From Plain Text to CTI – A Technological Solution for Gathering Cyber Threat Intelligence using Natural Language Processing

Robert Müller
Fraunhofer FKIE
GERMANY

robert.mueller@fkie.fraunhofer.de

Prof. Dr. Elmar Padilla
Fraunhofer FKIE
GERMANY

elmar.padilla@fkie.fraunhofer.de

ABSTRACT

A growing community of cybersecurity experts, companies, and hobbyists has evolved, who tend to publicly share their knowledge about cyber threats, gained from malware and attack analyses. In the context of cybersecurity, this type of knowledge is commonly called Cyber Threat Intelligence (CTI). Unfortunately, it is often provided as natural language text, for example as blog articles or PDF reports. To make it usable for automated security measures or information sharing platforms, it has to be converted into machine-readable formats, preferably through a largely automated process. Different approaches have been published to address this problem. However, they are all not truly user-oriented, which is essential in the field of CTI since the declaration of information to the state of intelligence heavily depends on its application in concrete operations. With that gap in mind, we provide a theory-based problem model that reflects domain-specific user requirements resulting from individual definitions of intelligence. We then use this model to implement a technological solution, drawing on a set of strongly tailored technologies, as well as on user-friendly interfaces for direct interaction.

Applied to a set of 2,000 natural language threat reports, our exemplary implementation delivered more than 150,000 CTI objects and relationships, containing 33,389 unique Indicators of Compromise (IoCs). The total recall of our solution was measured at 0.81, while precision was at 0.93. These results show the high potential hidden in the before mentioned publicly available sources, and demonstrate that there are automated measures to reliably uncover it and make it accessible to computer-assisted measures.

1. INTRODUCTION

The increasing global networking of devices and critical infrastructures offers hostile nations or APT groups a multitude of possibilities to harm economy, society and politics through cyberattacks. However, at the same time IT security experts increasingly use similar technologies to counter these threats. A growing community of experts, companies, and private individuals has emerged, who dedicatedly report about cyber threats, offer results of malware analysis, and discuss trends and predictions regarding attacks and attackers. Such information, commonly referred to as *Cyber Threat Intelligence* (CTI), is extraordinary interesting for *cybersecurity analysts* and *Security Operations Centers* (SOCs). However, since big amounts of these sources are provided as natural language text, experts have to take much effort in extracting the relevant information to make it usable for automated security solutions, visualization tools, or standardized sharing platforms. Rapidly growing volumes of data in combination with a continuing shortage of skilled analysts (e.g. as mentioned in [1] & [2]) make this increasingly harder.

In this paper, we present an approach to extensively support the manual analysis process, using concepts from the discourse of *Information Extraction* (IE), as well as technologies from the field of *Natural Language Processing* (NLP). To guarantee a close orientation to the practical context of *cybersecurity analysis*, we start with a view on some basic requirements from that field. In the next step, we describe the problem from the perspective of the IE discourse and derive a problem model. Since the domain of CTI is very specific, the types of entities to be extracted are largely specific as well. We will show, how this view on the problem allows for solving the resulting sub-problems at a high precision by using a variety of tailored NLP technologies. Our contributions to the scientific and technological discourse on extracting CTI from natural language text are the following:

- **C1:** Provide an abstract problem model for CTI extraction based on concepts from Information Extraction (IE), which is open to the specific requirements of concrete user contexts.
- **C2:** Integrate user requirements and knowledge into the CTI extraction process.
- **C3:** Implement and evaluate a solution following the provided problem model to extract a broad variety of CTI entities.

Paper Organization. In section 2, we give an overview over the research in the scope of this paper, in order to position it within the scientific and technological discourse, and to distinguish our approach from existing solutions. Section 3 documents our model of the CTI extraction problem, and section 4 covers the implementation of the approach. The resulting solution is evaluated in section 5, and discussed in section 6. In the final section, we draw our conclusion.

2. RELATED WORK

An extensive overview over the field of extracting CTI from natural language text is provided by Rahman et al. [3]. Their literature review reveals, that most contributions focus on extracting CTI from threat reports, forums, Twitter, and Blogs. Beside that, they found single articles focusing on Version Control Repositories, Logs, and Darknet marketplaces. All of these sources are primarily used for threat event identification, threat topic analysis, IoC extraction, or TTPs extraction. Beside the articles mentioned in [3], several more relevant contributions have been published, which can be classified as follows.

Technology-oriented Approaches. The first type of papers to mention in the context of our work, are studies with a primarily technological focus. Here, the authors‘ motivation is to evaluate the application of recent technologies in the field of CTI extraction. In this category, Ghazi et al. [2], Kim et al. [4], as well as Wang et al. [5] take place, who all train NER models to label CTI elements in natural language text. The same is true for Husari et al. [6] and Zhang et al. [7], who extract threat actions in the form of subject-verb-object triplets – the former by using the metrics of entropy and mutual information, the latter by using a set of five classification features.

Problem-oriented Approaches. The approach documented in our paper differs from all the before mentioned studies in that we start from the perspective of a cybersecurity analyst using the extracted data. Given that background, papers with a more problem-oriented view are of special interest for our work. As the following compilation shows, this commonly encompasses using a set of multiple technologies to cover all the different aspects of CTI extraction. The corresponding amount of papers can be subdivided into the levels of CTI (*tactical, operational, and strategic*) covered by their approaches.

