Modeling Cyber Threat Intelligence

Siri Bromander[1,2], Morton Swimmer[3], Martin Eian[1], Geir Skjotskift[1] and Fredrik Borg[1]
[1]*mnemonic AS, Oslo, Norway*
[2]*Department of Informatics, University of Oslo, Norway*
[3]*Trend Micro Research, Germany*

Keywords: Cyber Threat Intelligence, Security, Knowledge Graph, Ontology.

Abstract: For a strong, collective defense in the digital domain we need to produce, consume, analyze and share cyber threat intelligence. With an increasing amount of available information, we need automation in order to be effective. We propose a strict data model for cyber threat intelligence which enables consumption of all relevant data, data validation and analysis of consumed content. The main contribution of this paper is the strictness of the data model which enforces input of information and enables automation and deduction of new knowledge.

1 INTRODUCTION

In recent years we have seen several initiatives to structure and streamline Cyber Threat Intelligence (CTI). Organizations share CTI, and most of the CTI in an average organization comes from external sources. Consuming, normalizing and analyzing CTI from heterogeneous sources are major challenges for CTI analysts. Successful defense against threats depends on automation and to make available CTI more useful. Big data analysis and advanced reasoning may be applied, but these rely on consistent and structured data. We propose the ACT data model to address these challenges.

1.1 Research Motivation

Threat intelligence is served in several formats and channels, and with a varying degree of structure. Having worked with threat intelligence and incident response we had a need for a data model that enabled automation and analysis of our available threat intelligence. Combining all available data in one place, allowing for different data sources to be combined and analyzed, will increase the analysis capability of an analyst and remove repetitive tasks.

We find import and export of CTI from a system to be trivial given the data is stored in a consistent and structured manner, covering all relevant data. How we model our data is hence the foundation for everything else.

A key requirement for automation and analysis is data quality. Data quality is both content and format. We cannot enforce quality of content, but we can enable an analyst to evaluate this. Format consistency can be enforced by a strict data model. This means that a computer knows where to find a certain data type in a data set, and that the data found in that place always is the same type of data. With flexibility within the schema of a data model, this requirement will not be met, removing the ability to automate consumption and analysis across different platforms.

Threat intelligence analysis traditionally requires a large amount of knowledge from the analyst. Adding knowledge into the data model will make the knowledge available to more analysts.

Threat intelligence depends on collaboration between a range of organizations and communities. Any tool or system used by collaborators should be openly available to the community without restrictions, which has been a key motivation for this project.

2 RELATED WORK

There are several attempts at structuring cyber threat intelligence (CTI). The motivations for the different approaches seems to differ and these influence the results.

Barnum et al suggested the Structured Threat Information Expression (STIX) (Barnum, 2012) in

Bromander, S., Swimmer, M., Eian, M., Skjotskift, G. and Borg, F.
Modeling Cyber Threat Intelligence.
DOI: 10.5220/0008875302730280
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 273-280
ISBN: 978-989-758-399-5; ISSN: 2184-4356

2012. This was created with the motivation of *sharing* CTI, preferably as more than just data. STIX was intended as a data exchange format and not a suggestion for how to store the data. STIX was published in version 2.0 in 2017 (OASIS CTI TC, 2017) and argued to be the de facto standard for representing CTI (Sauerwein et al., 2017). At the same time, critics of STIX argue that the flexibility of STIX makes it less useful for automation. As there are different possibilities of expressing the same data and information in addition to a fair amount of data included in custom fields or as comments using English prose (Polzunov and Abraham, 2019), automating consumption and further analysis is difficult.

The Malware Information Sharing Platform (MISP) (Wagner et al., 2016) is a platform for rapid sharing of indicators of compromise and sightings of indicators. The MISP data model is under continuous development (MISP, 2019). The data contained within MISP platforms correspond well with the suggested data model in this paper, however the popularity of the platform and loose data governance has led, in our experience, to a decrease in the consistency of the data.

ATT&CK (MITRE, 2019) is a framework and knowledge base for describing adversary behavior through enumerating adversary groups, tactics, techniques and tools and the relationships between them. The knowledge base is maintained by MITRE, and it is published online. ATT&CK uses a data model with defined relationships for structuring their knowledge base.

