MDPI

Article

Metrics proposal to compare Threat Hunting Languages

Firstname Lastname [1,†,‡], Firstname Lastname [1,‡] and Firstname Lastname [2,*]

[1] Affiliation 1; e-mail@e-mail.com
[2] Affiliation 2; e-mail@e-mail.com
* Correspondence: e-mail@e-mail.com; Tel.: (optional; include country code; if there are multiple corresponding authors, add author initials) +xx-xxxx-xxx-xxxx (F.L.)
† Current address: Affiliation 3
‡ These authors contributed equally to this work.

Version July 23, 2020 submitted to Journal Not Specified

Abstract: Threat hunting is the process of identifying weaknesses, threats, and ongoing attacks through data analysis by proactively searching for indicators of compromise that has not yet been flagged by various detection technologies. For automated detection of malicious activity, rules are defined across languages. Some of these languages, such as Structured Threat Information Expression, MITRE Cyber Analytics Repository, Kusto Query Language, Endgame Query Language, and SIGMA were introduced to provide common means of sharing cyber-threat intelligence and threat hunting platforms. This paper investigates the landscape of these formats and languages. The contribution of the study is a proposal of metrics to compare them. It is validated with an example case that rules can be migrated from one language to another. The MISP tool is used, a platform for sharing threat intelligence. The demonstration is done with an example written in Sigma, from which a MISP event is generated. Such MISP event is then converted to STIX language. The analysis of the main findings suggests that the cybersecurity community lacks standardization covering the complete spectrum of threat hunting; therefore, sharing data and information about threats is essential.

Keywords: cybersecurity; cyber threat intelligence; indicators of compromise; threat exchange; threat hunting

1. Introduction

Defending an enterprise network against cyberattacks remains an increasingly difficult challenge that requires, among other things, advanced technologies and approaches for thwarting adversary goals. [1] The objective of security is not just to stop adversaries, it is also to control and minimize the overall damage from an incursion. The main method for finding adversaries already in our networks is threat hunting, an area on which security personnel are increasingly focusing their attention. Threat hunting is the act of aggressively tracking and eliminating cyber adversaries from your network as early as possible. [2]

The early detection of an adversary, as well as faster removal and repair of vulnerabilities uncovered during the hunt, are the main role played by threat hunting. Furthermore, a second valuable purpose of hunting is to verify and validate how well existing tools are working in your environment. Indicators of Compromise (IOCs) are detective in nature and describe how to recognize malicious or suspicious behavior that directly detects campaigns, tactics, techniques and procedures, attack patterns, malware, tools, and threat actors. [3]

Cyber threat information is any information that can help an organization identify, assess, monitor, and respond to cyber threats. Examples of cyber threat information include indicators (system

artifacts or observables associated with an attack), TTPs, security alerts, threat intelligence reports, and recommended security tool configurations. Most organizations already produce multiple types of cyber threat information that is available to share internally as part of their information technology and security operations efforts.

Cyber-threat intelligence (CTI) focuses on the capabilities, motivations and goals of an adversary and how these could be achieved. Intelligence is the information and knowledge gained about an adversary through observation and analysis; intelligence is not just data, but the outcome of an analysis and must be actionable to meet the needs of current defensive systems that have to deal with and respond to cyber-attacks. [4][5].

Many threat hunting formats were identified from CTI sources and the literature; these were selected for further analysis based on their popularity in the literature or the source feeds. The analysis carried out in this paper takes into account prominent representatives of CTI formats and threat languages such as the Structured Threat Information Expression, MITRE Cyber Analytics Repository, Kusto Query Language, Endgame Query Language, and SIGMA.

Exploring the capabilities of the aforementioned threat hunting languages is one of the paper's goals. While reviewing the relevant literature, several issues were identified that should be addressed to allow their wide adoption; these include:

- Problems when sharing cyber-threats due to the lack of integration with existing technologies on the market.
- Lack of standardization in field names used in threat hunting languages, making it difficult to create threat hunting rules.
- Difficulties to adapt the languages rules to our platforms according to the technologies used.

In order to study the alternatives that might solve the first problem identified in this paper, we look into the possibilities of migrating existing rules in one language to another. However, our main contribution would be the definition of metrics to compare these languages, based on the benefits they offer.