The first subcategory of approaches aims on extracting *tactical CTI,* which is commonly referred to as *Indicators of Compromise* (IoC). Huang et al. [8] have to be mentioned here, who extract domain names from mailing lists by using regular expressions and a Random Forests classifier. Niakanlahiji et al. [9] use the same combination of Random Forest classifier and regular expressions to find Tweets containing CTI and extract a broader range of IoC types (including IP addresses, URLs, hashes etc.). Kim et al. [10] also use regular expressions to extract IoCs, and then use requests to a malware database to enhance their results. Sun et al. [11] take a different route by exploiting the Document Object Model (DOM) of HTML documents, which are chosen by a Machine Learning classifier.

The second subcategory of papers found on our topic are studies that enhance their focus on *operational* and *strategic CTI*. The first studies to mention here are the ones from Koloveas et al. [12], from Satvat et al. [13], and from Husari et al. [14] who all present their solutions to extract primarily operational CTI using different sets of NLP technologies. Li et al. [15] use a set of regular expressions and NLP technologies to extract operational and strategic CTI, including CVEs, victimized devices, device manufacturers, and locations. Gao et al. [16] finally also cover strategic CTI, which is extracted using a NER approach based on a Conditional Random Field (CRF) model. Beside IoCs, they include threat actors, malware, attack patterns, and more.

This basic literature review shows some deficits with regards to the goal of our study. First, all of the before mentioned approaches are not truly user-oriented. In the best cases, requirements are explicitly or implicitly derived from commonly used CTI languages like STIX[1] and their types of possible entities. In none of them though, the user himself is directly included, for example to choose which entities are not only pieces of information but also have the status of intelligence in the specific user context. A second observation is that all of these studies lack an explicitly designed problem model based on theoretical considerations, which is open to the variety of possible entity and the relationship types. Such a model would open the space for more flexible solutions and enhancements concerning different definitions of intelligence. Both of these identified issues founded our intended contributions (C1 - C3, section 1).

3. PROBLEM MODELING

One premise of our approach is that the job of cybersecurity analysts cannot be replaced with fully automated measures. This is due to the reason that building intelligence is strongly bound to the operational environment it will be used in [17]. Thus, environmental factors are central elements for the process of generating intelligence: In a long-term view, *organizational* aspects, such as the core tasks or typical processes of an institution, play a role. In a short-term view, the *situational* context the analyst is positioned in (e.g. the concrete operation or task) is of importance. Respecting these two factors implies three major consequences: (1) a problem model needs to be abstracted to such an extent that it is flexible enough to adapt to the *organizational* context of the user, and to evolve depending on the operational environment. (2) The user must *situationally* be able to interfere in the CTI gathering process. (3) A CTI extraction approach must be seen as a tool the analyst interacts with instead of a stand-alone solution.

An abstract view on the problem of gathering information from natural language text is provided by the scientific discourse of *Information Extraction (IE)*. Although the term IE is often used in terms of a certain technology these days, it basically refers to some problem solving concepts to formalize unstructured text content. These concepts can be implemented with different technologies. According to the often cited model by Jiang [18], IE consists of two basic tasks: *Named Entity Recognition* (NER) and *Relation Extraction* (RE). While NER focuses on identifying text phrases that relate to real world entities, RE reveals semantic relationships between these entities.

The segmentation into NER and RE build the basis for our approach. However, we additionally subdivide the RE task into the tasks of *String Tagging* and *Contextual Classification*. The reason for that is, that a string, which matches a name is not necessarily referring to a CTI-relevant entity. For example, the string "hacking team" can either refer to any hacking group in a generic way, or to the Italian spyware vendor "Hacking Team". And an email address – as another example – can either be the contact address of a document’s publisher, or refer to a spear phishing campaign mentioned in the document.

To resolve such uncertainties, one has to respect context information. This is implicitly done by some NER solutions, which respect details like the terms next to the phrase or part-of-speech and syntactic dependencies. In our problem model however, we want to respect this segmentation explicitly to provide more flexibility for possible implementations. The core assumption behind that is, that the more a professional domain is broken into subtasks, the more specific an implementation can be, and thus the higher its recall and precision will be. Figure 1 shows the resulting main processing steps (blue units).

