ThreatCrawl: A BERT-based Focused Crawler for the Cybersecurity Domain

Philipp Kuehn*
Mike Schmidt
Markus Bayer
Christian Reuter
kuehn@peasec.tu-darmstadt.de
mike.schmidt@stud.tu-darmstadt.de
bayer@peasec.tu-darmstadt.de
reuter@peasec.tu-darmstadt.de
Science and Technology for Peace and Security (PEASEC), Technical University of Darmstadt
Darmstadt, Germany

ABSTRACT

Publicly available information contains valuable information for Cyber Threat Intelligence (CTI). This can be used to prevent attacks that have already taken place on other systems. Ideally, only the initial attack succeeds and all subsequent ones are detected and stopped. But while there are different standards to exchange this information, a lot of it is shared in articles or blog posts in non-standardized ways. Manually scanning through multiple online portals and news pages to discover new threats and extracting them is a time-consuming task. To automize parts of this scanning process, multiple papers propose extractors that use Natural Language Processing (NLP) to extract Indicators of Compromise (IOCs) from documents. However, while this already solves the problem of extracting the information out of documents, the search for these documents is rarely considered. In this paper, a new focused crawler is proposed called ThreatCrawl, which uses Bidirectional Encoder Representations from Transformers (BERT)-based models to classify documents and adapt its crawling path dynamically. While ThreatCrawl has difficulties to classify the specific type of Open Source Intelligence (OSINT) named in texts, *e.g.*, IOC content, it can successfully find relevant documents and modify its path accordingly. It yields harvest rates of up to 52%, which are, to the best of our knowledge, better than the current state of the art. The results and source code will be made publicly available upon acceptance.

KEYWORDS

focused crawling, security, classification

1 INTRODUCTION

In the field of Cyber Threat Intelligence (CTI), which is a sub-field of Open Source Intelligence (OSINT), publicly available information is used to learn from current threats and prevent attackers from using similar Tactics, Techniques & Procedures (TTPs) against other infrastructures. To do this, Indicators of Compromise (IOCs), *e.g.* IP addresses, are shared. Those IOCs can then be transformed into detection rules for Intrusion Detection Systems (IDSs) to detect and therefore eliminate threats before they can deal any damage [7].

Standards to exchange IOCs and other CTI-related information like MISP[1] [25] or STIX[2] can already be used to publish, import, and export those into CTI databases and appliances. Unfortunately, not all information gets published using those standards. CTI is often shared in unstructured ways like blog posts or threat reports from security companies or experts [13, 26] and scanning through online posts and extracting IOCs manually is time-consuming. Hinchy [12] surveyed 468 full-time security analysts about their work. 66% stated they spend over half their time on tedious manual work, 64% believe that more than half of their tasks can be automated, and 64% stated that they are likely to switch jobs in the next year if there are no modern and automated tools to support them. This only confirms that developing tools to automate tasks should be a number one priority in the field of CTI. Otherwise companies can lose valuable experts which reduces their overall level of security.

Different methods in this regard are already being developed and experimented with. IOC extractors [21] and threat detection methods [19] are are one example, which automatically extract IOCs out of usual text. Others are common vulnerability scoring system (CVSS) predictors based on open data sources [10, 18]. But to make use of these techniques, appropriate sources need to be used. Unfortunately, the task of finding those is rarely discussed in this research area. Usually some specific websites or security blogs are getting selected and then crawled to receive posts to extract the IOCs from, which leads to limited tools that only use a very small percentage of all the available information. Since the internet is a very dynamic environment, websites often change, stop publishing new content and new websites get added constantly. New sources of CTI would have to be added by researching them or through recommendations and the already used sources have to be reevaluated regularly to check, if they still provide new CTI-relevant information.