The organisation of the paper is as follows. We first provide a quick overview of the current state of the art in Section 2, to have a knowledge base and an informed perspective on the findings and main features. This is followed by Sections 3 that compare and analyze the limitation among the languages and present the main result of our analysis. Section 4 validate a case example of migration from Sigma to STIX, and we conclude in section 5.

2. State of the art

The threat landscape is complexe, the events occurs at higher speed every day, and the vast quantities of data involved in cyber threat intelligence and threat information sharing, establishing automation to aid human analysis or execute defensive actions at machine-speed is a prerequisite for any effective approach.[6] [7]

The combination of all of these factors will require standardized, structured threat information representations so the widest possible community of information sources can contribute and leverage information without knowing ahead of time who will be providing what information.[6]

Given all of these factors, there exists a need for structured representations of threat information that are expressive, flexible, extensible, automatable and readable. This paper discuss threat hunting languages, as a solution to this problem. [6]

Throughout this section, the main characteristics of Structured Threat Information Expression (STIX), MITRE Cyber Analytics Repository (CAR), Kusto Query Language (KQL), Endgame Query Language (EQL) and SIGMA are reviewed. It is presented a tool for sharing threat intelligence: MISP. It is divided into six sections, each of which summarizes the state of the art of a threat hunting language and MISP tool.

2.1. STIX

Structured Threat Information Expression is a language and serialization format. STIX is a rich and extensive XML format and aims to extend indicator sharing to enable the management and widespread exchange of significantly more expensive sets of indicators end other full-spectrum cyber threat information. [8] [9]

STIX is open source and free allowing those interested to contribute and ask questions freely [10]. A variety of high-level cyber security use cases rely on such information including: analyzing cyber threats, specifying indicator patterns for cyber threat, managing cyber threat response activities and sharing cyber threat information. STIX is an expressive, flexible, and extensible representation language used to communicate general information about threats. [6]

STIX architecture is comprised of several cyber threat information such as cyber observables, indicators, incidents, adversaries, tactics, techniques, procedures, exploit targets, courses of action, cyber-attack campaigns, and threat actors. [11]

Observables describe what has been or might be seen in cyber

Indicators describe patterns for what might be seen and what they mean if they are

Incidents describe instances of specific adversary actions

Adversary Tactics, Techniques, and Procedures describe attack patterns, malware, exploits, kill chains, tools, infrastructure, victim targeting, and other methods used by the adversary

Exploit Targets describe vulnerabilities, weaknesses, or configurations that might be exploited

Courses of Action describe response actions that may be taken in response to an attack or as a preventative measure

Campaigns describe sets of incidents and/or TTPs with a shared intent

Threat Actors describe identification and/or characterization of the adversary

Reports collect related STIX content and give them shared context

2.2. MITRE Cyber Analytics Repository

The MITRE Cyber Analytics Repository is a knowledge base of analytics developed by MITRE based on the MITRE ATT&CK adversary model. [12]

Analytics stored in CAR contain the following information:

- a hypothesis which explains the idea behind the analytic
- the information domain or the primary domain the analytic is designed to operate within (e.g. host, network, process, external)
- a pseudocode description of how the analytic might be implemented
- a unit test which can be run to trigger the analytic In addition to the analytics, CAR also contains a data model for observable data used to run the analytics and sensors that are used to collect that data. [12]

CAR contains analytics mapped to specific ATT&CK techniques and describes the high-level analytic hypothesis, pseudocode analytic implementation, unit tests, and the data model used to develop them so the analytics can be transcribed to various analytic platforms. CAR is intended to be used by cyber defenders throughout the community and serves as a mechanism for sharing behavioral-based analytics that can be used for adversary detection. [1] CAR analytics were developed to detect the adversary behaviors in ATT&CK. Development of an analytic is based upon the following activities:

- identifying and prioritizing adversary behaviors from the ATT&CK adversary model.
- identifying the data necessary to detect the adversary behavior.
- identification or creation of a sensor to collect the necessary data.
- the actual creation of the analytic to detect the identified behaviors.

To be able to use the rules of threat hunting provided by CAR in an organization there are some alternatives in the market that intend to enhance this pseudocode, such as KQL, EQL and SIGMA.