The OpenCTI platform (ANSSI et al., 2019) was published in late spring 2019 and is a platform aiming at consuming, analyzing and sharing cyber threat intelligence. The OpenCTI platform is including STIX observables and STIX relationships in its data model. Grakn[1] is used to enable graph querying of the data and includes rule-based reasoning to infer new relationships. To the best of our understanding, OpenCTI is limited to the scope of STIX and thus limits the possibilities of consumption and analysis within the platform.

An ontology, in the field of computer science, is a formal description of concepts and how they are related to each other, often referred to as classes and properties. In turn, ontologies provide computational meaning to data by building relations to the logic in the ontology and thus enables us to use reasoning methods (such as induction or deduction) on our data in our knowledge base. While there are many implementations of knowledge bases and ontologies, the World Wide Web Consortium (W3C) chose a triple model for facts and calls this the Resource Description Framework (RDF). RDF also allows us to implement the RDFS schema language[2] and OWL[3], the ontology language which builds upon RDFS.

There are several ontologies built with the aim to structure security relevant data. They cover a range of data and motivations like data validation, transformation or logical reasoning. An overview of available ontologies may be found in (Mavroeidis and Bromander, 2017). To the extent of our knowledge, none of the available ontologies are suitable for solving our problem alone, however the UTIM[4] ontology is being developed in parallel with this model and it is hoped that one day data from the ACT model will be freely interchangeable with data modeled with UTIM.

The rest of the paper is structured as follows: First we describe the methodology of our work in Section 3. Then we explain the details of the data model, with argumentation for our choices in Section 4, which includes a graph representation of the data model. We discuss our findings in Section 5, and conclude in Section 6.

3 METHODOLOGY

We developed the model using an iterative process basing our design on the relevant threat intelligence data we had available and then testing and updating as needed.

The platform we have used to implement the data model for prototyping and testing has been developed using agile development principles. This is a good fit for our iterative process of data model development.

3.1 Limitations

While the data model is an ontology, it is not implemented in RDFS or OWL, but all content can be exported as triplets. Initial testing of implementing the data model using Protégé[5] has been done in order to find improvements, but the desired reasoning capabilities lead to the need for rule based reasoning, which can be performed on top of the proposed data model with other tools as well.

We need a strict data model to avoid bad data in the knowledge base. The proposed data model requires a certain amount of work to consume new sources of data because of this chosen strictness.

[1] https://grakn.ai/

[2] https://www.w3.org/TR/rdf-schema/

[3] https://www.w3.org/TR/owl2-overview/

[4] Unified Threat Intelligence Model. See: http://www.ti-semantics.com

[5] https://protege.stanford.edu

4 RESULTS

We have created a data model and implemented it using an Apache Cassandra[6] and Elasticsearch[7] backend. We have implemented an Apache TinkerPop[8] graph engine which enables graph querying with the use of the graph query language Gremlin[9].

Note that a graph view is not the same as a graph database. You can display any kind of data, even a flat text file, as a graph, but you cannot use graph queries unless you have a graph engine interfacing with your data.

An implementation of the data model can be found on GitHub[10] under the ISC license. An openly available instance of the same implementation can be found online[11].

We have divided the results section into three: The foundation of our data model and the discussion leading to it, the schema improvement due to additional data, and the choice of allowing placeholder objects.

4.1 Foundation: Objects and Facts

The foundation for our work has been a data model consisting of objects and facts. We can define different object types and different fact types. Thinking of graphs, objects are the vertices and facts are the edges. Objects can be described as nodes and facts may be described as relationships. In the following we use the terms objects and facts.

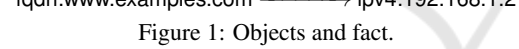

Figure 1: Objects and fact.

The specifications and restrictions to this model is given in the next sections.

4.1.1 Immutable Objects - Retraction of Facts

Objects are defined globally and are immutable. There are no properties linked to an object, everything you know about one object is stored as facts. A fact may connect to one or two objects. A fact is directed, and can be bidirectional.

Deleting a fact is also not possible, however a new fact can be added that retracts the old one. In this way we make sure nothing is deleted and we can prevent repudiation. This way, we also preserve history and check the history of the data set.

