of the email used for testing.

5.3.1 Single Service

In this test case, only a single service is selected to enrich the input. After the playbook has been run, a report containing 137 objects, from which 66 were relationships, gets generated. A distribution of all objects contained within the report can be seen in Figure 5.2. For this report, Enodo calculated the metrics seen in Figure 5.3.

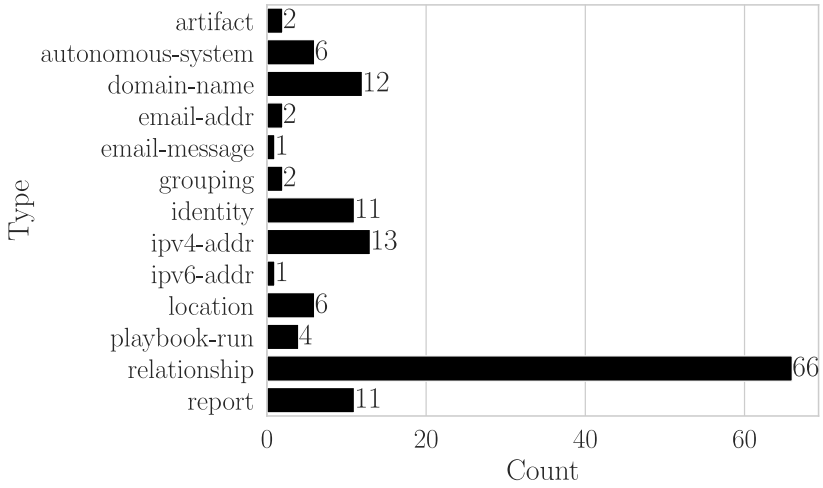


Figure 5.2: Types contained within the resulting STIX bundle.

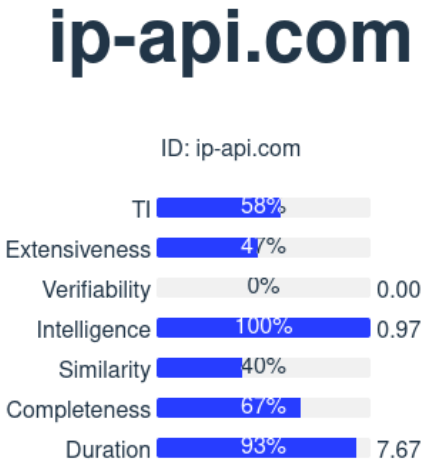


Figure 5.3: Metrics for the email workflow with a single service enabled.

Extensiveness The extensiveness metric yields a value of 0.47, so less than half the possible attributes for objects from the source are filled. For example, thirteen `ipv4-addr` objects are contained within the bundle. But for each object, only the

`value` parameter is set, leaving the `resolves_to_refs` and `belongs_to_refs` parameters empty. Also the contained `domain-name` objects only contain `value`, leaving them with an extensiveness of 0.5 as the `resolves_to_refs` parameter is left empty.

Verifiability The verifiability metric is 0 for the source. This is due to the fact that the source does not reference any external sources in its objects. No `external_references` parameter was set in any object.

Intelligence Intelligence is measured by the number of relationships compared to the number of SDOs from the source. The intelligence metric itself is 1.0, this is because no other source was available to normalise to. Thus, the source has the highest intelligence rating in the world view, resulting in a normalised value of 1.0. The true, non-normalised intelligence metric is 0.97. Thus, the object has slightly fewer relationships than SDOs. In the STIX bundle, there are 31 relationships attributed to the source, while 32 STIX objects are attributed to the source, thus resulting in an intelligence score of 0.97.

Similarity The similarity metric is 0.4, which means that objects from the source resemble objects from other sources on an average of about 40%. While only a single service was used to analyze the input data, Guardia also inserts some STIX objects into the bundle; these are extracted from the input data. While these objects do not show up in the source metric listings, they are still compared against the metrics of services. In this case, Guardia adds some *Domain Name* objects to the bundle. The similarity between domain names is often very high as only some letters may differ. For example, the bundle includes two domain names; `mx-gate04-conova.antispameurope.com` and `mx-gate06-conova.antispameurope.com`. Both only differ by a single digit, 4 and 6. The source contributes the 06 domain name, resulting in a high similarity value.

Completeness Similar to *Similarity*, the completeness metric also takes into account objects created by Guardia itself. As *ip-api.com* already contributes 32 objects to the bundle, with 49 objects counted, the completeness yields a value of 0.65. As some objects are very similar and thus result in a similarity higher than the set threshold for completeness, they are also counted to the completeness for the source, resulting in a final completeness of 0.67.

Duration The run time duration of the service is an average of 7.67 seconds. Converting it to a metric via the formula described in Section 3.5 yields a value of 0.93.

Trust Indicator The final *TI* metric for the source *ip-api.com* yields a value of 0.57. This value is aggregated from all other metric values, without any weighting done.

5.3.2 Multiple Services

This test uses the same email as the test with the single service. However, a multitude of services has been selected to enrich the input data.

The final STIX bundle of the workflow contains 490 objects, with 234 relationships. 60 identities are contained and 58 reports were created by them. An overview of all objects can be seen in Figure 5.4.

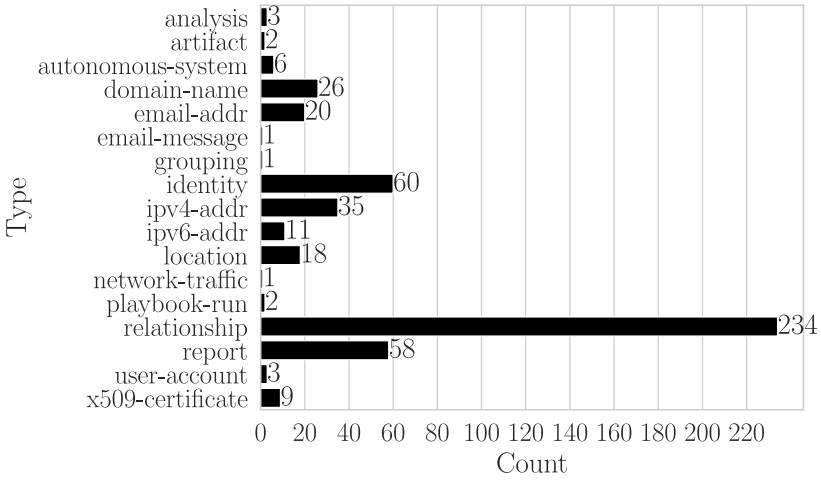


Figure 5.4: Types contained within the resulting STIX bundle.