2.3. Kusto Query Language

The Kusto Query Language, developed by Microsoft, is a simple but powerful language for consulting structured, semi-structured and unstructured data. It assumes a relational data model of tables and columns with a minimum set of data types. The language is very expressive, easy to read and understand the intent of the query, and optimized for authoring experiences. [13]

KQL is a read-only query language, which processes the data and returns results [14]. The query consists of a sequence of query statements, delimited by a semicolon (;), with at least one statement being a tabular expression statement which is a statement that produces data arranged in a table-like mesh of columns and rows. The query's tabular expression statements produce the results of the query. [15]

The syntax of the tabular expression statement has tabular data flow from one tabular query operator to another, starting with data source (e.g. a table in a database, or an operator that produces data) and then flowing through a set of data transformation operators that are bound together through the use of the pipe (|) delimiter. [15]

For analytics, KQL uses the Kusto Engine to query big data sets, and specifically the large datasets from Azure. The data can be ingested, also, from eexternal sources as well. It can be from custom code in any preferred language like Python, .Net SDK, R, etc. using the Azure Data Explorer API. Data can also be ingested using Event Hub's and Event Grid's, and from the CSV file as well. There are different formats that are supported, like – CSV, TXT, PSV, JSON, Multiline JSON, Avro, Zip, and GZip. Allowing a huge spectrum of data sources for ingesting data. [14]

2.4. Endgame Query Language

EQL is a language that can match events, generate sequences, stack data, build aggregations, and perform analysis. EQL is schemaless and supports multiple database backends. It supports field lookups, boolean logic, comparisons, wildcard matching, and function calls. EQL also has a preprocessor that can perform parse and translation time evaluation, allowing for easily sharable components between queries. [16]

Developed by Endgame (2018) Event Query Language is simple and concise, designed to be a user intuitive SQL-like language. Schema-independent and OS-agnostic. Designed for real-time detection with stream processing. Supports multi events behaviors, stacking and sifting through data Function syntax instead of keyword explosion (e.g. length (field)). [17] Supporting the necessary logic to express relationships between security-relevant events. EQL can compare values and fields with exact or wildcard matching and supports basic AND, OR, NOT boolean search operations. Fields can be compared to strings, integers, decimal values, and against other fields. Most importantly, matching is enabled across a series of events including different types of events (e.g. process, file, and registry) rather than matching on only a single event. [18] EQL has some significant advantages respect other languages. These include:

- Minimal learning curve to write analytics: EQL looks like many other query languages. It is intended to search across structured data in an intuitive manner which is highly conducive to quickly writing behavioral analytics. This also leads to excellent readability of each analytic. It also supports traditional IOC searching, but EQL makes it easy to accurately describe activity and behavior beyond simple IOCs. [18]
- No dependence on particular data sources or schema: Other technologies are tied closely to a given data source, and those writing analytics need to focus heavily on that particular data source and schema to write an analytic instead of just focusing on the logic they're trying to express. EQL's method of abstracting data sources via extensible schema mappings is powerful and allows for easy use without any need to pre-normalize data. [18]
- Built to hunt: EQL includes strong native post-processing capabilities such as sorting, filtering, and stacking, which allow a user to easily filter out noise. Its schema translation capability also makes

it straightforward to extend across multiple data sources without a need for data normalization. Data normalization into the universal schema is supported for users who want to eliminate the need for query-time normalization. [18]

2.5. SIGMA

Sigma is a generic and open signature format that allows you to describe relevant log events in a straight forward manner. The main purpose of SIGMA is to provide a structured form in which researchers or analysts can describe their once developed detection methods. Sigma enables analytics re-use and sharing across organisms. Furthermore decouples rule logic from SIEM vendor and field names.

Sigma rules are written in YAML. The rule format is very flexible, easy to write and applicable to any type of log file. [19]The scheme compound of metadata, log source, detection and condition includes the following classifiers: [20]:

1. Metadata: Title, status, descrption, references, tags, etc.
2. Log Source: Type, brand, and service from the log
3. Detection: List of Selectors
4. Condition: Logic for selector matching

There are a wide currently supported outputs for Sigma, including Splunk , QRadar , ArcSight, Elasticsearch (Elastalert, Query strings, DSL, Watcher, & Kibana) , Logpoint , Qualys , Windows Defender ATP, PowerShell , grep. [20] Sigma rules as object type are supported by MISP, one of the best free Threat Intel Platforms. MISP has wide usage in enterprise and integrates well with other tools via open API. The tool sigma2misp import Sigma rules to MISP events. Sigma is meant to be an open standard in which detection mechanisms can be defined, shared and collected in order to improve the detection capabilities for everyone.[19]