[1] https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html

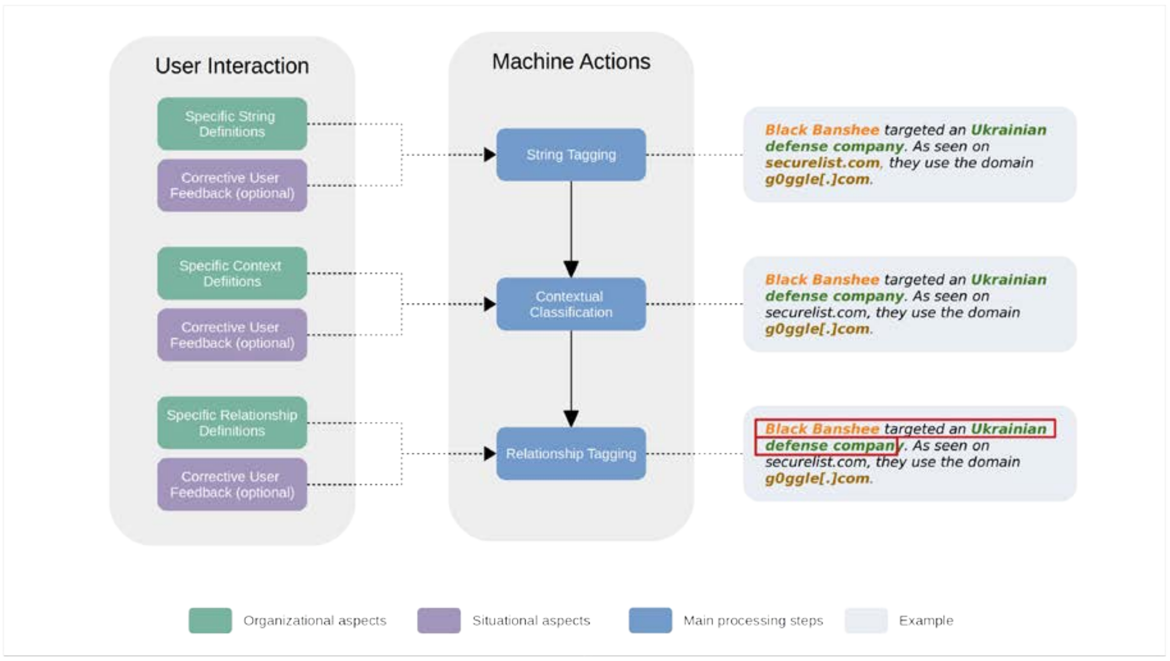

User Interaction
Specific String Definitions
Corrective User Feedback (optional)
Specific Context Defitions
Corrective User Feedback (optional)
Specific Relationship Definitions
Corrective User Feedback (optional)
Machine Actions
String Tagging
Contextual Classification
Relationship Tagging
Black Banshee targeted an Ukrainian defense company. As seen on securelist.com, they use the domain g0ggle[.]com.
Black Banshee targeted an Ukrainian defense company. As seen on securelist.com, they use the domain g0ggle[.]com.
Black Banshee targeted an Ukrainian defense company. As seen on securelist.com, they use the domain g0ggle[.]com.
Organizational aspects
Situational aspects
Main processing steps
Example

Figure 1: Problem model for CTI extraction.

To address the premise of a system interacting with an analyst, the user interaction must be explicitly respected. As mentioned before, two temporal horizons play a role here: First, *organizational aspects* have a long-term influence on the extraction process. They determine which *types* of entities, contexts, and relationships need to be respected. For example, an analyst who wants to draw a picture on the worldwide threat landscape will be focusing on strategic intelligence (e.g. attacker groups, targeted nations, industry sectors, etc.), while a security operator of a private company might have more interest in IoCs. Second, *situational aspects* prescribe which *concrete* entities, contexts, and relationships are respected depending on the current operational context. Of course, there might be use cases with only permanent processing requirements, so that situational aspects hardly play a role. Thus, in our concept they are marked as optional. The left column in figure 1 represents the different types of user interaction.

4. IMPLEMENTATION

We describe our implementation of the before mentioned approach along with the three main processing steps *String Tagging*, *Contextual Classification*, and *Relationship Tagging* (as shown in figure 1). For our exemplary implementation, we chose the set of entity types shown in table 1 (left column).

Table 1: Entity and relationships chosen for exemplary implementation.

Entity Types	Relationship Types
• **Indicator**: Malicious IP addresses, domains, URLs, email addresses, and hash values • **Malware**: Occurrences of malware names • **Tool**: Tools used by attackers • **Attacker Group**: Occurrences of intrusion set names • **Targeted Entity**: Targets of attacks, encompassing organizations, individuals, and technological systems • **Location**: Regions and countries • **Report**: Every report is represented as an entity	• **Targeting Relationship**: Relations between targeted entities and malware or attacker groups • **Usage Relationship**: Relations between attacker groups and the malware families used by them • **Location Relationship**: Regional information about targeted entities • **Report Reference**: All entities will be referenced to the corresponding report entity to ensure backtracking of all information in the resulting data

We chose them with respect to assumptions about their importance to a potential analyst as well as to their likelihood to occur in a textual threat report. For choosing the relationship types, we considered the selection of entity types (table 1, left) on the one hand, and the reasonableness of explicitly formulated occurrences in threat reports on the other hand. The chosen relationship types are shown in table 1 (right).

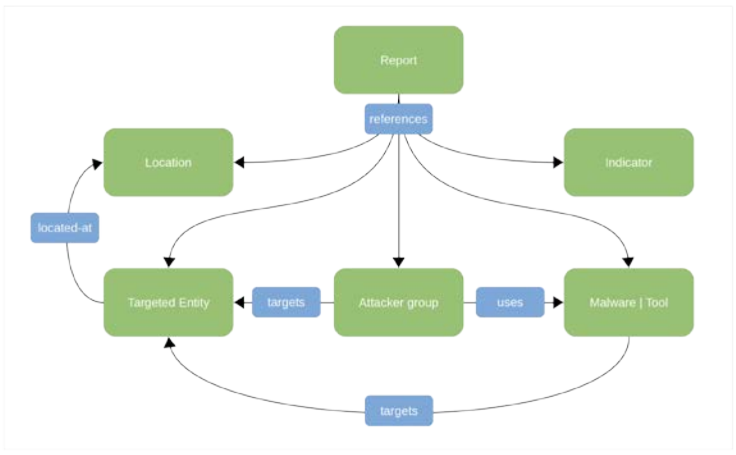


Figure 2: Entity and relationship model for exemplary implementation

Unfortunately, relationships between IoCs and other entities are hardly explicitly formulated. In most reports they are simply listed after the main text. A practical but fuzzy solution to that problem could be, to associate all IoCs of a report to the groups and malware mentioned in the report. However, we won‘t consider this in our implementation since such decisions are highly dependent on the specific user context. Figure 2 shows our resulting entity and relationship model. It can be seen as first organizational requirements given by a fictional user context our implementation takes place in.