For brevity, the metrics are transformed into a table as seen in Table 5.3. The results as displayed on the webpage are shown in Figure A.1. Notably, some custom objects

Name	TI	Ext.	Ver.	Int.	Sim.	Comp.	Dur.
EmailRep	0.49	0.25	0.00	1.00	0.73	0.01	0.93
Maltiverse	0.48	0.37	0.00	0.38	0.75	0.43	0.94
NetworksDb	0.47	0.40	0.00	0.64	0.68	0.12	0.96
VirusTotal	0.46	0.41	0.00	0.23	0.51	0.66	0.95
ip-api.com	0.46	0.50	0.00	0.37	0.55	0.37	0.96
Nextpart	0.43	0.34	0.00	0.27	0.61	0.93	-
Shodan	0.43	0.50	0.00	0.57	0.48	0.08	0.97
SocialScan	0.27	0.20	0.00	0.17	0.29	0.04	0.92

Table 5.3: Metrics for the email workflow with all services enabled.

are contained within the bundle; namely `analysis` and `playbook-run`. These two types are introduced by Nextpart and their implementation must be done manually, i.e. what properties they contain. Else, unknown object types will fall back onto the common properties defined by the STIX standard. This will reduce the accuracy of the metrics. All custom objects received within this bundle are implemented, thus no loss of accuracy is observed.

Extensiveness Extensiveness varies greatly between services, with a range between 0.2 and 0.5. SocialScan receives the lowest extensiveness score with 0.2. Upon closer inspection of the objects attributed to the source, the low score becomes clear; the service returns two types of objects; `domain-name` and `user-account`. The `domain-name` objects only provide the `Value` property, with the `ResolvesToRefs` property unset; resulting in an extensiveness value of 0.5. The `user-account` objects properties are completely empty, resulting in a score of 0. So the final score of SocialScan can be calculated by summation of all values and by taking the average. 5 objects are returned; 2 domain names and 3 user accounts, thus, the extensiveness metric of SocialScan can be calculated: $\frac{0.5+0.5}{5} = 0.2$. This is also the score calculated by Enodo.

Verifiability The verifiability metric is 0 for all services; no `external_references` are set by any source.

Intelligence Values of the intelligence parameter vary between 0.17 and 1.0. Again, SocialScan returns the least intelligence with a score of 0.17, while EmailRep returns an intelligence score of 1.0. The non-normalised scores as seen in Figure A.1 are 3.5 for EmailRep and 0.6 for SocialScan.

Similarity The services are quite similar to each other, with most services having a similarity between 0.48 and 0.75, with only SocialScan having a severely lower score of 0.29.

Completeness Approximately 9 in 10 objects are contributed by internal Nextpart services, with the Nextpart service having a score of 0.93. The contributions of other services vary between 0.01 and 0.66, with EmailRep receiving the lowest completeness score of 0.01, meaning that while the Intelligence included in its contributions is rated the best of all services, it only contributes a small fraction to the final result.

Duration All services respond in a similar duration, with ratings between 0.92 and 0.97.

Trust Indicator All services receive a similar trust indicator between 0.43 and 0.49, with only SocialScan, due to its low intelligence and completeness score, receiving a rating of 0.27.

5.4 Malicious Hash

Guardia allows the input of file hashes into the GUI. This file hash is then analyzed by multiple services. The used file hash is `ff20333d38f7affbfde5b85d704ee20cd6-0b519cb57c70e0cf5ac1f65acf91a6`, retrieved from the *MalwareBazaar* database.

Two services are able to work with file hashes, VirusTotal and MalwareBazaar, as seen in Figure 5.6. The STIX bundle contains 211 objects, 95 of which are relationships, see Figure 5.5. Furthermore, 83 objects are of type `identity`. These identities are created by VirusTotal and represent different engines used by it to analyze files.

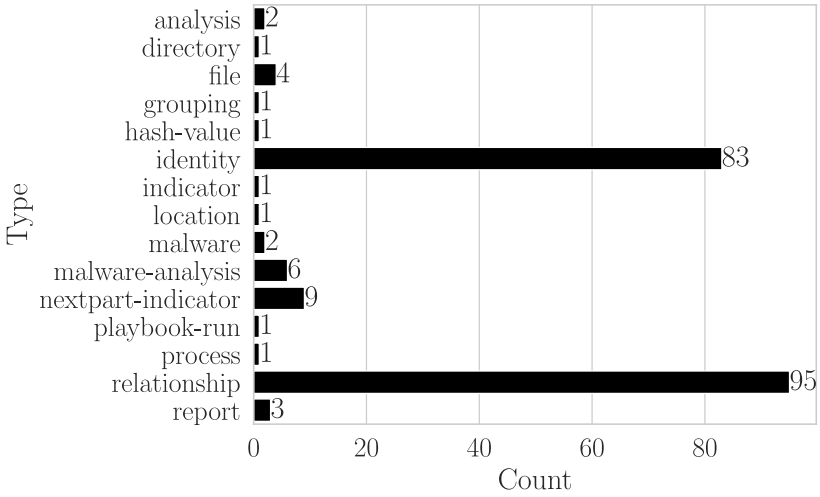


Figure 5.5: Types contained within the resulting STIX bundle.