Goal. Maintaining an up-to-date set of CTI-relevant websites requires a lot of manual and tedious work. This leads to a lack of knowledge about personally unknown, but currently active, attack campaigns. Additionally the manual work leads to less effective and motivated employees, since they have to invest huge amounts of

*Corresponding author

[1] https://www.misp-project.org/
[2] https://oasis-open.github.io/cti-documentation/stix/intro

time to search for CTI. Those two problems ultimately result in less infrastructure security. To avoid this, a new and precise approach to find CTI-relevant websites in a fast and efficient way, based on a focused crawling technique, needs to be developed. Focused crawlers "[...] selectively seek out pages that are relevant to a pre-defined set of topics" Chakrabarti et al. [6]. Since this tactic avoids irrelevant regions of the web, it saves hardware as well as network resources. This work focuses on those unstructured ways to distribute and share CTI-related information, published in the surface web [8].

Therefore, this work aims to answer the following research question *"How can CTI relevant information be automatically crawled from the clear surface web while adating the crawling path iterativly?"* ***(RQ)*** Based on this main question, we derive the following sub-questions: *how can websites in the CTI domain be efficiently crawled and ranked?* ***(RQ1)***, *how can websites be categorized automatically as "CTI-relevant"?* ***(RQ2)***, and *how can those two parts be combined to get an ordered list of CTI-relevant websites that is as precise as possible?* ***(RQ3)***

Contributions. This work proposes a focused crawling pipeline called ThreatCrawl which combines different techniques of crawling, classifying, and ranking content. It enables the gathering of specific CTI-relevant information in the surface web. The presented crawler uses embedding techniques to create document embeddings of websites during the crawling process to decide, whether sources should be followed **(C1)**. Such documents are classified according to the type of information and are ranked according to their suitability in the field of CTI **(C2)**. The result of this work can be used as a foundation to create fully automated solutions to extract up-to-date CTIs-relevant information. This could then be leveraged, for example, to create attacker signatures and block them without manual work **(C3)**.

Outline. Initially, an overview over current works of legal issues in crawling, crawling in general, and different approaches to classify websites (*cf.* § 2). The theoretical concept of ThreatCrawl is shown in § 3. § 4 discusses different implementation details. § 5 shows and discusses different metrics to describe the efficiency of the classifier and the crawler as a whole. § 6 concludes this work with an overview over the results and findings as well as their implications and future work.

2 RELATED WORK

This section presents research in the fields of document classification and focused crawling, and outlines the research gap.

2.1 Document classification

For the classification we aim at using document embeddings. Different work in this direction has already been presented. Peters et al. [24] and Devlin et al. [9] propose context-aware word embeddings, *i.e.*, embeddings are generated based on their surrounding context. Peters et al. [24] handle Out-of-Vocabulary (OOV) cases by using character convolutions, while Devlin et al. [9] recursively split tokens up into sub-tokens until each token is recognized. Devlin et al. [9] use Bidirectional Encoder Representations from Transformers (BERT) that can be trained for specific domains and are often used for a variety of Natural Language Processing (NLP) tasks due to their flexibility. Reimers and Gurevych [28] build upon BERT by proposing a method to generate sentence embeddings, called Sentence-BERT (S-BERT). The authors fine-tune the model to create more optimized sentence embeddings. One work leveraging S-BERT to classify web content is proposed by Tawil and Alqaraleh [32].

2.2 Focused Crawling & Threat Extraction

Wang et al. [36] present a Term Frequency-Inverse Document Frequency (TF-IDF) algorithm in combination with a Naive Bayes classifier to find pages relevant to a specific topic. Liu et al. [22] extend this work and use a semantic disambiguation graph to remove ambiguation terms that are irrelevant to the topic. They then use a semantic vector space model to compute the similarity between texts, using TF-IDF as weights. A work searching for data lakes is presented by Zhang et al. [37]. It aims to fill the gap between datasets, annotated to be found by the Google Dataset Search and openly available datasets missing this metadata. Koloveas et al. [14] propose a Machine Learning (ML)-based integrated framework for acquiring, analysing, managing, and sharing CTI-related information. For the present work, especially the part of crawling and classifying is of interest. The authors propose a two-step approach, first, a very broad crawl and later a filtering step for the relevant content. For relevance classification Gensim [27] is used, an open-source implementation of Word2Vec [23]. When classifying a downloaded document, all words present in the topic vocabulary are extracted, embedded, and summed up. The resulting vector is then compared to the topic vectors and a cosine similarity between those is computed to receive a relevance score.