[6]http://cassandra.apache.org/

[7]https://www.elastic.co/

[8]http://tinkerpop.apache.org/

[9]https://tinkerpop.apache.org/gremlin.html

[10]https://github.com/mnemonic-no/act-platform

[11]https://act-eu1.mnemonic.no/

4.1.2 Time

Because facts cannot be deleted, we are able to traverse the available data back and forth in time. Using the available threat intelligence in an incident response setting, this is useful for two reasons:

Firstly, knowing exactly what we knew at a given point in time. In situations where a range of decisions are made within a time frame of months, it is useful to be able to turn back time in order to know what information were available at the time when the decision was made. When incorrect decisions have been made, the ability to go back in time and see what information was available at that time will provide the ability to learn from mistakes.

Secondly, knowing how a threat has evolved over time. To know what infrastructure, behavior and resources a given threat actor has used at different times is useful in order to separate threat actors from each other, to identify copycats or impersonation and in order to evaluate how advanced the threat actor is. A threat actor using novel techniques, but abandoning them when they become normal behavior may be considered more advanced than others.

4.2 Data Model

Based on our object/fact foundation, we have defined a set of object types and fact types that are relevant and necessary for our domain.

The initial selection of object types were done influenced by STIX (Barnum, 2012), the Detection Maturity Model (Stillions, 2014), the Diamond Model (Caltagirone et al., 2013), available Open Source Intelligence extracted with the use of Natural Language Processing (NLP) and our own experience.

Fact types were added as we found them useful, with an increasing attention to the semantics and the characteristics of each of them. As our use cases for querying the data expanded, we saw the usefulness of differentiating between fact types.

Figure 2 shows the complete data model schema as a graph. The diamond shapes represent the values of fact types connected to only one object type.

We have populated the data model with a range of sources. A list of openly available sources used so far may be found in Table 1. The data model has been developed and improved along with introduction of new data.

In the following we explain the background and reasoning for the choices we have made, and include results from importing different data sources.

Table 1: The sources influencing the data model.

Source	Relevant object types
ATT&CK	Tactics, Techniques, Tools, Threat Actor
VirusTotal	IPv4, IPv6, FQDN, URI, Content, Hash, Tool, ToolType, Query, Path, Scheme, Basename
Shadowserver ASN	IPv4, IPv4Network, ASN[12], Organization, Country
Passive DNS	IPv4, IPv6, FQDN
MISP Galaxies	Tool, Threat Actor, Sector
STIX vocabularies	Sector
Open Source Intelligence extracted with NLP	All