4.1. String Tagging

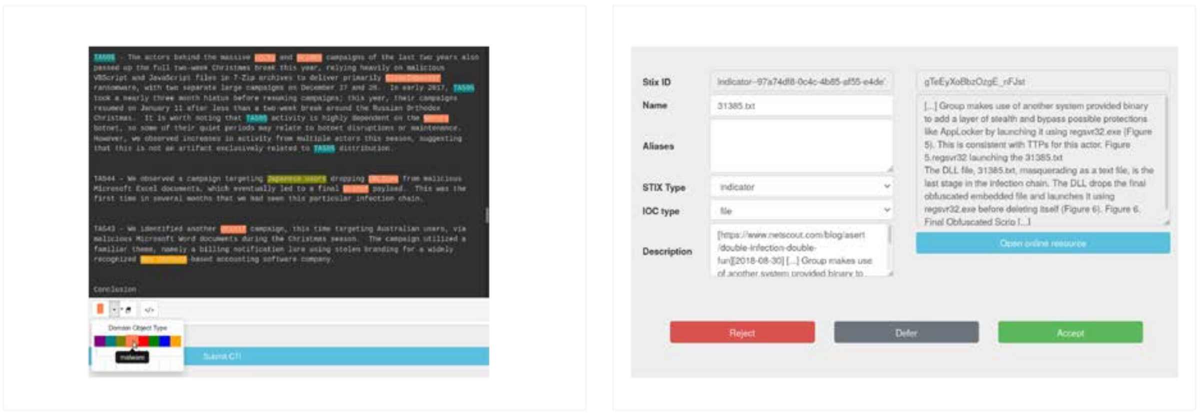
The String Tagging task will probably be the easiest step in most user scenarios. Its goal is to find strings within the document that probably represent entities of interest in the real world.

Organizational aspects. In our solution, the organizational aspect of string tagging is implemented with the following three types of string definitions, each depending on the considered entity types to be tagged:

1. **Name-matching Strings**: Since most names of known attacker groups, malware, tools, and locations, as well as their aliases and spelling variants can be found through public sources, a simple string match is the fastest, easiest and most precise way to find occurrences in text documents. As name resource for our implementation, we used the well-known MISP Galaxy Clusters[2] that are maintained on a regular basis.
2. **RE-matching Strings**: IoCs, and particularly IP addresses, domains, URLs, email addresses, and hash values, have distinct character structures, which differ extensively from natural language words. Thus, they can easily be detected by using regular expressions.
3. **Verb-related Strings**: Targeted entities, like nations or industry sectors, do not match to predefined names or regular expressions. However, they embody a certain semantic role in a natural language sentence, which is highly determined by the verb they are syntactically dependent on. Thus we implemented a mechanism to syntactically analyze all of the document's sentences (using spaCy[3]), then match their verbs with a predefined set of words (that are semantically close to the verb "target"), and extract all objects syntactically related to these verbs.

All of these three mechanisms differ from established NER solutions, which are predominantly based on statistical methods. Since the professional domain of CTI is highly specific, and statistical NER methods are known to be largely domain dependent, the above mentioned approaches a presumably way more effective. Initial attempts to go in that direction, as well as the results from other research (e.g. [2]) confirm this assumption.

Situational aspects. To respect the situational aspects of the user context, we implemented a web-based editor, which displays the plain text of a document with colored tags as found by the system, and provides tools to tag further strings or remove tags made by the system (figure 3, left).


TA505 – The actors behind the massive [illegible] and [illegible] campaigns of the last two years also passed up the full two-week Christmas break this year, relying heavily on malicious VBScript and JavaScript files in 7-Zip archives to deliver primarily [illegible] ransomware, with two separate large campaigns on December 27 and 28. In early 2017, TA505 took a nearly three month hiatus before resuming campaigns; this year, their campaigns resumed on January 11 after less than a two-week break around the Russian Orthodox Christmas. It is worth noting that TA505 activity is highly dependent on the [illegible] botnet, so some of their quiet periods may relate to botnet disruptions or maintenance. However, we observed increases in activity from multiple actors this season, suggesting that this is not an artifact exclusively related to TA505 distribution.

TA544 – We observed a campaign targeting Japanese users dropping [illegible] from malicious Microsoft Excel documents, which eventually led to a final [illegible] payload. This was the first time in several months that we had seen this particular infection chain.

TA543 – We identified another [illegible] campaign, this time targeting Australian users, via malicious Microsoft Word documents during the Christmas season. The campaign utilized a familiar theme, namely a billing notification lure using stolen branding for a widely recognized [illegible]-based accounting software company.