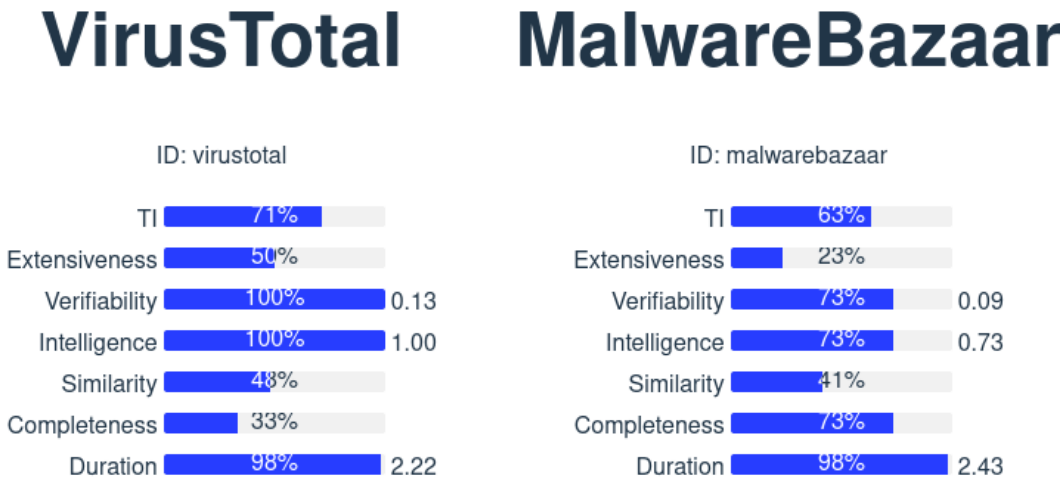


Figure 5.6: Metrics for the hash workflow with all services enabled.

Extensiveness Both services receive scores no higher than 0.5, with MalwareBazaar having less than 1/4 of all possible properties filled on average, resulting in a score of 0.23. VirusTotal fares slightly better with half of the properties filled, yielding a score of 0.5.

Verifiability As the input data is pointing to a malicious file, the services are able to reference external websites containing information about the malware referenced by the hash. MalwareBazaar receives a verifiability rating of 0.73 with 2 external references set in 22 objects, thus $\frac{2}{22} = 0.09$. VirusTotal contains just a single reference, with 8 objects attributed to the source, thus $\frac{1}{8} = 0.125$. As the parameter is normalised, VirusTotal receives a verifiability score of 1.0, with MalwareBazaar receiving $\frac{0.09}{0.125} = 0.72$.

Intelligence VirusTotal provides 8 relationships for 8 SDOs, thus receiving a score of 1.0. MalwareBazaar only includes 16 relationships within its response while serving 22 SDOs, thus a metric score of 0.73 is given.

Similarity The similarity between both services ranges between 0.41 and 0.48. This is due to the fact that both reference the same malicious hash, thus some returned objects contain the same information.

Completeness In terms of completeness, MalwareBazaar contributes more objects to the final STIX graph with a score of 0.73, while VirusTotal only contributes 0.33.

Duration Both services take the same time to respond, thus the duration score is 0.98 for both services.

Trust Indicator VirusTotal receives a higher score in most metrics except completeness, thus its trust indicator is also higher with a value of 0.71. MalwareBaraar, while contributing more to the final STIX graph, is lacking in the other metrics, thus a score of 0.63 is assigned.

CHAPTER 6

Discussion & Outlook

6.1 Discussion

The Enodo framework is able to rate input data with different characteristics, including workflows with only a single service, workflows with multiple services and workflows analyzing malicious data.

Workflows containing only a single service receive the least amount of usable metrics. As some metrics are normalized and some depend on multiple services within the worldview, they have reduced usefulness if used to rate a single service. *Verifiability* and *Intelligence* both are normalized and thus yield 0.0 or 1.0 as their value. As a workaround, their absolute values could be taken into account to determine their quality. *Similarity* and *Completeness* also depend on the worldview but are by nature normalized. Thus, the only useful metrics in connection with single service workflow runs are *Extensiveness* and *Duration*.

Workflows containing non-malicious data are able to be rated on all metrics presented, however, the verifiability metric suffers from a lack of external references; as no Common Vulnerabilities and Exposures (CVE)s or virus databases are able to be referenced, all services receive a score of 0.0, thus reducing the trust indicator. One possibility to mitigate this problem is to drop *Verifiability* from the trust indicator calculation if no service lists any external references. While this does not alter the ranking of services, it normalizes the global average trust indicator rating of a service when observing it in the long run as fewer fluctuations will take place.

Workflows with malicious data receive the most meaningful metrics, with all scores having non-zero values. The example in Section 5.4 shows that services return external references when malicious data is fed. Having external references set leads to an increased verifiability metric, resulting in the *Malicious Hash* run having the highest trust indicators for services of all runs.

It can also be noted that no service returns the full spectrum of object properties offered by STIX, with most services having an extensiveness between 0.25 and 0.5.

Another note is the difference of *Completeness* between services. Taking Section 5.3.2 as an example, the highest ranked service by the trust indicator, *EmailRep*, only has a completeness of 0.01, thus its impact on the worldview is minor. Its similarity is also quite high at 0.73. So it may be possible that the information returned by the *EmailRep* service is already contained within the responses of the other services, and the *EmailRep* service can be omitted from the playbook.

On the other hand, *SocialScan* received the lowest trust indicator with 0.27. The service is neither extensive nor has a high intelligence count. Its completeness value is set at 0.04, meaning it has little impact on the worldview. Yet, it has the lowest similarity with all other services with its similarity value at 0.29. This suggests that the service introduces information that is not yet supplied by other services, however, the quality of said information is lacking. A security analyst may take a look at the objects of the service and determine if they are extensive enough to be kept or if the service is superfluous in the current playbook. Security analysts must always consider each metric individually before recommending a service, as some may yield good values in some metrics, but pose a negligible impact on the worldview.