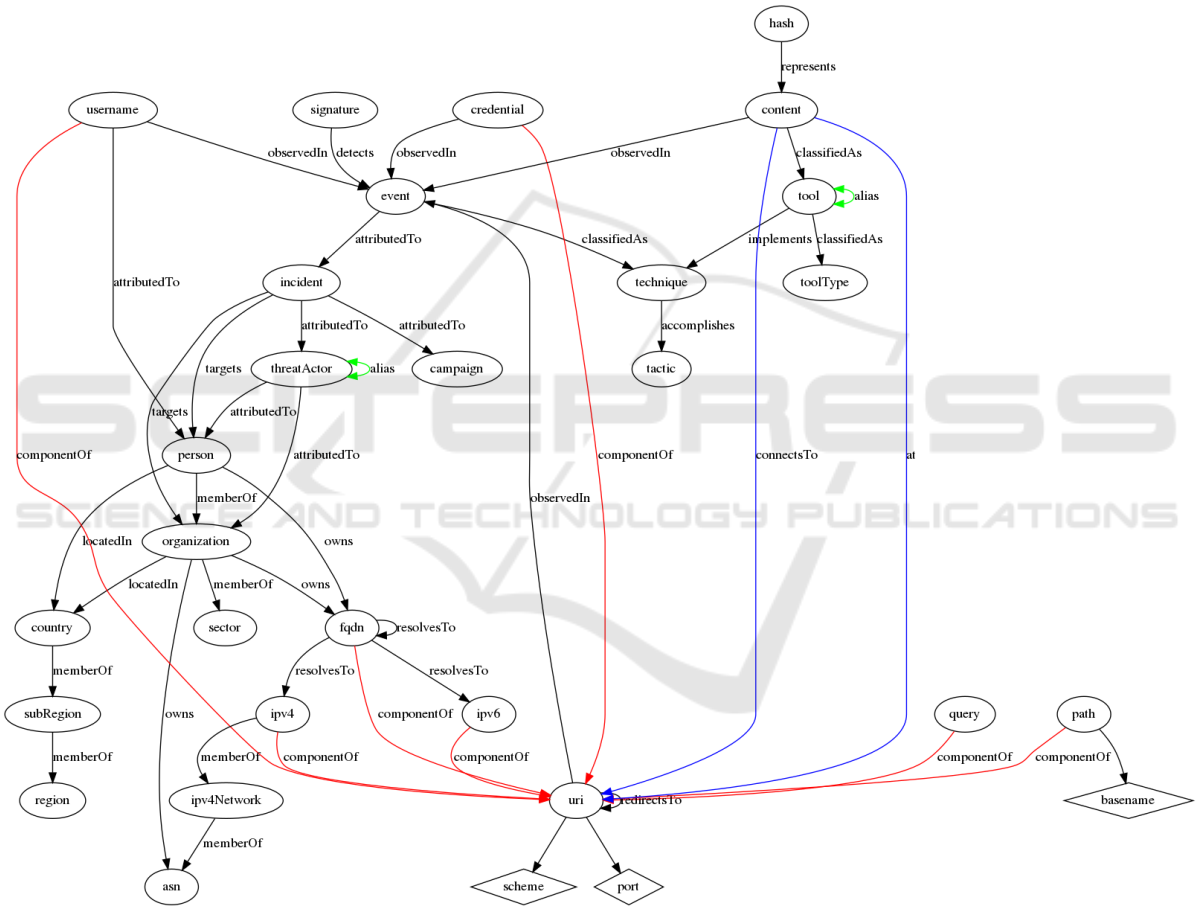

Figure 2: The complete data model represented as a graph.

4.2.1 Consistent Data

There are restrictions on which fact types can be used to link which object types as seen in Figure 2. These restrictions enforce data consistency by preventing different representations of the same data, which is a known problem with current attempts to model CTI. If we allow flexibility in how different data can be represented and introduce a range of users, then the data quality in terms of format is quickly reduced.

The granularity of the data model is intended to be aligned with pivot points normally used by analysts. This has been achieved by covering all sources in the currently available models and sources of CTI, described in the introduction of this section.

We started with differentiation between *malware*, *tool* and *utility*, all instances of software. However, we saw that the definitions of the different groups varied in different sources and became difficult to maintain. This is consistent with the known problem of classifying malware, and we do not attempt to solve this in our data model. When the content of the CTI was not consistent we found that there was no value of using it at all. Therefore in our platform these concepts were all rolled up into *tool*, with the possibility of tagging them as *malware* or *utility* as appropriate.

4.2.2 Enrichment and Query/Analysis Across Sources

One of our first observations was that our graph ended up being a series of subgraphs, and we wanted to be able to connect them. The simple solution was enrichment. As we added more enrichment sources, the graph gradually became more and more interconnected, and we could find new connections between clusters of information that were originally separate.

Pivoting on an object is useful, as it lets you find related information and give you a more comprehensive context. One simple example is from DNS: start with a domain name, find all of the IP addresses that it has resolved to, and then find all other domain names that have resolved to those IP addresses.

Passive DNS (pDNS) data is a historic record of DNS lookup resolutions and is important for an investigation. From 2013 mnemonic has collected pDNS data. By 2017, when we had the initial version of the platform ready for data consumption, we had a TLP:White data set of approximately 100 GB of data. By analyzing super nodes in the data set, we have discovered new and unknown sinkholes. We tag known sinkholes with a fact connecting to the object in order to filter them out when traversing the graph further.

A more advanced solution was to use classifiers to bridge technical, tactical, operational and strategic threat intelligence. An example of this is using VirusTotal to bridge technical indicators to tactical information in MITRE ATT&CK. We extracted the malware family name from anti virus signatures and normalized it. We then normalized the Software entries from MITRE ATT&CK, e.g. “TrickBot” became “trickbot”. Automated enrichment with VirusTotal then connects file hashes and network infrastructure to the “trickbot” object, which is again linked to the tactical threat intelligence in ATT&CK.