2.6. An Open source threat sharing platform: MISP overview

MISP is a threat intelligence platform for sharing, storing and correlating IoC of targeted attacks, threat intelligence, financial fraud information, vulnerability information or even counter-terrorism information. [21]

Its characteristics make it in an efficient IoC and indicators database allowing to store technical and non-technical information about malware samples, incidents, attackers and intelligence. It allow automatic correlation finding relationships between attributes and indicators from malware, attacks campaigns or analysis. MISP store data in a structured format with an extensive support of cyber security indicators.[21]

Many open source and proprietary tools integrate MISP support in order to extend their tools. A series of additional software are supported and handled by the MISP project. Among the Software within the MISP project it is found the expansion modules, the export modules and import modules. [21]

Importing and exporting data can be done in various ways:

- export: generating IDS (Suricata, Snort and Bro are supported by default), OpenIOC, plain text, CSV, MISP XML or JSON output to integrate with other systems (network IDS, host IDS, custom tools)
- import: bulk-import, batch-import, free-text import, import from OpenIOC, GFI sandbox, ThreatConnect CSV or MISP format.

3. Comparison and Limitations

In order to analyze the common characteristics, identify the differential elements and study the advantages and limitations of the languages mentioned in this section, we define which parameters are the object of the research, and the proposal of these metrics is the main contribution of this work.

With the purpose mentioned we must study the following metrics, which are involved in the detection process:

1. Use ATTCK to identify common behaviors, instead of just tools
2. Explore the mind of the attacker
3. Express detection logic for your platform
4. Continuosly create, test, and refine analytics
5. Share with the community

3.1. Use ATTCK to identify common behaviors, instead of just tools

MITRE ATTCK is a curated knowledge base and model for cyber adversary behavior, reflecting the various phases of an adversary's attack lifecycle and the platforms they are known to target. ATTCK focuses on how external adversaries compromise and operate within computer information networks. At a high-level, ATTCK is a behavioral model that consists of the following core components:

- Tactics, denoting short-term, tactical adversary goals during an attack;
- Techniques, describing the means by which adversaries achieve tactical goals;
- Sub-techniques, describing more specific means by which adversaries achieve tactical goals at a lower level than techniques; and
- Documented adversary usage of techniques, their procedures, and other metadata.

MITRE ATTCK framework, proposes a common language, a standard of techniques and tactics that the community can use. The TTPs described in ATTCK were selected based on observed APT intrusions from public reporting, and are included in the model at a level of abstraction necessary for effectively prioritizing defensive investments and comparing endpoint intrusion detection capabilities. Tactics represent the highest level of abstraction within the ATTCK model. They are the tactical goals an adversary has during an operation. The ATTCK tactic categories are listed here:

- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Execution
- Collection
- Exfiltration
- Command and Control

The techniques in the ATTCK model describe the actions adversaries take to achieve their tactical objectives.

STIX, CAR, EQL, and SIGMA use this ATTCK modeling approach. It is released the ATTCK content as STIX in the GitHub repository and published the ATTCK Navigator, which uses the STIX content to provide an interactive visualization of the ATTCK matrices. The current set of ATTCK tactics and techniques covered by CAR are captures in the ATTCK Navigator layer. In the case of EQL, it is implemented through eqllib, a library of event-based analytics, written in EQL to detect adversary behaviors identified in MITRE ATTCK. In the case of SIGMA, it is implemented through the Sigma2attack tool, which Generates a MITRE ATTCK Navigator heatmap from a directory containing sigma rules. The use of the methodology proposed by MITRE for threat hunting languages is a step forward in language standardization.

3.2. Explore the mind of the attacker

Attacker TTP must be researched and understood to know what to search for in collected data. Information about attacker TTP most often derives from signatures, indicators, and behaviors observed from threat intelligence sources [22]. This added context should include targeted facilities, what systems were affected, protocols manipulated, and any other information pertinent to better understanding an attacker's TTP [23].

By monitoring event patterns that affect many clients, it can be make timely decisions that improve the ability to mitigate emerging threats or provide the community with essential information. In the languages analyzed so far, the behavior of the attackers is explored.

Looking for signatures of known malware or infrastructure IOCs is helpful, but it's not enough. Security practitioners must assume that adversaries have breached defenses and are conducting operations inside their networks.

3.3. Express detection logic for your platform

It is necessary to adapt the rules we define through these languages to the technologies we have in our organization. Pobably the most significant challenge regarding the security community has been the coupling between everyone's unique data sources and their analytics built on top of that data and corresponding schema(s). This has made it difficult to share actionable analytics between teams.