Other proposals in CTI research suggest to focus on Twitter, since it is an easy access to cyber threat information [29, 31, 33, 34]. Others use a set of pre-known websites [21]. While both ways result in valid studies, they are based on pre-known information, either in forms of twitter-handles or websites, without the possibility to widen the view

2.3 Research Gap

Current works that combine web crawling, classifying, and page ranking techniques for CTI into a single pipeline are rare. Tawil and Alqaraleh [32] describe a crawler that uses embeddings (S-BERT) to assign labels to downloaded documents. With this approach crawling and classifying are two fully independent processes, *i.e.*, everything gets crawled and is classified only afterwards. Koloveas et al. [14] present another two-step approach for crawling and classifying their documents. The difference in their approach is, that they use a focused crawler, which filters certain documents, but still performs a very broad search. This leads to a very low relevance rate ($\approx 1\%$) of downloaded websites [15]. Hence, the actual benefits of focused crawling (saving hardware and network resources) are diminished. It should be mentioned that Koloveas et al. [14] also implement additional crawlers that are specifically tailored for fixed websites. Thus, an open question is the development of a generalized one-step focused crawling approach, that combines the crawling, classifying, and ranking of CTI-relevant content.

	Lee	Brown and Lee	Brown and Stirparo
#1	IOCs	Information about vulnerabilities being targeted by attackers	Detailed information about malware being used in attacks
#2	TTPs	Detailed information about malware being used in attacks	Information about vulnerabilities being targeted by attackers
#3	Strategic analysis of the adversary	Indicators of Compromise (IOCs)	Broad information about attacker trends

Table 1: Top 3 most useful Cyber Threat Intelligence (CTI) types according to SANS CTI survey [4, 5, 20].

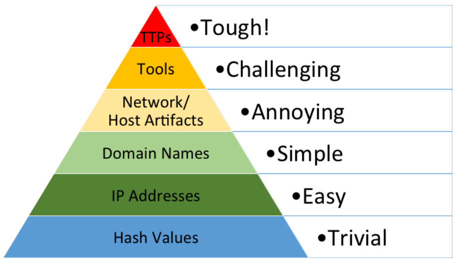

Figure 1: Pyramid of Pain by Bianco [3]

3 THREATCRAWL

This section presents the architecture for ThreatCrawl. First, we gloss the abstract concept of relevance, followed by legal issues when dealing with web crawling. The architectural design decisions are presented afterwards, followed by the classifications process and the crawler seed. Details concerning the implementation and datasets are shown in § 4.

3.1 Relevance

Research is trying to capture a definition of CTI relevance to empirically support automated CTI research [17]. CTI can be defined as "the set of data collected, assessed and applied regarding security threats, threat actors, exploits, malware, vulnerabilities and compromise indicators" [30].

Likewise, Bianco [3] defines the Pyramid of Pain (PoP) (*cf.* Fig. 1), where ranks and IOC-types correspond with the effectiveness on preventing attacks. While hash values are very easy to collect and detect, their effectiveness is also very limited. TTPs on the other hand are hard to collect and detect, but being able to identify them is very effective to detect a certain attacker.

The SANS Institute[3] conducts an annual CTI survey where they interview security professionals from various organizations. The top 3 most valuable and useful CTI types are shown in Tab. 1. In 2020 the most useful type was IOCs. In 2021 those were only on 3^{rd} place and in 2022 they were moved to 5^{th} place. Instead of this very specific CTI type, more general types moved into the most useful types, having "Broad information about attacker trends" in 3^{rd} place in 2022. Looking again at the PoP, the focus is shifting from the lower levels towards the upper levels of the pyramid.