We also observed that we could create uncommon pivot points, and our URI object type is an example of this. A URI object is just a UUID connecting different components to each other for a complete URI. Figure 2 shows the facts connecting to a URI in red and blue color. Given a URL, we split it into the host (domain/IP) part, the path and the query parameters. Pivoting on query parameters proved useful when tracking spam campaigns with specific phishing kits, as all of the other pivot points changed for each spam run, but the query parameter stayed the same.

4.2.3 Aliasing

Our data model allows for aliasing different names for the same object.

Instead of giving a threat actor a primary name, like in MISP Galaxy, we use *alias* as a fact type between threat actor names that are known or suggested to be the same. This may also be seen in Figure 2 with green color. Adding information on any threat actor’s name is then done by linking to the name given at the source. In this way, if an alias turns out to be wrong, you only need to retract that one alias, and the rest of your information is still correct.

The problem of different names for the same object is a common situation in CTI. Often, we find different providers of CTI gives a primary name for the object, and connect all information about this object to that name. For instance, if selecting “APT28” as the main name for a threat actor, and receive information about “Fancy Bear” (an alias for APT28), then such a solution will connect the information to “APT28”. This information can be wrong. If you at some point in the future decide that “Fancy Bear” is not an alias for “APT28”, then you would have a large manual task in correcting your data.

The *alias* fact type is used between threat actors and tools and might be applied to other object types in the future.

4.2.4 What is Content?

The concept of *content* is an example of where we need to be precise in order to enable automation. In the context of CTI, we handle not just files, but also stream segments, text strings and parts of content that has been found in memory. This is all “content”, but should not all be classified as files. Furthermore, even in the case of a file, we find that it is seen as unique based on more than one property. We argue that the file name, the actual content, and the location of the content together is what we refer to when we describe something as a unique file.

To illustrate the above we use the example of two files with the file system path `/etc/hosts` on two different Linux machines. In a given situation, the name and content may be the same, but they are still not the same file due to the fact that they reside on different machines. In a different scenario you can find

two files with the same name on the same machine, but with completely different content. In both cases, everyone agrees on the files being different from each other.

To be able to describe these things in a precise manner, and to identify similarities and identical objects, we saw the need for splitting them. The result was *content* linked to *uri* with the fact types as seen in Figure 2 with blue color. Basename (which includes the filename) is included within the URI.

The *at* fact found connecting a content object to a *uri* object is in the meaning of *seen at* and *downloaded from*. The general *at* was selected so as to not exclude any of the terms. The additional *connectsTo* fact represents a content which has been seen connecting to a *uri* and show the two very different scenarios where there is a link between the two object types. This is an example of the importance of semantics when handling CTI.

4.3 Placeholders to Preserve Information

In the ACT data model, you cannot link objects without a defined fact type between the object types. From adding new sources in various structures and formats, we found ourselves in need of adding more fact types based solely upon the information we wanted to consume. This resulted in a vast amount of fact types, and no consistency in representation of information. This is one of the most commonly mentioned weaknesses of the structure given in STIX, where there are several ways of representing the same CTI, resulting in problems digesting all information, especially without manual work and deduplication.

Looking for solutions we found the need for describing "things we know exist, but know little about". Blank nodes has been a solution for this problem in the field of ontologies (Hogan et al., 2014) and is part of the standardized W3C RDF Semantics (W3C, 2014). We introduced the same thought in our data model, by using what we called "placeholders". The idea is that the user may find information about the object in the future, and then replacing the placeholder with an actual object through a new fact. In this way, we were able to strictly define how the data are truly connected to each other, without worrying about having all data in a chain in order to consume it.

As an example, Figure 3 - 5 explains a typical scenario when working with CTI.

After implementing placeholders in our data model and restricting the fact types' possible connections, we found that adding and searching the data

Figure 3: A typical piece of information received as CTI.

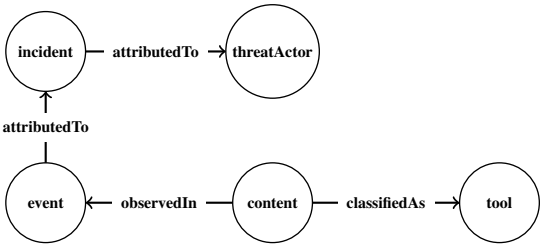

Figure 4: The information needed to give the statement in Figure 3.