The open source threat hunting languages will improve the community's collective ability to express detection logic and share amongst teams and will summon rapid growth in the number of security teams and researchers working to understand the detection opportunities and challenges in the post-compromise space.

3.4. Continuously create, test, and refine analytics

Continuously create, test, and refine analytics will improve the community's collective ability to express detection logic and share amongst teams. Having carried out an exploratory study of the languages in question, we mention those in which it is possible to contribute.

With the CAR project it's possible make contributions creating new analytics, updates to the data model, and new sensor mappings. There are open issues to all of that.

STIX profiles are a mechanism to describe a particular usage of STIX as practiced by a community, organization, or tool. Producer profiles describe how one particular producer will be creating STIX documents. This is useful for tool developers, commercial threat intelligence feeds, and sharing programs where a single producer is publishing to many consumers.

Endgame provides the EQL core language, a sysmon integration, and a python-based EQL engine, which includes CLI functionality to run EQL queries over JSON. With this toolkit, users can immediately begin prototyping analytics in EQL. The goal is provide a robust documentation on the many interesting EQL primitives and operators which are part of this release. To that end there is available a rich set of analytics called EQLLib to help people become familiar with the language and try things out. With the purpose of writing queries in EQL syntax, the language count with a query guide divided into 7 section: basic syntax, sequences, joins, pipes, functions, implementation details and grammar.

Sigma is an open standard for rules that allow you to describe searches on log data in generic form. These rules can be converted and applied to many log management or SIEM systems and can even be used with grep on the command line. The rule creation process in SIGMA is possible, if you use some tools and draft. With Sigma repository and Sigma rule you can use existing rules as example and create new rule based on a similar existing one. Then, these rules can be tested by using "sigmac".

3.5. Share with the community

In the cyber security community, there is currently a strong need for the exchange of data to support the management of vulnerabilities, threats and incidents, as well as other cyber security activities [24]. The efficient sharing of CTI is at the core of cyber-threat detection and prevention, as it allows building multi-layer automated tools with sophisticated and effective defensive capabilities that continuously analyse the vast amounts of the heterogeneous CTI related to attackers' tactics, techniques and procedures (TTPs), indicators of ongoing incidents, etc[25][26].

STIX is designed as collaborative threat analysis, automated threat exchange, automated detection and response, and more. However, if we look at CAR, is a good starting point for many organizations and can be a great platform for open analytic collaboration-but it isn't the be-all/end-all for defending against the threats described by ATTCK. In the case of EQL and SIGMA, there are conversion tools that allow us to migrate the existing rules of these languages, depending on the technologies we use in our organization.

3.6. Summary and Limitations

Table 1 summarizes the main metrics and characteristics examined in this section and in the literature consulted.

Language	STIX	CAR	KQL	EQL	SIGMA
Use ATTCK to identify common behaviors, instead of just tools	✓	✓	✗	✓	✓
Explore the mind of the attacker	✓	✓	✗	✓	✓
Real Time	✗	✗	✗	✓	✓
Open Source	✓	✓	✗	✓	✓
Create, test, and refine analytics	✓	✓	✗	✓	✓
Share with the Community	✓	✓	✗	✓	✓

Table 1. Metrics Summary

After making an exhaustive review of the existing bibliography of these languages, we conclude that in the design of rules for techniques detection, there are some drawbacks. Different technologies use different field names to define events. It is necessary to standardize the names of the processes. The use of a common nomenclature will allow threat information sharing between different organizations.

In addition to the lack of standardization, other limitations were identified when studying the afore mentioned languages. STIX is a well-defined language, which provides multiple properties such as impact, efficacy, and confidence to describe the usefulness of the shared CTI. However, most of the shared STIX reports do not employ these properties. STIX reports provide indicators and their observables most of the time and they share fewer Course of Actions (COAs), which are instrumental components for the prevention and response phases of cyber threat management.[27] KQL is a language developed by Microsoft and therefore a good tool when using others that are not open source. The CAR project proposes rules expressed in pseudo-code, but it cannot be implemented by itself in an organization. Some proposals on the market advise to reuse this pseudo code, such as EQL and SIGMA. For the design, EQL is great if the first focus is on suspicious and/or anomalous activity. Then, if you are focus on how threat actors are abusing the system, Sigma is preferred. EQL and SIGMA have emerged as languages that can be used in real-time, but the problem of having different definitions persists, which is troublesome for the scientific community.