When testing different approaches, IOCs turned out to be bad indicators. While just detecting those kind of information can be done very easily with regular expressions, IOCs like email addresses are often not related to any kind of attacker and hence are not relevant in this context. Therefore, the top 4 most useful types of CTI of the latest SANS survey will be used in this work, namely:

(1) Detailed information about used malware in attacks (*Malware used*)
(2) Information about targeted vulnerabilities by attackers (*Vulnerability targeted*)
(3) Broad information about attacker trends (*Broad information*)
(4) *Tactics, Techniques & Procedures*

For the context of this work, every website that contains information belonging to one or more of the above types is considered CTI-relevant.

3.2 Legal Requirements

When crawling the internet, legal requirements must be followed. Krotov et al. [16] discuss in their paper different aspects of crawling regarding legality and ethics. The authors criticize that despite a lot of research in that area, legal and ethical issues often get ignored. In terms of legality, web scraping is still a grey area. Websites can define *terms of use*, but since users need to agree to them specifically, this does not really apply to automated tools. Other legal issues include the use and processing of copyrighted material and the purpose of web scraping. While a lot of work can be categorized as "fair use", any fraudulent or prohibited use of the collected data is usually still illegal by law. In terms of ethics, the privacy of users appearing in the collected data should be respected. User-generated content like reviews is being collected without considering the implications for those users. Additionally, crawling should not be used to create products that directly compete with the content that was crawled from other sources.

To fully comply to all terms of use, a crawler first would have to locate, download, and analyze them, which results in a incredibly complex task, considering that each owner can define those differently. To still enable website administrators to block unwanted crawlers, the Robots Exclusion Protocol was developed[4]. Our implementation should respect this protocol and act as defensively as possible.

3.3 Architecture

While there are different crawlers available, like the ACHE Focused Crawler[5] Koloveas et al. [14] or BERT based Topic-Specific Crawler [32], none of those offer to use a custom embedding mechanism for documents. Hence, we opt for a new architecture called ThreatCrawl, which implements a focused crawler in one step rather the presented two-step approaches [14, 32]. Fig. 2 depicts the basic design of the developed solution. ThreatCrawl is divided into five logical modules, namely the storage, retriever, extractor, classifier, and monitor.

[3]https://www.sans.org

[4]https://www.rfc-editor.org/rfc/rfc9309.html

[5]https://github.com/VIDA-NYU/ache/

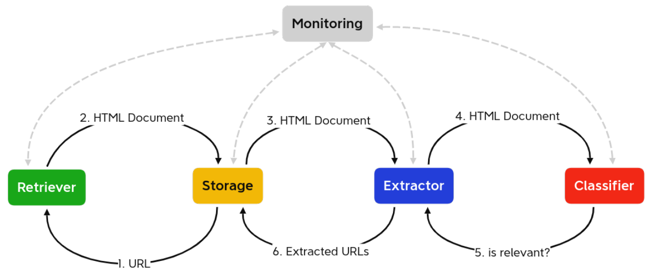

Figure 2: The architecture

Storage. The storage module contains the Uniform Resource Locator (URL) queue, crawled URLs, the robots.txt database, domain timers, and databases for the processed and unprocessed HyperText Markup Language (HTML) files. The URL queue contains all URLs that need to be requested. The retriever processes this queue while the extractor puts new ones into the queue. All URLs that have been visited will be saved in this list to avoid multiple and unnecessary requests to the same resource.

The robots.txt database manages all tasks related to "robots.txt" files that the retriever downloads from different domains, including access rules and crawl-delays. If no crawl-delay is defined, a default is used to ensure a defensive crawling process. Additionally, this component enforces the access rules defined by the crawled server in the robots.txt.

Lastly, the storage module saves unprocessed and processed HTML documents that are requested by the retriever module and processed by the extractor module, respectively. The processed HTML documents are enriched with the extracted URLs and the result of the classification process.