One noticeable drawback of some metrics, namely *Verifiability* and *Intelligence*, is their metric dependence on the worldview. As the metrics are normalized against the maximum of the current run, the same response from a source will be rated differently depending on the responses of other services. A non-normalized intelligence score of 0.75 will yield a normalized score of 0.8 in one run and 0.55 in another one as other services yield different intelligence metrics. This issue could be mitigated by using a fixed normalization factor which does not change between runs, for example defining a maximum number of external references useful for the security analyst and any service above that maximum number will receive an intelligence metric of 1.0, no matter the number of external references. This fixed number will replace the currently used normalization of the maximum average number of references, thus decoupling the intelligence metric for a single service from the other services.

6.2 Outlook

In Section 1.2, the research question *How can data quality be measured within Security Orchestration, Automation, and Response platforms?* and following sub-questions have been established:

1. What CTI measurements do already exist?
2. How can these be altered to work with CTI data of a SOAR platform?
3. How can a CTI rating framework be incorporated into a SOAR platform?

Chapter 3 shows that a variety of metrics already exist within the literature. However, not all are suitable for SOAR platforms, with some totally unviable for use while others must be altered for use within SOARs. Alterations must also be made to SOAR platforms as well as some metrics are requiring a deeper knowledge of sources than is conveyed solely by the STIX response. Multiple metrics also rely on observation of the source with two or more data points captured.

Alterations made to the metrics concern clarifications of formulas, adding alternative normalization and adding SCOs to the supported STIX types of the metric. On the technical side, the proposed textual comparison [37] is replaced by a more granular comparison which compares objects by each property.

The integration of a CTI rating framework into a SOAR platform does not pose major complications; thanks to a universal standard like STIX, CTI can easily be rated without knowledge of the platform behind it as just STIX needs to be transferred between both implementations. However, some metrics require metadata about services and run times to work; these metadata values are not part of the STIX standard and must be transferred via other means such as JSON. Additionally, these metrics must be aggregated from within the SOAR platform.

As demonstrated in Chapter 5, the metrics are able to rate services in depth. For a single service, different metric values can vary by a wide margin, exposing weaknesses or strengths of a service. A single metric can also be compared against the metric of another service, allowing one to discern higher quality services within a playbook.

Further research directions are additional metrics, especially in the direction of the Schlette *et al.* [35] metric *Appropriate amount of data*. Developing a metric that not only indicates if there is too little but also if there is too much data is in line with the purpose of the rating framework; lowering the information overload on security analysts.

There is also the possibility of adding more metrics to the trust indicator score, further enhancing its capability of determining data quality. Another direction of research is the implementation of a framework concerned with automating the exclusion of irrelevant services from a playbook. Instead of relying on a security analyst to recommend or remove services from a playbook, the metrics in cooperation with a validation component could be used to remove services dynamically.

The metrics currently implemented in the Enodo framework can also be further improved by user feedback. The usage of these metrics in production may uncover flaws or oversights in metrics that are not inherently visible. Long-term usage may also result in the discovery of factors not yet taken into account by the existing metrics.

As mentioned, multiple metrics require historic data. While this is not feasible for the Enodo framework, another component could be implemented to also cover these metrics.

Some metrics could also be improved by feeding it more metadata. For example, Verifiability currently only takes the length of external references into account. An improved

version could also check for the validity of these references, for example by checking if the external reference exists and the URL is reachable.

In conclusion, the implemented framework provides a starting point for managing data quality within SOAR platforms. However, more research is needed to further improve the analysis of services and eventually completely automate playbook composition and service recommendation.

APPENDIX A

Technical Details


```
{
  "runID": "222871ba",
  "bundle": {
    "type": "bundle",
    "id": "bundle--f65107e1",
    "objects": [
      {
        "created": "2022-07-05T14:35:39.062930Z",
        "modified": "2022-07-05T14:35:39.062930Z",
        "created_by_ref": "identity--f65107e1",
        "spec_version": "2.1",
        "type": "grouping",
        "id": "grouping--f65107e1",
        "name": "New Case",
        "context": "unspecified",
        "object_refs": [
          "playbook-run--3b5ca1bf"
        ]
      },
      ...,
      {
        "created": "2022-07-05T14:36:07.467793Z",
        "modified": "2022-07-05T14:36:11.696547Z",
        "spec_version": "2.1",
        "type": "relationship",
        "id": "relationship--481763f5",
        "relationship_type": "located-at",
        "source_ref": "identity--2df176ed",
```

```
          "target_ref": "location--f00799b0"
        }
      ]
    },
    "services": {
      "ShovelApi": {
        "durations": [
          3.083839178085327,
          4.069600582122803,
          3.5667531490325928,
          7.970273971557617
        ]
      },
      ...,
      "WhoisApi": {
        "durations": [
          5.3900086879730225,
          5.079076766967773,
          4.930721044540405,
          4.130650043487549
        ]
      }
    }
}
```