4. A Case Study: From one language to another using MISP tool

In this section existing rules in a language are converted into another. The purpose is to extend the functionalities of these, analyzing if they are compatible or not, since its implementation in the organizations depends on that. MISP tool is used, the example case is the conversion of SIGMA rules to a MISP event, and then this event transforms it to STIX language.

4.1. Sigma2misp validation

While intended to sharing threat intelligence information, MISP can be used to share other things as well such as Sigma rules. In this subsection , we will be describing how Sigma rules are converted into MISP events.

The Sigma tool sigma2misp pushes rules from files into MISP events. In the directory `misp@misp:~/sigma/tools$` it is modified the file misp.conf, and it is define the following parameters:

```
url https://host
key XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Sigma rules are converted into a new MISP event loading the following command and it is shown in Figure 1.

```
sigma2misp @misp.conf --same-event --info "Test Event" -r /path/to/sigma_rules -I
```

```
misp@misp:~/sigma/tools$ sigma2misp @misp.conf --same-event --info "Test Event" -r ../rules/linux/* -I
Importing Sigma rule ../rules/linux/auditd/lnx_auditd_logging_config_change.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_auditd_ld_so_preload_mod.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_auditd_susp_exe_folders.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_network_sniffing.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_auditd_alter_bash_profile.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_auditd_create_account.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_auditd_auditing_config_change.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_auditd_susp_cmds.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_auditd_user_discovery.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_auditd_masquerading_crond.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_auditd_web_rce.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_data_compressed.yml into MISP event 5...
Importing Sigma rule ../rules/linux/auditd/lnx_auditd_susp_C2_commands.yml into MISP event 5...
Importing Sigma rule ../rules/linux/modsecurity/modsec_mulitple_blocks.yml into MISP event 5...
```

Figure 1. Load Sigma rules in directory rules/ into one newly created MISP event with info set to Test Event

Figure 2 shows the attributes of the MISP event generated. They are obtained through the MISP web interface.

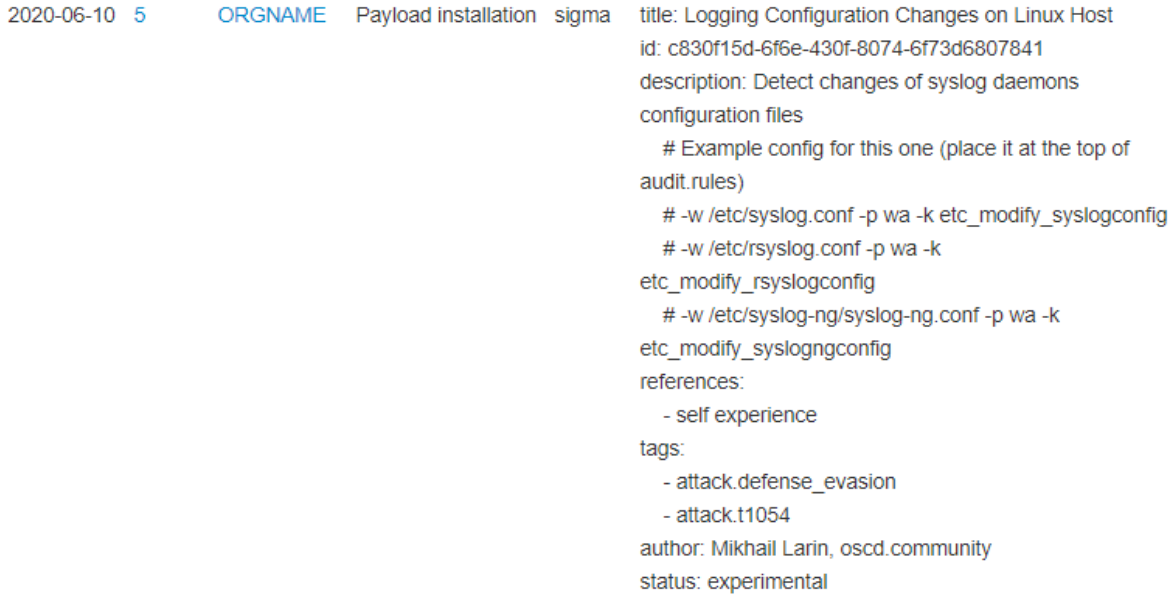

Figure 2. Attributes of the MISP event generated

4.2. STIX Module

By generating STIX rules, MISP allows you to automatically import data in your detection systems resulting in better and faster detection of intrusions.To convert a MISP event to STIX language it is

used the repository MISP-STIX-Converter. This open source tool is part of a MISP project and others contibutors.

During the installation of the repository, the configuration file misp.login located in the MISP-STIX-CONVERTER directory must be created using as template the file misp.login.example. Then mus have been fill in with the data requires to publish MISP events, wich are host address and key.