Retriever. The retriever is requests HTML documents and handles all outgoing connections. It uses the URL queue and robots.txt database to ensure a proper crawling access and delay. After the request the received HTML document is saved in the unprocessed HTML database. To give administrators of crawled webpages more information about ThreatCrawl, a link is embedded in the user-agent with a description and instructions to prevent it from accessing their site.

Extractor Module. The extractor scans unprocessed HTML documents and extracts all valid URLs[6]. It parses those documents and hands it over to the relevance classifier. If the document is marked as irrelevant, no further processing is done. Otherwise the extractor extracts all valid *href* attributes (*e.g.*, removal of mailto-link, check for `nofollow` tag) of anchor objects in the HTML's Document Object Model (DOM) and transformes them into absolute URLs. The remaining addresses are compared to a blacklist (*e.g.*, youtube links), added to the URL queue, and the processed document is saved in the HTML database.

Classifier. The classifier is responsible whether HTML are relevant. Websites contain a lot of irrelevant elements (*e.g.*, footers, headers). Hence, the classifier removes all of those elements and only extract the main content of the received documents. After the extraction, a document embedding is created that represents the content of the document. This embedding is compared to a ground truth vector (*cf.* § 3.4) to identify whether the document is relevant.

[6]https://developers.google.com/search/docs/crawling-indexing/robots-meta-tag

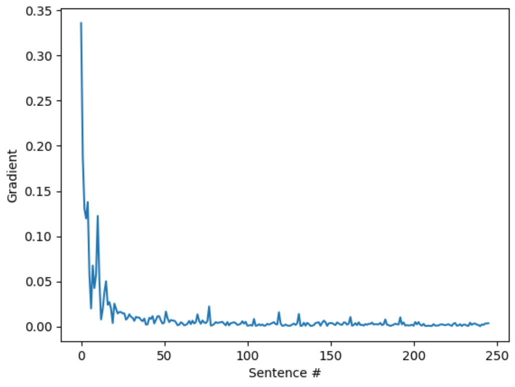

Figure 3: Example of the change of gradients over the course of a document

Monitoring Module. The monitor module stores the state of each retriever and extractor thread. Every time a thread changes its state, it communicates that transition to this module. Additionally, in case any critical error appears during the execution of ThreatCrawl, the monitor module can stop all threads immediately.

3.4 Classification

The process of classifying can be split into two parts. In the first part, ground truth vectors have to be generated to act as reference vectors for each label. By splitting up the label "relevant" into multiple sub-labels, like the ones mentioned in § 3.1, we can classify the content even more precisely. The second part is the classification and labeling of single documents, which is done when running ThreatCrawl. An embedding for each requested document is created and compared to the generated ground truth vectors.

Ground Truth Vectors. Contextualized embeddings are used to create ground truth vectors for each label. The sum of all token embeddings for a document generates the document embedding [15]. To generate a ground truth vector for a label, every document with that label in the train dataset is embedded and added up to one vector, which is then normalized.

Calculating Allowed Distance. With the ground truth vector, the angle of every document of that label to the ground truth vector is calculated to approximate the *allowed distance.* This is done by using the cosine similarity and either use the maximum of all resulting distances or the average over all distances per label. The lower the angle, the higher the semantic similarity between two embeddings. The ground truth vector and its allowed distance is saved as output of the training phase for each label.