Conclusion

Stix ID: indicator--97a74df8-0c4c-4b85-af55-e4de
Name: 31385.txt
Aliases:
STIX Type: indicator
IOC type: file
Description: [https://www.netscout.com/blog/asert/double-infection-double-fun][2018-08-30] [...] Group makes use of another system provided binary to

gTeEyXoBbzOzgE_nFJst

[...] Group makes use of another system provided binary to add a layer of stealth and bypass possible protections like AppLocker by launching it using regsvr32.exe (Figure 5). This is consistent with TTPs for this actor. Figure 5.regsvr32 launching the 31385.txt
The DLL file, 31385.txt, masquerading as a text file, is the last stage in the infection chain. The DLL drops the final obfuscated embedded file and launches it using regsvr32.exe before deleting itself (Figure 6). Figure 6. Final Obfuscated Scrip [...]

Figure 3: User interface. Left: manual *String Tagging*, right: manual *Contextual Classification.*

4.2. Contextual Classification

Even if a string matches a predefined string definition, it doesn't mean that it refers to an entity of the class the string definition was aiming at. In many cases, contextual knowledge is needed. For example, can a mentioned email address either be part of a phishing campaign, or it can be a contact address of the document's publisher. But context-based classification can even go beyond this binary decision. It can for

[2] https://github.com/MISP/misp-galaxy/tree/main/clusters

[3] https://spacy.io/

example be used to determine a victim's industry sector, or help to distinguish between a malware and an attacker group with the same name.

Organizational aspects. For the organizational view on the task of Contextual Classification, the following rule sets represent three different contexts:

1. **Proper-noun Contexts:** This rule set is applied to all entity candidates that were identified by a simple name string match (malware, attacker group, tool, location). Primarily, the system checks whether the string also matches an English word. For example the string "lead" can either refer to the attacker group "LEAD" or to the English word "lead". To decide such cases, the string is checked for capital letters.
2. **Indicator Contexts**: This rule set is built for indicator entities. First, we check whether the indicator candidate is an IP address or hash value, since entities of these types are almost always malicious when occurring in threat reports. Further, the rule set encompasses a blacklist check, a check whether the indicator is part of a list, and whether there are any measures to disarm a domain or URL (e.g. hxxp://domain.com). In case of an IP address candidate, we try to distinguish it from four-digits version numbers of software products (e.g. 'version 1.1.3.4').
3. **Targeted-identity Contexts**: The check for targeted identity candidates is rather sophisticated. First, we apply some filtering rules. We start with checking whether the candidate is part of a positive statement, which means that the surrounding statement is neither a question nor negated. Then we check if a malware's or attacker group‘s name is included within the string, which is a sure sign that it is not a victim. The third filtering operation uses a set of predefined strings, which were found to be strong indicators of non-identity entities. After the filtering, we start with a set of positive checks. Whenever one of these is positive, the whole check is finished and assessed positive. First, we check whether the syntactical subject of the surrounding statement is an attacker group. Then we check the entity‘s string as well as the surrounding tokens against a set of predefined strings, that strongly indicate a targeted victim (e.g. "compromised", "targeted" etc.). Finally we check if there are any signs of an industry sector within the string.

In a further step, we try to classify all targeted identities into industry sectors. This is simply done by checking the string for predefined substrings. These substrings were empirically derived from a set of 100 examples, which appeared to be enough for a solid degree of theoretical saturation for the string set.

Situational aspects. Our implementation respects the requirement of situational interaction with the user through a small interface that allows the user to manually classify entities, which could not finally be classified through the organizational definitions. The user has the opportunity to decline or accept an entity candidate, as well as to optionally change its parameters (figure 3, right).

4.3. Relationship Tagging

All entities that could be classified are finally checked for relationships with other entities. Relationship types to be respected in our implementation are "uses", "targets" and "located-at" (figure 2). Relationships of type "references" between reports and the entities contained in them are automatically built and will not be outlined further in this paper.

Organizational aspects. The organizational definitions of Relationship Tagging are made up as follows:

1. **Syntax-based Relationships**: Relationships of type "targets" and "uses" (figure 2) are identified by using the syntax of the document‘s sentences. For example, for every malware entity its surrounding sentence is investigated on whether the entity is object of a statement and whether the subject of this statement is an attacker group entity. This constellation indicates with a high

probability that the attacker group is using the malware in some way. The same is true for every other possible "uses" or "targets" relationship shown in figure 2.

2. **Embedded-entity Relationships**: In our implementation, relationships of type "located-at" only occur between targeted identities and location entities (figure 2). Our organizational definition is simply given by a check whether a location entity is embedded in the string of a targeted identity. For example, the identity "government of Italy" encompasses the location entity "Italy". Thus, we draw a relationship between them.

Situational aspects. To give the user an opportunity to influence the drawing of relationships, we enhanced the editor shown in figure 3 (left) with some tools to remove existing relationships and add new ones (figure 4). To do so, the user has to select a piece of text containing the two entities that he wants to relate. Then he chooses the type of relationship from a drop-down menu, and the relationship will appear on the list on the left side. Figure 4 shows an example. The red line represents the relationship selected on the list on the left side.