```
cp /path/to/config/misp.login.example /path/to/config/misp.login
```

In the example case of this paper, the MISP event generated from the file written in SIGMA language is downloaded. Once the JSON file has been downloaded, the following command is used to convert from MISP to STIX, where INFILE and OUTFILE are generic json files, in MISP and STIX language respectively.

```
misp-to-stix.py -f INFILE.json --format JSON -o OUTFILE.json
```

The file OUTFILE.json contains the rules in STIX format and can be seen in the Listing 1.

Listing 1: OUTFILE.JSON

```
"id": "MISPtoSTIX:Package-ebdb9a27-a71b-488b-b81f-237081f37cf7",
    "version": "1.2",
    "stix_header": {
        "title": "Sigma import"
    },
    "observables": {
        "cybox_major_version": "2",
        "cybox_minor_version": "1",
        "cybox_update_version": "0"
    },
    "indicators": [
        {
            "id": "MISPtoSTIX:indicator-0c15ccf0-64f6-47e5-88f9-d11a8af40ab2",
            "timestamp": "2020-06-02T17:01:25.291134+00:00",
            "suggested_coas": {},
            "sightings": {},
            "kill_chain_phases": {},
            "related_indicators": {},
            "related_campaigns": {},
            "related_packages": {}
```

5. Conclusion

Adversarial activity is no longer described purely in terms of static IOCs. Focusing solely on IOCs leads to detections which are brittle and ineffective at discovering unknown attacks, because adversaries modify toolkits to easily evade indicator-based detections. Instead, practitioners need durable detections based on malicious behaviors. [28]

With a comprehensive and robust model of adversarial behavior in place, the best is to build an architecture for event collection that supports hunting and real-time detection, along with a language that promotes usability and performance. The EQL have been created for hunting and real-time detection, with a simple syntax that helps practitioners express complex queries without a high barrier to entry. [28]

The key finding in this research is that there is no common definition of threat hunting sharing platforms. Beside the standards for describing and sharing of threat hunting, research and practice

have not yet developed a comprehensive definition and common understanding of what constitutes a threat hunting sharing platform. Therefore, different types of platforms were identified. [29]

Another conclusion is that most platforms focus on data collection instead of analysis the "intelligence" provided by the majority of threat intelligence sharing. In the context of information security "intelligence" is the product of the intelligence lifecycle model, which includes several activities like planning, data collection, analysis and, dissemination. However, we found that the majority of tools primarily focuses on data collection and more or less neglects the other activities of the intelligence lifecycle. Therefore, most currently available threat intelligence platforms resemble data warehouses more than "real" intelligence sharing platforms. Moreover not all platforms are open source. [29]

By exchanging cyber threat information within a sharing community, organizations can leverage the collective knowledge, experience, and capabilities of that sharing community to gain a more complete understanding of the threats the organization may face. Using this knowledge, an organization can make threat-informed decisions regarding defensive capabilities, threat detection techniques, and mitigation strategies. By correlating and analyzing cyber threat information from multiple sources, an organization can also enrich existing information and make it more actionable. This enrichment may be achieved by independently confirming the observations of other community members, and by improving the overall quality of the threat information through the reduction of ambiguity and errors. Organizations that receive threat information and subsequently use this information to remediate a threat confer a degree of protection to other organizations by impeding the threat's ability to spread. Additionally, sharing of cyber threat information allows organizations to better detect campaigns that target particular industry sectors, business entities, or institutions.