Adaptive Sentence Usage. The calculation time of ground truth vectors varies with the number of sentences per document. To

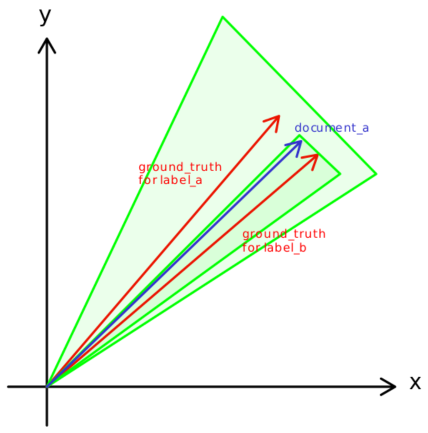

Figure 4: Illustration of the relative distance

reduce the processing time when generating embeddings, a predefined amount of sentences could be used, but this number depends on the dataset. Other metrics like median or average over all documents are not suitable, since the amount of sentences per document varies greatly, which can be seen in the analysis of the dataset (*cf.* § 5). While generating a the document embedding, the change of the embedding after each embedded sentence is monitored by measuring the distance between before and after each sentence. An example on how these gradients change is shown in Fig. 3. It can be seen that the first sentences clearly have the most influence, while all later sentences have significantly less impact on the document embedding. To boost the performance of the classification process, an adaptive sentence usage can be calculated. To do that, a fixed gradient value is chosen. As soon as the gradient after each newly added sentence embedding is below that value, the embedding process can be stopped, since additional sentences will not change the document embedding significantly anymore.

This adds a preparation phase to the generation of the ground truth vectors, where the ideal amount of sentences is calculated for every single document in the training set. This number is then averaged over all documents, resulting in the adaptive sentence usage that then is used in the ground truth generation. While this roughly doubles the processing time while training, it reduces the processing time when classifying documents later.

The result of the training process is therefore a ground truth vector, an allowed distance, and an adaptive amount of sentences per label.

Classifying Single Documents. When running ThreatCrawl, each received document is labeled by calculating the document embedding as described previously. After that, the distance to each ground truth vector is calculated. If the distance is within than the allowed distance, the document can be labeled according to that ground truth vector.

Unfortunately, when having overlapping ground truth vectors, the normal distance will lead to wrong labels. In Fig. 4 two different ground truth vectors (red) and their allowed distances (green) are shown. The embedding for *document_a* is inside the allowed distance for both labels, but closer to *label_b*. Relatively to the ground label and and its allowed distance, the document should be labeled with *label_a*, but by using the normal distance it will be labeled with *label_b*. To avoid this, additionally to the distance, the *relative distance* will also be calculated by dividing the distance by the allowed distance.

The relative distance can also be used as an inverted relevance score. The lower the relative distance, the more relevant a document is.

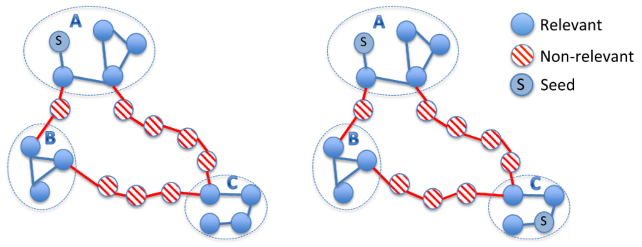

Figure 5: Example of isolated components [35]

3.5 Choosing the Right Seed

When a crawler is initially started, it needs to be given a initial seed to start processing. Choosing the right seed for a focused crawler is crucial for its performance [35]. Since focused crawlers only look for relevant information, their performance can suffer if there are no connections between different clusters of relevant information like depicted in Fig. 5 [35]. Clusters B and C will never be discovered by the crawler since there are no connections between A and B/C. This problem can be solved or mitigated by choosing a more diverse seed. In the right part of Fig. 5 we can see, that now a part of C is also used as seed, which enables the crawler to find more clusters of relevant information. Vieira et al. [35] find, that a seed has to be constructed automatically, since using resources like web directories will often lead to ineffective seeds for topics that are underrepresented or do not exist in those directories. They propose a framework called Bootstrapping Focused Crawlers (BFC) that iteratively performs queries to search engines and classifies the returned URLs, to find the ones that are more likely to be suitable seeds. That way the authors create a high number of appropriate and diverse URLs that improve the performance of a focused crawler.