Figure 4: User interface for manual *Relationship Tagging*

4.4. System Architecture

The whole system architecture is shown in figure 5. Original documents, tagged plaintexts, and CTI objects are stored in an *Elasticsearch*[4] instance. The processing steps are represented by a set of microservices (blue). The same is true for some additional preprocessing and postprocessing steps (green units). The preprocessing module prepares the plain text and extracts some metadata, the meta-analyzing module performs some cross-document analysis, and the extraction module is responsible for converting the recognized entities and relationships into STIX objects. All microservices are coordinated by a message broker that observes the state of the documents and for each state triggers the next microservice.

[4] https://elastic.co

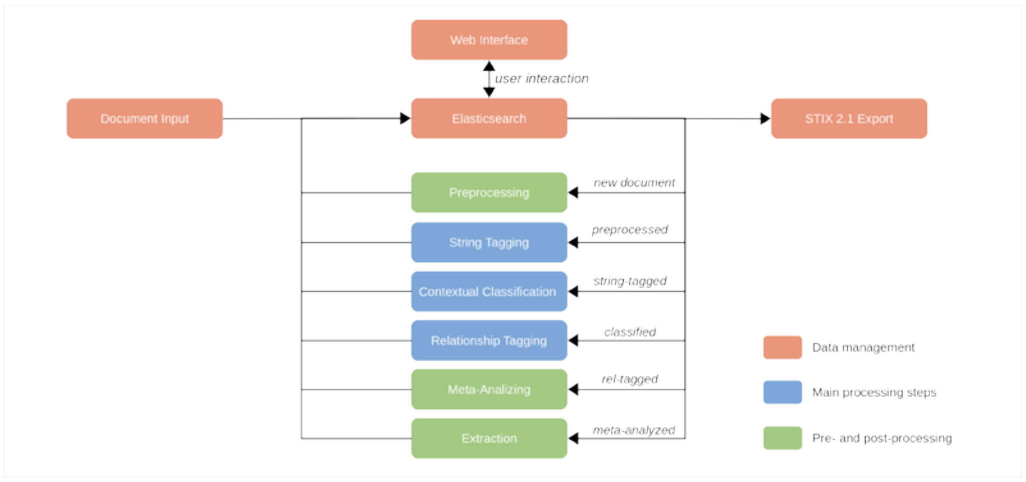


Figure 5: Microservice architecture of our CTI extraction implementation.

5. EVALUATION

Processing Results. To get an impression of the performance of the approach, we parsed 2,000 textual reports taken from the malware database Malpedia[5]. We only respected *organizational aspects*, which means that there has been no user interaction to cover *situational aspects* during the processing. Figure 6 (left) shows the distribution of all 154,317 extracted entities.

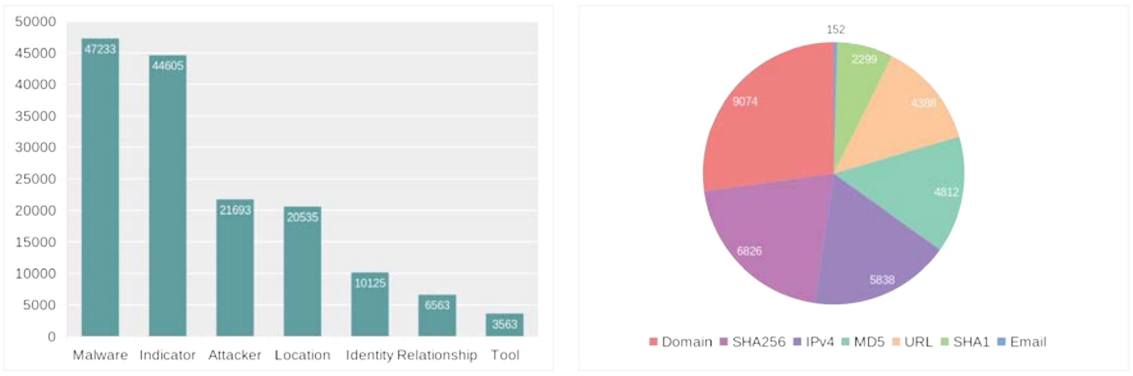


Figure 6: Distribution of objects extracted from 2,000 textual documents. Left: Type distribution of 154,317 extracted entities; right: Type distribution of 33,389 *unique* IoCs

Of course, there are multiple occurrences of almost every real-life entity in the parsed documents. This fact is of particular importance in practical user contexts of IoCs, since these are often used without relationships or document references (e.g. in Firewalls or Intrusion Detection Systems). That means the effective number of IoCs is below the counted instances shown in figure 6 (left). With regards to this, in figure 6 (right) we list the distribution of *unique* IoCs, which make up 33,389 entities in total.

Precision and Recall. To evaluate the quality of the extracted data, we manually parsed 20 threat reports, which were chosen from the stock of the before mentioned 2,000 documents. Considering the very heterogeneous distribution of different data types (figure 6, left), we decided to only choose documents that

[5] https://malpedia.caad.fkie.fraunhofer.de/

each contain at least (a) one entity that would be found by name-matching string tagging (malware, attacker, location), (b) one found by RE-matching string tagging (indicator), and (c) one found by verb-related string tagging (identity). Furthermore, at least one relationship between two entities must be explicitly articulated. The idea behind this is to cover as many mechanisms of the approach as possible. From all documents fulfilling these conditions, we randomly chose 20 documents to process them.