Author Contributions:

Funding:

Acknowledgments:

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Strom, B.E.; Battaglia, J.A.; Kemmerer, M.S.; Kupersanin, W.; Miller, D.P.; Wampler, C.; Whitley, S.M.; Wolf, R.D. Finding Cyber Threats with ATT&CK-Based Analytics **2017**.
2. Cole, E. Threat Hunting Open Season on the Adversary. *SANS* **2016**.
3. Mavroeidis, V.; Bromander, S. Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence. 2017 European Intelligence and Security Informatics Conference (EISIC), 2017, pp. 91–98. ISSN: null, doi:10.1109/EISIC.2017.20.
4. Roberts, S.J.; Brown, R. *Intelligence-Driven Incident Response: Outwitting the Adversary*; "O'Reilly Media, Inc.", 2017. Google-Books-ID: wfwxDwAAQBAJ.
5. Menges, F.; Sperl, C.; Pernul, G. Unifying Cyber Threat Intelligence. Trust, Privacy and Security in Digital Business; Gritzalis, S.; Weippl, E.R.; Katsikas, S.K.; Anderst-Kotsis, G.; Tjoa, A.M.; Khalil, I., Eds.; Springer International Publishing: Cham, 2019; pp. 161–175.
6. Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX^{TM}) **2013**.
7. Shu, X.; Araujo, F.; Schales, D.L.; Stoecklin, M.P.; Jang, J.; Huang, H.; Rao, J.R. Threat Intelligence Computing. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security; ACM: Toronto Canada, 2018; pp. 1883–1898. doi:10.1145/3243734.3243829.
8. Riesco Granadino, R. Contribution to dynamic risk management automation by an ontology-based framework. PhD Thesis, Universidad Politécnica de Madrid, 2019. doi:10.20868/UPM.thesis.57575.
9. Ramsdale, A.; Shiaeles, S.; Kolokotronis, N. A Comparative Analysis of Cyber-Threat Intelligence Sources, Formats and Languages. *Electronics* **2020**, *9*, 824. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, doi:10.3390/electronics9050824.
10. STIX Version 2.0. Part 1: STIX Core Concepts **2017**. p. 71.

11. Stix Project Documentation, 2020.
12. Corporation, T.M. Welcome to the Cyber Analytics Repository. Library Catalog: car.mitre.org.
13. microsoft/Kusto-Query-Language, 2020. original-date: 2019-04-23T12:59:49Z.
14. The Kusto Query Language, 2019. Library Catalog: azure-training.com.
15. orspod. Overview - Azure Data Explorer. Library Catalog: docs.microsoft.com.
16. Endgame. EQL Documentation Release 0.8.6, 2020.
17. Akpınar, K.; Hua, K.A. EQL: Event Query Language for the Sharing of Internet-of-Things Infrastructure and Collaborative Applications Development. Service-Oriented Computing – ICSOC 2016 Workshops; Drira, K.; Wang, H.; Yu, Q.; Wang, Y.; Yan, Y.; Charoy, F.; Mendling, J.; Mohamed, M.; Wang, Z.; Bhiri, S., Eds.; Springer International Publishing: Cham, 2017; Lecture Notes in Computer Science, pp. 73–78. doi:10.1007/978-3-319-68136-8_6.
18. EQL for the masses, 2018. Library Catalog: www.elastic.co.
19. Roth, F. Neo23x0/sigma, 2020. original-date: 2016-12-24T09:48:49Z.
20. Hubbard, J. Sharing is Caring: Improving Detection with Sigma **2018**. p. 34.
21. MISP features and functionalities.
22. Liao, X.; Yuan, K.; Wang, X.; Li, Z.; Xing, L.; Beyah, R. Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security; ACM: Vienna Austria, 2016; pp. 755–766. doi:10.1145/2976749.2978315.
23. Gunter, D. A Practical Model for Conducting Cyber Threat Hunting. p. 16.
24. Dandurand, L.; Serrano, O. Towards improved cyber security information sharing. 2013, pp. 1–16.
25. An Actionable Threat Intelligence system using a Publish-Subscribe communications model | Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security.
26. Poputa-Clean, P. Automated Defense - Using Threat Intelligence to Augment. p. 39.
27. Iqbal, Z.; Anwar, Z. SCERM—A novel framework for automated management of cyber threat response activities. *Future Generation Computer Systems* **2020**, *108*, 687–708. doi:10.1016/j.future.2020.03.030.
28. Introducing Event Query Language, 2019. Library Catalog: www.elastic.co.
29. Sauerwein, C.; Sillaber, C.; Mussmann, A.; Breu, R. Threat Intelligence Sharing Platforms: An Exploratory Study of Software Vendors and Research Perspectives. p. 15.

Sample Availability: Samples of the compounds ..... are available from the authors.