4 IMPLEMENTATION

In this section, specific details about the implementation of all concepts mentioned in § 3 are given.

4.1 Used Libraries

As mentioned in § 3.3, a new crawler called ThreatCrawl had to be developed. To make ThreatCrawl as simple and lightweight as possible, native Python was primarily used. The following libraries/modules are the only ones that are used additionally to the built-in ones:

- Beautiful Soup[7]: library for parsing traversing and manipulating HTML content and documents

[7]https://www.crummy.com/software/BeautifulSoup/

- Protego[8]: library for parsing "robots.txt" files
- NumPy[9]: library for scientific computing
- PyTorch[10]: ML framework
- Transformers[11]: library for loading and handling tokenizers and ML models
- Trafilatura[12]: library for discovering and extracting web content

To extract the main content of a webpage ThreatCrawl uses Trafilatura [1], which provides a simple function that extracts the main content of an HTML document and returns it as parsed and plain text.

A blacklist for domains is essential to block common domains like Youtube or Facebook. Websites like these are widespread but have no benefit when searching for CTI relevant information, especially when focusing on text documents. Even worse, since a lot of them hide their content behind logins, they can not even be considered part of the Clear Web. Unfortunately, there is no realistic list of all websites that do not belong to the Clear Web. Therefore, the top 100 most used websites were used as a starting point to filter websites, that usually do not serve any content relevant for this work.

Since no appropriate dataset was available, a new one had to be created. To do so, multiple blogs and websites containing articles with CTI-relevant information as defined in § 3.1 were selected. This resulted in a dataset of 259 URLs. Out of these 259 URLs, 153 were labeled *relevant* and 106 *not relevant*. To enable a more precise classification, the label *relevant* was divided into sub-labels. The final dataset therefore consists of the following documents per label:

- Relevant: 153
  - *Tactics, Techniques & Procedures*: 55
  - *Broad information*: 24
  - *Malware used*: 36
  - *Vulnerability targeted*: 38
- Not Relevant: 106

It is important to note, that the sub-labels often overlap. Articles describing TTPs, for example, also contain information about used malware and the vulnerabilities that they target. This can also be seen in the analysis of the most important sentences per label (*cf.* § 5.2). In Fig. 6 the amount of sentences extracted per label can be seen.

4.2 Contextualized Embeddings

To create the contextualized embeddings, a pre-trained BERT model was used, called CySecBERT [2]. CySecBERT is a fine-tuned version of the *bert-base-uncased* model and was trained on 528M tokens collected from content in the IT security environment. The resulting model has 12 hidden layers with 768 hidden units in each layer.

Before embedding a sentence, it has to be pre-processed by tokenizing it and adding the start token and the seperator token at the end:

$$\text{Input}(sentence) = [\text{CLS}]\ \text{Tokenize}(sentence)\ [\text{SEP}]$$

[8]https://github.com/scrapy/protego
[9]https://numpy.org/
[10]https://pytorch.org/
[11]https://pypi.org/project/transformers/
[12]https://trafilatura.readthedocs.io/en/latest/