Listing A.1: Analyzer JSON response from the scraper.


```
{
  "runId": "Example",
  "caseId": "example",
  "bundle": {
    "type": "bundle",
    "id": "bundle--44af6c39-c09b-49c5-9de2-394224b04982",
    "objects": [
      {
        "type": "identity",
        "spec_version": "2.1",
        "id": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283",
        "created": "2015-01-21T19:59:17.000Z",
        "modified": "2015-01-21T19:59:17.000Z",
        "name": "Acme Cybersecurity Solutions"
      },
      {
        "type": "report",
```

```
  "spec_version": "2.1",
  "id": "report--84e4d88f-44ea-4bcd-bbf3-b2c1c320bcbd",
  "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74
    ↪ e4f1658283",
  "created": "2015-12-21T19:59:11.000Z",
  "modified": "2016-05-21T19:59:11.000Z",
  "name": "The Black Vine Cyberespionage Group",
  "description": "A simple report with an indicator and
    ↪ campaign",
  "published": "2016-01-20T17:00:00.000Z",
  "report_types": [
    "campaign"
  ],
  "object_refs": [
    "indicator--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
    "campaign--83422c77-904c-4dc1-aff5-5c38f3a2c55c",
    "relationship--f82356ae-fe6c-437c-9c24-6b64314ae68a"
  ]
},
{
  "type": "indicator",
  "spec_version": "2.1",
  "id": "indicator--26ffb872-1dd9-446e-b6f5-d58527e5b5d2
    ↪ ",
  "created": "2015-12-21T19:59:17.000Z",
  "modified": "2016-05-21T19:59:17.000Z",
  "name": "Some indicator",
  "indicator_types": [
    "malicious-activity"
  ],
  "pattern": "[ file:hashes.MD5 = '3773
    ↪ a88f65a5e780c8dff9cdc3a056f3' ]",
  "valid_from": "2015-12-21T19:59:17Z",
  "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74
    ↪ e4f1658283"
},
{
  "type": "campaign",
  "spec_version": "2.1",
  "id": "campaign--83422c77-904c-4dc1-aff5-5c38f3a2c55c",
  "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74
    ↪ e4f1658283",
  "created": "2015-12-21T19:59:17.000Z",
```

```
        "modified": "2016-05-21T19:59:17.000Z",
        "name": "Some Campaign"
      },
      {
        "type": "relationship",
        "spec_version": "2.1",
        "id": "relationship--f82356ae-fe6c-437c-9c24-6
           ↪ b64314ae68a",
        "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74
           ↪ e4f1658283",
        "created": "2015-12-21T19:59:17.000Z",
        "modified": "2015-12-21T19:59:17.000Z",
        "source_ref": "indicator--26ffb872-1dd9-446e-b6f5-
           ↪ d58527e5b5d2",
        "target_ref": "campaign--83422c77-904c-4dc1-aff5-5
           ↪ c38f3a2c55c",
        "relationship_type": "indicates"
      }
    ]
  },
  "services": []
}
```

Listing A.2: Example report, taken from the STIX 2.1 standard[18].

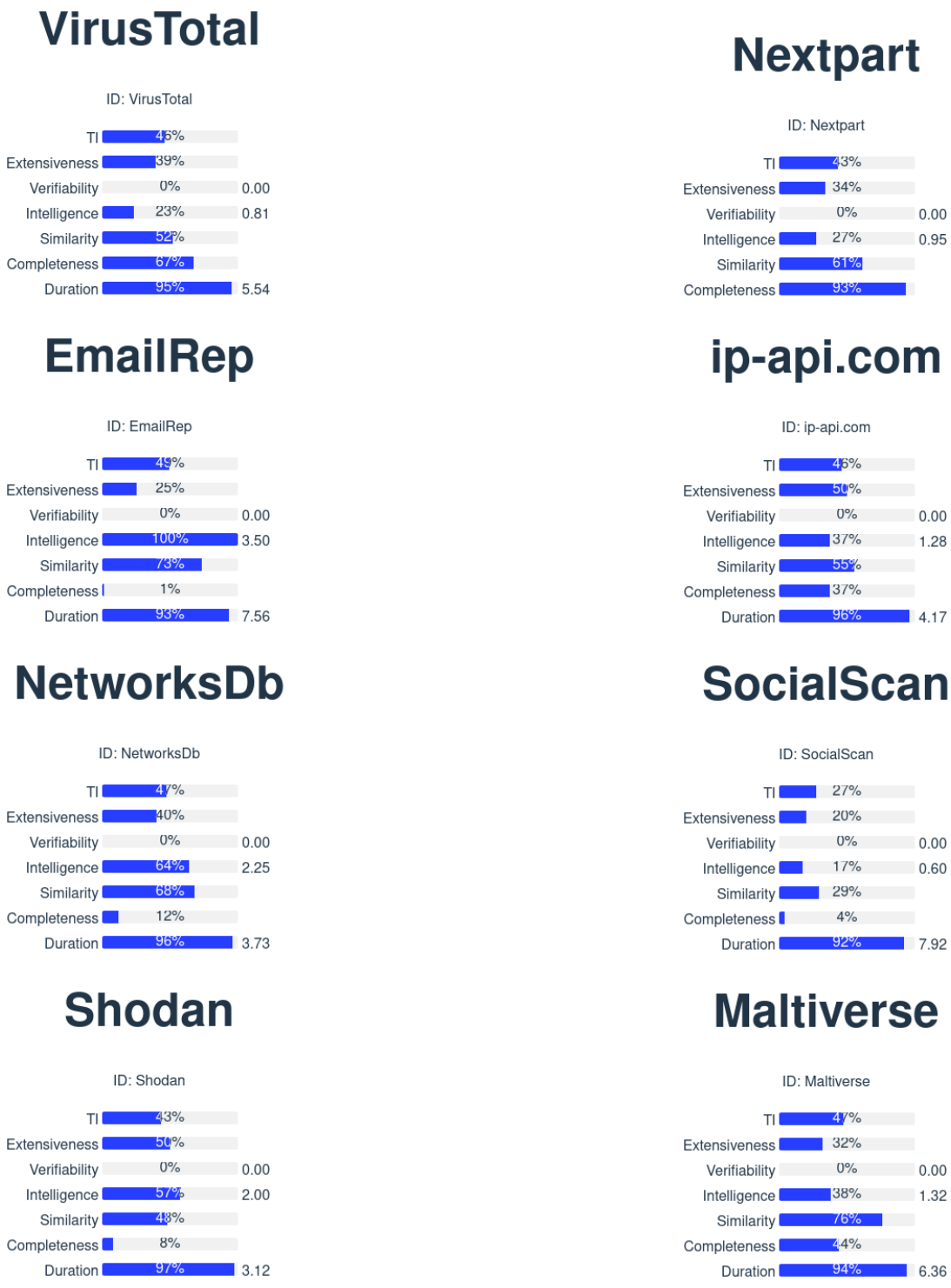


Figure A.1: Metrics for an email with all services enabled.