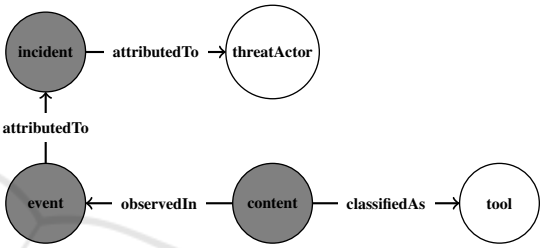

Figure 5: The need for placeholders: the information we know exist in gray, but is often not available for sharing.

gave us an easy overview over what data is missing. This is a very interesting benefit for security analysts receiving or searching data on a relevant incident, both to know what data you do not have, but also to know what data others will need to be in possession of when sending you data. In evaluation of different CTI sources, this is a relevant analysis to perform.

5 DISCUSSION

The data model that we propose is strict: it restricts which relationships may be added to connect two objects, and it enforces that objects may not be added directly but through facts. The main benefit from this is a consistent data set which enables automation and improves data quality. It reduces the computational load of graph queries. It also provides for easier graph queries as there is no need to know the data you query so long as the user understands the data model. As an example there is a limited amount of traversals of the graph between *threat actor* and *technique*. Knowing this makes it trivial to find all connections between known *threat actors* and the *techniques* we know it has used without missing any available data. With this, we argue that building the data model has transferred some of the advanced knowledge from CTI

professionals into the model itself, which enables less skilled professionals to analyze the same data with consistent results.

In the course of developing this model we have discussed various solutions, implemented them and been surprised by some of the findings. The following discourse will look into the most relevant insights.

5.1 Data Validation

Large data sets often include some data that does not comply to the given specification. Adding data to our model, data outside specifications will be identified fast as they will fail upon consumption.

Bad CTI may lead to bad results when analyzing both as they may cause incorrect conclusions but also because they may ruin some of the other data. Most times errors exists due to mistakes entered at the source, because of the complexity of the subject matter or because multiple authors use different methods or terminologies.

We have found that the model allows for data validation. As an example, when querying the data from ATT&CK using our data model, we found that there actually was one technique called *Shared Webroot* without a link to any threat actors or any tools, which in threat intelligence is an interesting observation. Knowing that ATT&CK only includes data they have a reported observation of, means that this technique has been observed, but not described by openly available sources. This was obvious when we applied our model.

Adding MISP Galaxy for threat actors[13] where there is a range of users adding data with limited restrictions on data inclusion, we found that all threat actors were listed under a main name, with all information about them linking to this name. There are aliases listed underneath, but with no capability of reasoning on these aliases, the result is that a large portion of the threat actors actually are connected and seen as one. This meant that the value of the information was diluted as almost all information known about one threat actor was also stated to be valid for a large amount of other threat actors. This is an example of validation that may be used for evaluation of CTI sources, and it shows the importance of the chosen solution of aliasing as chosen in our model.

5.2 Evaluation of CTI Sources

When evaluating different sources of CTI, it is useful to evaluate the quality of the offered data. Our data model may be used for this purpose. Firstly, by adding context and knowledge to your data, which enables you to interpret the data you receive. Extensive aliasing, wrongful classifications or attributions may be easily found through such evaluation. Secondly, it helps finding data with errors, inconsistencies or bad formatting. The strictness of the data model excludes the possibility of importing data with errors, inconsistencies or bad formatting. When working to include new data sources these shortcomings will surface. Thirdly, to check what data is missing. When utilizing the data model with a given data set, if there is missing data it can be identified by identifying missing data in between data points. We can also find what object and fact types are used in that data to evaluate the range of CTI provided from the source.

5.3 Agreeing on Terms and Relationships

The terms and concepts within CTI are often referred to with different understanding. An example of this is *campaign* which often is used to describe standalone incidents and relevant threat actors in addition to the collection of incidents by the same threat actor targeting a given sector or geographical location. When connecting each concept to other concepts in a defined way, the data is given context, and with this additional meaning to a user. In this way we argue that ambiguity in terms and definitions will be reduced.