During the manual tagging of targeted identities, and their relationships to malware and attackers, we only respected occurrences where the corresponding circumstances were *explicitly formulated* in a sentence. Other cases, where the reader could derive a relationship or an entity's role as victim by *respecting previous parts of the document*, were left out. This is due to the reason that our implementation is not designed to capture such cases. In total, we made 2,516 manual annotations. The resulting recall and precision values, distributed over the various entity types, are shown in figure 7.

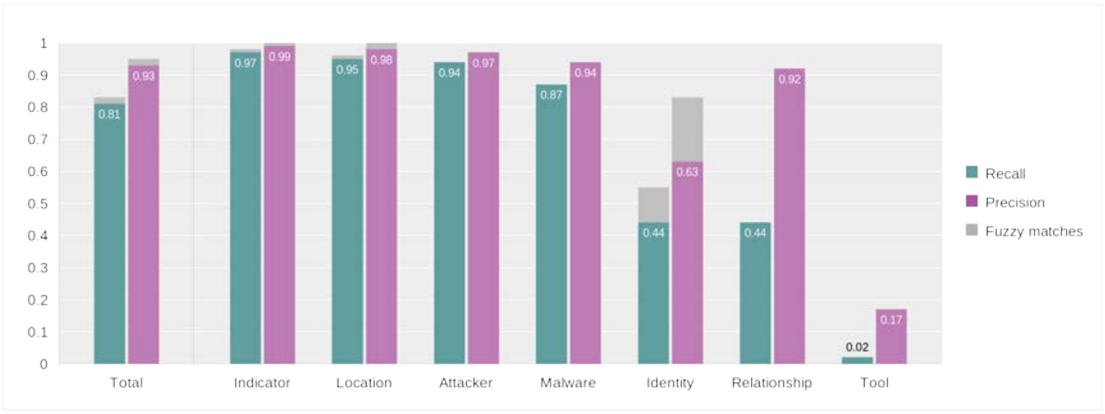


Figure 7: Recall and precision broken down by entity types.

First thing to notice here are the high recall (0.87 - 0.97) and precision (0.94 - 0.99) values of *indicators*, *locations*, *attackers*, and *malware*. All of these entity types were tagged by using name matching and regular expressions. However, that did not work at all for the *tools* entity type. The reason for that is, that our source for tool names[6] differed greatly from the definition of tools we had in mind. For example, *Powershell* as a frequently mentioned tool, was not respected in our source. However, in the future this can easily be resolved by choosing a better fitting resource for tool names, or by building an own one. The second thing that catches one's eyes is the recall of targeted identities, which is below 0.5. One thing to consider here is, that for a noteworthy amount of missed occurrences, simply the start or ending of the tag did not perfectly match. These cases are what we call *fuzzy matches* in figure 7. Respecting these partly mismatching tags, the recall rises up to 0.55 and precision reaches 0.82 (grey bars). Many of the still remaining missing cases result from syntactically complicated statements like passive constructions. The same is true for many of the missed relationships, which have a recall below 0.5 as well.

6. DISCUSSION

Discussion of Contributions. The first claimed contribution of this paper (C1) was to provide an abstract problem model derived from concepts of the IE discourse that can be used as a basis for concrete implementations respecting specific user contexts. In section 3, we presented our proposal to meet these requirements (figure 1). We then presented an implementation of the model, based on some assumptions about a fictional user context. It could be shown, that the problem model is flexible enough to respect the

[6] https://raw.githubusercontent.com/MISP/misp-galaxy/main/clusters/mitre-tool.json

different context-derived data types (e.g. indicators, malware etc.) by using different strongly tailored technologies (string matching, regular expressions, context-related rules etc.).

As a second contribution, we claimed to define how to integrate user requirements and knowledge into the CTI extraction process. This is likewise covered by the model shown in figure 1. It explicitly includes the more long-term *organizational aspects*, as well the short-term *situational aspects* of direct user interaction. Our concrete implementation shows, how the organizational aspects can be covered by *algorithmic definitions*, while the situational aspects need some sort of *user interface*. Of course, there are conceivable user settings, where no direct user interaction is needed. For example, in a more mass data-oriented workflow, an analyst could rely on a solution automatically extracting as much information as possible fulfilling organizational requirements. Later, in a concrete operation, he explores the collected data for information that is of particular interest in the context of a specific task.

The third contribution provided by this paper is the concrete implementation documented in section 4. Our evaluation in section 5 demonstrates its high potential to make use of publicly available sources providing CTI in the form of natural language text. In total, 154,317 information units that are relevant to the fictional user context could be identified. The extracted units contained 33,389 unique indicators, consisting of 41.7 % hash values (SHA256, MD5 SHA1), and 58.3 % system and resource references (domains, IP addresses, URLs, email addresses). With a precision of 0.93, most of the extracted entities can be considered largely reliable. An outlier from this is the tool entity type, which underlines the importance of entity definitions to be strongly tailored to the operational context.

Limitations. Of course, our problem model as well as our implementation come with limitations. The first thing to notice is that the problem model is settled on a high abstraction level, which lets the user alone with more concrete decisions on the way to an implementation. The next limitation to mention is the fact, that our implementation is based on a fictional user setting without any empirical data but only based on assumptions about a typical cybersecurity analysis context. Furthermore, the chosen technologies for the different implementation steps have their limitations as well. First, the recall of entity types that are detected by name matching, are completely dependent on available names dictionaries and their update cycles. Second, regular expressions often cannot respect all exceptions found in natural language text (e.g. unusual line breaks or rare characters). And finally, recall and precision of entities that are found by using a sentence‘s syntax are dependent of the precision of the applied NLP package as well as of the plethora of exceptions that can be found in a natural language text.