Algorithm A.1: Comparing the properties of two STIX objects.

```
1  Function CompareProperties(O_This, O_Other):
2      for Property ∈ O_This.Properties do
3          if Property ∉ O_Other.Properties then
4              Skip;
5          end
6          OtherProperty := O_Other.Properties[Property.Name];
7          begin
8              switch Property.Name do
9                  case ExternalReference do
10                     SimilarityMap[Property.Name] =
                        ExternalReferencesSimilarity(Property.ExternalReferences,
                        OtherProperty.ExternalReferences);
11                 end
12                 case KillChainPhases do
13                     SimilarityMap[Property.Name] =
                        KillChainPhasesSimilarity(Property.KillChainPhases,
                        OtherProperty.KillChainPhases);
14                 end
15                 case Default do
16                     begin
17                         switch Property.Type do
18                             case List do
19                                 SimilarityMap[Property.Name] =
                                    SliceSimilarity(Property.Elements,
                                    OtherProperty.Elements);
20                             end
21                             case String do
22                                 SimilarityMap[Property.Name] =
                                    StringSimilarity(Property, OtherProperty);
23                             end
24                             case Number do
25                                 SimilarityMap[Property.Name] =
                                    NumberSimilarity(Property, OtherProperty);
26                             end
27                         end
28                     end
29                 end
30             end
31         end
32     end
33     return SimilarityMap;
```

List of Figures

2.1 Components of a SOAR platform. Source [13]. 8
2.2 An email file being uploaded to Guardia. 9
2.3 The resulting STIX graph as displayed by Guardia. 10

3.1 The three steps of FeedRank: Feed Collection, Correlation Graph & Contribution Analysis and Feed Rating. Source [30]. 14
3.2 Three data quality levels. Source [35]. 18

4.1 The Enodo architecture. 32
4.2 The Enodo database structure. 33
4.3 Dagster workflow as shown on the Dagster GUI. 34
4.4 Sequence diagram of the hook workflow. 35
4.5 Sequence diagram of the scraper workflow. 36
4.6 The Enodo frontend. 47

5.1 Metrics of the report bundle provided by the STIX standard. 50
5.2 Types contained within the resulting STIX bundle. 52
5.3 Metrics for the email workflow with a single service enabled. 52
5.4 Types contained within the resulting STIX bundle. 54
5.5 Types contained within the resulting STIX bundle. 56
5.6 Metrics for the hash workflow with all services enabled. 56

A.1 Metrics for an email with all services enabled. 67

List of Tables

2.1 The taxonomy of STIX. Source [18]. 6

3.1 The extensiveness computation [34]. 16
3.2 The freshness computation [34]. 16
3.3 Overview of metrics and their fulfilment of requirements. 26
3.4 The final metrics and their corresponding formulas. 30

4.1 Items needed for the analyzer and their sources. 34

5.1 System specifications of the test system. 49
5.2 Weight map for the similarity calculation. 50
5.3 Metrics for the email workflow with all services enabled. 54

List of Algorithms

4.1 Calculation of Extensiveness. 38
4.2 Calculation of Verifiability. 39
4.3 Calculation of Intelligence. 40
4.4 Calculation of Similarity. 41
4.5 Calculation of the similarity map. 42
4.6 Calculation of object similarity. 43
4.7 Calculation of Completeness. 43
4.8 Calculation of an indicator. 44
4.9 Calculation of Duration. 44

A.1 Comparing the properties of two STIX objects. 68

Glossary

Dagster Dagster is a data orchestration platform used to schedule playbooks. 31–37, 45, 69, 75

Docker Docker is a software platform allowing for simple deployment of applications within containers. 50

Enodo Enodo is a CTI scoring framework within the Guardia architecture. 31–33, 39, 46, 47, 49, 50, 52, 55, 59, 61, 69, 75

Go Go, also called Golang, is a programming language by Google and is designed to be used in microservices. 31, 45

GraphQL GraphQL is a query language for APIs used by Dagster and Enodo. 36, 37, 46

Guardia SOAR platform created by Nextpart. vii, 2, 8–12, 22, 25, 28, 29, 31, 32, 35–39, 44–46, 49–51, 53, 55, 69, 75

Nextpart Nextpart Security Intelligence GmbH is the primary stakeholder of this thesis. vii, 2, 3, 45, 54, 55, 75

OASIS Organization for the Advancement of Structured Information Services is a nonprofit consortium providing the STIX standard. 19

Op Op is an atomic code fragment used within Dagster workflows to execute playbooks. 33

Python Python is a programming language used frequently in data science and general programming. 31–33, 45

Svelte Svelte is a front end compiler, combining Hypertext Markup Language (HTML) and JavaScript code. 46

Vite Vite is a build tool for web projects. 46

WHOIS WHOIS queries can be used to obtain information about domains, IP blocks or autonomous systems. 9

Acronyms

API Application Programming Interface. 24, 25, 32, 33, 35–37, 45, 46

CSV Comma-seperated Values. 22

CTI Cyber Threat Intelligence. vii, ix, xi, 1–3, 5, 6, 10–15, 17, 19–25, 27, 28, 31, 32, 44, 60, 61, 75

CVE Common Vulnerabilities and Exposures. 59

CybOX Cyber Observable Expression. 1

DQ data quality. ix, 17, 18, 21, 62, 69

E-EWS ECHO - Early Warning System. 15

GUI Graphical User Interface. 34, 46, 55, 69

HTML Hypertext Markup Language. 75

HTTP Hypertext Transfer Protocol. 38

IP Internet Protocol. 76

JSON Java Script Object Notation. 5, 6, 19, 37, 38, 42, 61

OpenIOC Open Indicators of Compromise. 1, 22

ORM Object–Relational Mapping. 31, 45, 46

SCO STIX Cyber-observable Object. 6, 7, 30, 45, 46, 61

SDO STIX Domain Object. 6, 7, 17, 41, 51, 53, 57

SIRP Security incident response platform. 7

SMO STIX Meta Objects. 6

SOA Security orchestration and automation. 7

SOAR Security Orchestration, Automation, and Response. ix, xi, 1–3, 5, 7–10, 13, 26–28, 30, 45, 60–62, 69, 75

SOC Security Operations Center. 1, 5, 7, 11

SRO STIX Relationship Object. 6, 7, 17, 51

STIX Structured Threat Information Expression. ix, xi, 1, 3, 5, 6, 9–14, 17–19, 21, 22, 24–30, 32, 35–40, 42, 44, 45, 50, 52–54, 56, 57, 60, 61, 66, 68, 69, 71, 73, 75

TAXII Trusted Automated Exchange of Intelligence Information. 45

TIP Threat intelligence platform. 7

TQM Trust and Quality Tool. 15, 25

URL Uniform Resource Locator. 37, 62

VERIS Vocabulary for Event Recording and Incident Sharing. 1

XML Extensible Markup Language. 5

Bibliography

[1] R. Brown and R. M. Lee, „2021 SANS Cyber Threat Intelligence (CTI) Survey“, SANS Institute, Tech. Rep., 2021. [Online]. Available: https://www.sans.org/white-papers/40080/.