5.4 Differences in Object Types and Fact Types

There is a difference between objects that may be observed directly, and objects that are a result of human decision or analysis. Example of these types are *incident* and *tool* (not *content* or *hash*). The relationships going to and from these may also imply analysis, like *classifiedAs* and *attributedTo*. These facts are not a directly observable link. The trust we have in the source of these facts is thus more significant.

The differences in meaning of the different fact types shows the importance of semantics. There are object types which have multiple possible fact types connecting them, and where the semantics of the chosen fact type significantly differentiates.

An example of this is *content* $\xrightarrow{connectsTo}$ *URI* and *content* $\xrightarrow{at}$ *URI* as described in Section 4.2.4.

[13] https://github.com/MISP/misp-galaxy/blob/master/clusters/threat-actor.json

5.5 Sharing CTI

Newer publications suggest that still about 78% of shared CTI is unstructured (Sauerwein et al., 2019). Without any structure, we can only automate sharing of *data* as no relationships are present. With the choice of only adding information as facts (relationships) in ACT, we force all CTI to be stored with/as relationships. With this baseline we can automate sharing of triplets which is a significant improvement from sharing data and allows for sharing of graphs.

6 CONCLUSIONS AND FURTHER STUDY

We have proposed a strict data model based on objects and relationships, with the ability to represent available CTI. We have populated it with relevant data, and have identified new information through analysis enabled by the data model. The most prominent results from the data model is data validation, seamless enrichment, excellent analysis capabilities and flexibility of CTI ingest.

Future development of the data model will include hierarchical object types and fact types (using relationships borrowed from ontologies such as *subClassOf* and *subPropertyOf*) which will enable inheritance, more precision and reasoning.

In the implemention of our data model we allow external workers to access the content and add new facts. In this context we are exploring the use of an OWL-implemented version of our data model to infer new facts based on rule based reasoning using Semantic Web Rule Language (SWRL) (W3C, 2004).

ACKNOWLEDGEMENTS

This work was supported by the Research Council of Norway and mnemonic under the ACT and TOCSA projects.

The authors would like to thank the anonymous reviewers for valuable input that improved the paper.

REFERENCES

ANSSI, Luatix, and CERT-EU (2019). The OpenCTI Platform.

Barnum, S. (2012). Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX^{TM}). *MITRE Corporation*, 11.

Caltagirone, S., Pendergast, A., and Betz, C. (2013). The diamond model of intrusion analysis. Technical report, DTIC Document.

Hogan, A., Arenas, M., Mallea, A., and Polleres, A. (2014). Everything you always wanted to know about blank nodes. *Journal of Web Semantics*, 27:42–69.

Mavroeidis, V. and Bromander, S. (2017). Cyber threat intelligence model: an evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence. In *2017 European Intelligence and Security Informatics Conference (EISIC)*, pages 91–98. IEEE.

MISP (2019). The MISP platform.

MITRE (2019). Adversarial Tactics, Techniques and Common Knowledge (ATT&CK). https://attack.mitre.org/.

OASIS CTI TC (2017). Structured threat information expression (STIX^{TM}) 2.0. https://oasis-open.github.io/cti-documentation/.

Polzunov, S. and Abraham, J. (2019). EVALUATE OR DIE TRYING - A Methodology for Qualitative Evaluation of Cyber Threat Intelligence Feeds.

Sauerwein, C., Pekaric, I., Felderer, M., and Breu, R. (2019). An analysis and classification of public information security data sources used in research and practice. *Computers & Security*, 82:140–155.

Sauerwein, C., Sillaber, C., Mussmann, A., and Breu, R. (2017). Threat Intelligence Sharing Platforms: An Exploratory Study of Software Vendors and Research Perspectives.

Stillions, R. (2014). The DML Model. http://ryanstillions.blogspot.com/2014/04/the-dml-model_21.html.

W3C (2004). Semantic Web Rule Language.

W3C (2014). RDF 1.1 semantics.

Wagner, C., Dulaunoy, A., Wagener, G., and Iklody, A. (2016). Misp: The design and implementation of a collaborative threat intelligence sharing platform. In *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*, pages 49–56. ACM.