7. CONCLUSION

The growing amount of freely available CTI given as natural language text documents makes it increasingly harder to leverage the full potential of this resource. Hence, in this paper we introduced a theory-based problem model and a user-oriented solution to assist cybersecurity analysts to cope with this challenge. The model explicitly provides aspects of organizational and situational user requirements to help the developer of an implementation to integrate the requirements and knowledge of a domain-specific user into the extraction process.

Our solution is a concrete implementation of this model, and is based on assumptions about a fictional user context. We used different technologies that are specifically tailored to each of the types of entities to be extracted. Our assumption behind that is, that this allows for a higher degree of recall and precision than using the same technology for every type of extracted information unit. In the evaluation of our implementation, we demonstrated the high potential of the approach for leveraging publicly available text resources to gain actionable CTI. The total recall was measured at 0.81. A total precision of 0.93 suggests that most of the extracted objects are largely reliable. Since we did not include user feedback and specially crafted parameters for our approach in the evaluation, the results can be seen as a lower border. The approach itself offers multiple opportunities for future work and optimization, for example the aforementioned parameterization and user feedback.

REFERENCES

[1] M. S. Abu, S. R. Selamat, A. Ariffin und R. Yusof, „Cyber threat intelligence--issue and challenges," *Indonesian Journal of Electrical Engineering and Computer Science,* Bd. 10, pp. 371-379, 2018.

[2] Y. Ghazi, Z. Anwar, R. Mumtaz, S. Saleem und A. Tahir, „A supervised machine learning based approach for automatically extracting high-level threat intelligence from unstructured sources," in *2018 International Conference on Frontiers of Information Technology (FIT)*, 2018.

[3] M. R. Rahman, R. Mahdavi-Hezaveh und L. Williams, „A Literature Review on Mining Cyberthreat Intelligence from Unstructured Texts," in *2020 International Conference on Data Mining Workshops*, 2020.

[4] G. Kim, C. Lee, J. Jo und H. Lim, „Automatic extraction of named entities of cyber threats using a deep Bi-LSTM-CRF network," *International journal of machine learning and cybernetics,* Bd. 11, pp. 2341-2355, 2020.

[5] X. Wang, Z. Xiong, X. Du, J. Jiang, Z. Jiang und M. Xiong, „NER in Threat Intelligence Domain with TSFL," in *CCF International Conference on NLP and Chinese Computing*, 2020

[6] G. Husari, X. Niu, B. Chu und E. Al-Shaer, „Using entropy and mutual information to extract threat actions from cyber threat intelligence," in *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2018.

[7] H. Zhang, G. Shen, C. Guo, Y. Cui und C. Jiang, „EX-Action: Automatically Extracting Threat Actions from Cyber Threat Intelligence Report Based on Multimodal Learning," *Security and Communication Networks,* Bd. 2021, 2021.

[8] C. Huang, S. Hao, L. Invernizzi, J. Liu, Y. Fang, C. Kruegel und G. Vigna, „Gossip: Automatically identifying malicious domains from mailing list discussions," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017.

[9] A. Niakanlahiji, L. Safarnejad, R. Harper und B.-T. Chu, „Iocminer: Automatic extraction of indicators of compromise from twitter," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019.

[10] D. Kim und H. K. Kim, „Automated dataset generation system for collaborative research of cyber threat analysis," *Security and Communication Networks,* Bd. 2019, 2019.

[11] T. Sun, P. Yang, M. Li und S. Liao, „An Automatic Generation Approach of the Cyber Threat Intelligence Records Based on Multi-Source Information Fusion," *Future Internet*, Bd. 13, p. 40, 2021.

[12] P. Koloveas, T. Chantzios, S. Alevizopoulou, S. Skiadopoulos und C. Tryfonopoulos, „inTIME: A Machine Learning-Based Framework for Gathering and Leveraging Web Data to Cyber-Threat Intelligence," *Electronics,* Bd. 10, p. 818, 2021.

[13] K. Satvat, R. Gjomemo und V. N. Venkatakrishnan, „EXTRACTOR: Extracting Attack Behavior from Threat Reports," *arXiv preprint arXiv:2104.08618,* 2021.

[14] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu und X. Niu, „Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources," in *Proceedings of the 33rd Annual Computer*

Security Applications Conference, 2017.

[15] K. Li, H. Wen, H. Li, H. Zhu und L. Sun, „Security OSIF: Toward automatic discovery and analysis of event based cyber threat intelligence," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation*, 2018.

[16] P. Gao, X. Liu, E. Choi, B. Soman, C. Mishra, K. Farris und D. Song, „A System for Automated Open-Source Threat Intelligence Gathering and Management," in *Proceedings of the 2021 International Conference on Management of Data*, 2021.

[17] J. C. of Staff, „Joint Publication 2-0: Joint Intelligence," 2013.

[18] J. Jiang, „Information extraction from text," in *Mining text data*, Springer, 2012, pp. 11-41.