[2] T. D. Wagner, K. Mahbub, E. Palomar, and A. E. Abdallah, „Cyber threat intelligence sharing: Survey and research directions“, *Computers & Security*, vol. 87, p. 101 589, Nov. 2019, ISSN: 01674048. DOI: 10.1016/j.cose.2019.101589. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S016740481830467X.

[3] Camille Singleton, Charlotte Hammond, Vio Onut, *et al.*, „X-Force Threat Intelligence Index 2022“, IBM, Tech. Rep., 2022.

[4] ISACA, „State of Cybersecurity 2022“, Tech. Rep., 2022.

[5] Secureworks, „Threat Intelligence Executive Report“, Tech. Rep., 2022.

[6] K. I. Sgouras, A. D. Birda, and D. P. Labridis, „Cyber attack impact on critical Smart Grid infrastructures“, in *ISGT 2014*, IEEE, Feb. 2014, pp. 1–5, ISBN: 978-1-4799-3653-3. DOI: 10.1109/ISGT.2014.6816504. [Online]. Available: http://ieeexplore.ieee.org/document/6816504/.

[7] M. J. Covington and R. Carskadden, „Threat Implications of the Internet of Things“, in *2013 5th International Conference on Cyber Conflict (CYCON 2013)*, K. Podins, J. Stinissen, and M. Maybaum, Eds., Tallinn: NATO CCD COE, Jun. 2013, pp. 1–12.

[8] K. Boeckl, M. Fagan, W. Fisher, N. Lefkovitz, K. N. Megas, E. Nadeau, D. G. O'Rourke, B. Piccarreta, and K. Scarfone, „Considerations for managing Internet of Things (IoT) cybersecurity and privacy risks“, National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., Jun. 2019. DOI: 10.6028/NIST.IR.8228. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8228.pdf.

[9] Federal Bureau of Investigation, „Internet Crime Report 2021“, Tech. Rep., 2021. [Online]. Available: https://www.ic3.gov/Media/PDF/AnnualReport/2021_IC3Report.pdf.

[10] R. Brown and P. Stirparo, „SANS 2022 Cyber Threat Intelligence Survey“, SANS Institute, Tech. Rep., Feb. 2022. [Online]. Available: https://www.sans.org/white-papers/sans-2022-cyber-threat-intelligence-survey/.

[11] C. Sauerwein, C. Sillaber, A. Mussmann, and R. Breu, „Threat Intelligence Sharing Platforms: An Exploratory Study of Software Vendors and Research Perspectives“, in *Proceedings der 13. Internationalen Tagung Wirtschaftsinformatik*, St. Gallen, 2017, pp. 837–851.

[12] Ponemon Institute, „The Value of Threat Intelligence: Annual Study of North American & United Kingdom Companies“, Tech. Rep., Feb. 2019.

[13] C. Lawson and A. Price, „Market Guide for Security Orchestration, Automation and Response Solutions“, Gartner, Tech. Rep., Jun. 2022.

[14] T. D. Wagner, E. Palomar, K. Mahbub, and A. E. Abdallah, „A Novel Trust Taxonomy for Shared Cyber Threat Intelligence“, *Security and Communication Networks*, vol. 2018, pp. 1–11, Jun. 2018, ISSN: 1939-0114. DOI: 10.1155/2018/9634507. [Online]. Available: https://www.hindawi.com/journals/scn/2018/9634507/.

[15] X. Bouwman, H. Griffioen, J. Egbers, C. Doerr, B. Klievink, and M. Van Eeten, „A Different Cup of TI? The Added Value of Commercial Threat Intelligence“, in *Proceedings of the 29th USENIX Conference on Security Symposium*, USA: USENIX Association, 2020, ISBN: 978-1-939133-17-5.

[16] R. M. Blank and P. D. Gallagher, „Guide for conducting risk assessments“, National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., 2012. DOI: 10.6028/NIST.SP.800-30r1. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-30r1.pdf.

[17] C. S. Johnson, M. L. Badger, D. A. Waltermire, J. Snyder, and C. Skorupka, „Guide to Cyber Threat Information Sharing“, National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., Oct. 2016. DOI: 10.6028/NIST.SP.800-150. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-150.pdf.

[18] B. Jordan, R. Piazza, and T. Darley, *STIX Version 2.1*, Jun. 2021. [Online]. Available: https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html.

[19] OASIS Open, *Comparing STIX 1.X/CybOX 2.X with STIX 2*, Apr. 2022. [Online]. Available: https://oasis-open.github.io/cti-documentation/stix/compare.

[20] M. Vielberth, F. Bohm, I. Fichtinger, and G. Pernul, „Security Operations Center: A Systematic Study and Open Challenges“, *IEEE Access*, vol. 8, pp. 227 756–227 779, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3045514. [Online]. Available: https://ieeexplore.ieee.org/document/9296846/.

[21] FireEye, *What is SOAR? Security definition | FireEye*. [Online]. Available: https://www.fireeye.com/products/helix/what-is-soar.html.

[22] N. Schrock and M. Gasner, *Dagster: The Data Orchestrator*, Aug. 2020. [Online]. Available: https://dagster.io/blog/dagster-the-data-orchestrator.

[23] A. Tundis, S. Ruppert, and M. Mühlhäuser, „A Feature-driven Method for Automating the Assessment of OSINT Cyber Threat Sources“, *Computers & Security*, vol. 113, p. 102 576, Feb. 2022, ISSN: 01674048. DOI: 10.1016/j.cose.2021.102576. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167404821004004.

[24] H. Griffioen, T. Booij, and C. Doerr, „Quality Evaluation of Cyber Threat Intelligence Feeds“, in *Applied Cryptography and Network Security*, M. Conti, J. Zhou, E. Casalicchio, and A. Spognardi, Eds., Cham: Springer International Publishing, 2020, pp. 277–296, ISBN: 978-3-030-57878-7. DOI: 10.1007/978-3-030-57878-7_14.

[25] V. G. Li, M. Dunn, P. Pearce, D. McCoy, G. M. Voelker, S. Savage, and K. Levchenko, „Reading the Tea Leaves: A Comparative Analysis of Threat Intelligence“, in *Proceedings of the 28th USENIX Conference on Security Symposium*, ser. SEC'19, USA: USENIX Association, 2019, pp. 851–867, ISBN: 9781939133069.

[26] G. González-Granadillo, M. Faiella, I. Medeiros, R. Azevedo, and S. González-Zarzosa, „ETIP: An Enriched Threat Intelligence Platform for improving OSINT correlation, analysis, visualization and sharing capabilities“, *Journal of Information Security and Applications*, vol. 58, p. 102 715, May 2021, ISSN: 22142126. DOI: 10.1016/j.jisa.2020.102715. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2214212620308589.

[27] A. Ramsdale, S. Shiaeles, and N. Kolokotronis, „A Comparative Analysis of Cyber-Threat Intelligence Sources, Formats and Languages“, *Electronics*, vol. 9, no. 5, p. 824, May 2020, ISSN: 2079-9292. DOI: 10.3390/electronics9050824. [Online]. Available: https://www.mdpi.com/2079-9292/9/5/824.

[28] F. Menges and G. Pernul, „A comparative analysis of incident reporting formats“, *Computers & Security*, vol. 73, pp. 87–101, Mar. 2018, ISSN: 01674048. DOI: 10.1016/j.cose.2017.10.009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167404817302250.

[29] O. Serrano, L. Dandurand, and S. Brown, „On the Design of a Cyber Security Data Sharing System“, in *Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security - WISCS '14*, ser. WISCS '14, New York, New York, USA: ACM Press, 2014, pp. 61–69, ISBN: 9781450331517. DOI: 10.1145/2663876.2663882. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2663876.2663882.

[30] R. Meier, C. Scherrer, D. Gugelmann, V. Lenders, and L. Vanbever, „FeedRank: A tamper- resistant method for the ranking of cyber threat intelligence feeds“, in *2018 10th International Conference on Cyber Conflict (CyCon)*, vol. 2018-May, IEEE, May 2018, pp. 321–344, ISBN: 978-9-9499-9042-9. DOI: 10.23919/CYCON.2018.8405024. [Online]. Available: https://ieeexplore.ieee.org/document/8405024/.

[31] L. Page, S. Brin, R. Motwani, and T. Winograd, „The PageRank Citation Ranking: Bringing Order to the Web.“, Stanford InfoLab, Tech. Rep. 1999-66, Nov. 1999. [Online]. Available: http://ilpubs.stanford.edu:8090/422/.

[32] S. Brin and L. Page, „The anatomy of a large-scale hypertextual Web search engine“, *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, Apr. 1998, ISSN: 01697552. DOI: 10.1016/S0169-7552(98)00110-X. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S016975529800110X.

[33] Wikipedia contributors, *PageRank — Wikipedia, The Free Encyclopedia*, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=PageRank&oldid=1100639192.

[34] K. B. Mavzer, E. Konieczna, H. Alves, C. Yucel, I. Chalkias, D. Mallis, D. Cetinkaya, and L. A. G. Sanchez, „Trust and Quality Computation for Cyber Threat Intelligence Sharing Platforms“, in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, IEEE, Jul. 2021, pp. 360–365, ISBN: 978-1-6654-0285-9. DOI: 10.1109/CSR51186.2021.9527975. [Online]. Available: https://ieeexplore.ieee.org/document/9527975/.

[35] D. Schlette, F. Böhm, M. Caselli, and G. Pernul, „Measuring and visualizing cyber threat intelligence quality“, *International Journal of Information Security*, vol. 20, no. 1, pp. 21–38, Feb. 2021, ISSN: 1615-5262. DOI: 10.1007/s10207-020-00490-y. [Online]. Available: http://link.springer.com/10.1007/s10207-020-00490-y.

[36] M. S. Charikar, „Similarity estimation techniques from rounding algorithms“, in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing - STOC '02*, New York, New York, USA: ACM Press, 2002, p. 380, ISBN: 1581134959. DOI: 10.1145/509907.509965. [Online]. Available: http://portal.acm.org/citation.cfm?doid=509907.509965.

[37] T. Schaberreiter, V. Kupfersberger, K. Rantos, A. Spyros, A. Papanikolaou, C. Ilioudis, and G. Quirchmayr, „A Quantitative Evaluation of Trust in the Quality of Cyber Threat Intelligence Sources“, in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, New York, NY, USA: ACM, Aug. 2019, pp. 1–10, ISBN: 9781450371643. DOI: 10.1145/3339252.3342112. [Online]. Available: https://dl.acm.org/doi/10.1145/3339252.3342112.

[38] P. Jaccard, „Lois de distribution florale dans la zone alpine“, *Bulletin de la Société vaudoise des sciences naturelles*, vol. 38, no. 144, 1902.

[39] Elementl, *Op Hooks | Dagster*. [Online]. Available: https://docs.dagster.io/0.15.0/concepts/ops-jobs-graphs/op-hooks.

[40] J. Bret and O. Morozov, *FreeTAXII/libstix2*. [Online]. Available: https://github.com/freetaxii/libstix2.

[41] O. Krauss and K. Papesh, „Analysis of Threat Intelligence Information Exchange via the STIX Standard“, in *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, forthcoming.