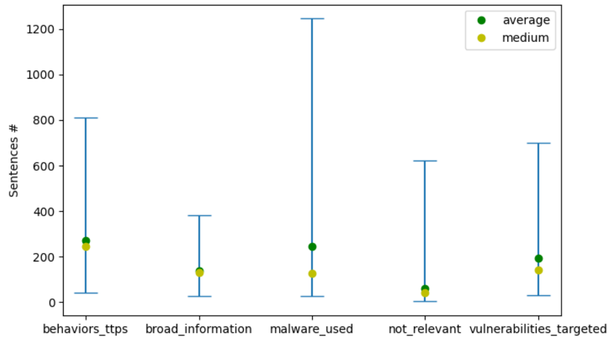

Figure 6: Sentences per label in the dataset

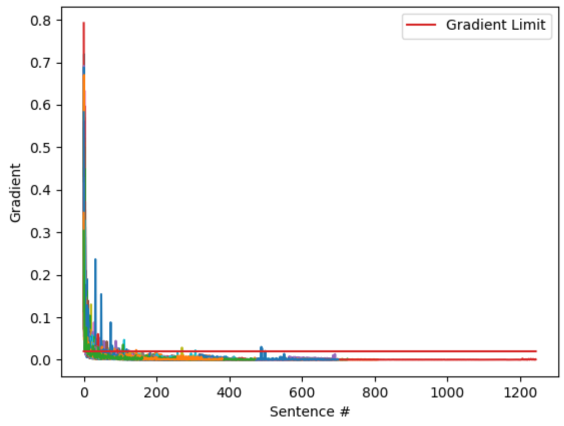

Figure 7: The sentence gradients

Since the input layer can only handle a maximum of 512 tokens, every token over that limit has to be removed. Devlin et al. [9] state that concatenating the last four hidden layers achieves high scores even without fine-tuning the model. Using the result of the last four layers, a contextualized embedding for each token in the sentence is created. Since each layer contains 768 hidden units, each embedding has a length of 4 * 768 = 3027. These token vectors can then be added up to create sentence vectors, which afterwards can be added up to create one single document vector.

Adaptive Amount of Sentences. To calculate the amount of sentences, sentence gradients for all relevant documents in the dataset were calculated (*cf.* Fig. 7). A value of 0.02 was chosen as the minimum that has to be reached, before no more significant value will be gained by considering more sentences.

Choosing the Right Seed. To maintain reproducibility, a fixed seed was used. The seed consists of 51 URLs with a huge variety between websites that typically host CTI relevant content and websites that do not. Some of the domains in the seed can also be found in the dataset and some are completely new.

5 EVALUATION

In this section the evaluation method will be described (*cf.* § 5.1), followed by an evaluation of the classification module (*cf.* § 5.2) in the context of correct classification of documents. Lastly, ThreatCrawl will be evaluated in the context of a focused crawler as a whole, *e.g.*, by evaluating its harvest rate (*cf.* § 5.3).

5.1 Evaluation Method

By analyzing the most important sentences, characteristics for each label can be observed. To extract the most important sentences in each document, the embedding of each sentence in a document is compared to the calculated embedding of the whole document. The sentence with the highest similarity is the one sentence that best captures the essence of the document.

To evaluate the classifier, a K-fold cross-validation, with K set to 5, was used, which is especially effective on sparse datasets [11]. To evaluate the CySecBERT model [2], multiple evaluations with 2 different parameters were conducted. The first parameter is the *amount of sentences per document* when creating the document embedding. As described in § 3.4, instead of a fixed amount, an adaptive amount can be used (set as default). The second parameter is the *calculation of allowed distance*, which can either be the average or the maximum distance over all distances for each label (*cf.* § 3.4) (average set as default).

For ThreatCrawl's performance the ratio of downloaded documents to downloaded relevant documents is measured, yielding the focusing efficiency. Additionally, an analysis of the resulting list of relevant documents and their relationship to each other is done.

All calculations were done on a Intel® Core™ i7-8550U processor with 4 GHz.

5.2 Evaluation of the Classification Module

An excerpt of the most important sentences results for each label is shown in Tab. 2. The sentences in *Tactics, Techniques & Procedures* contain a lot of general information about threat actors and their behavior (*e.g.*, sentence 2). On the other hand, sentence 11 describes no behaviors or TTPs at all, which shows that it is hard to separate the labels and capture the most important parts of a document. Moreover, documents that describe TTPs also contain information about malware or vulnerabilities, which changes the focus of the embedding. The same can be seen at the label *Broad information*. While it clearly focuses on general information, *e.g.*, motivation and the evolution of threat actors (*e.g.*, sentence 2), it shares information with the *Tactics, Techniques & Procedures*. Some sentences (*e.g.*, sentence 3) name vulnerabilities and others (*e.g.*, sentence 15) the researcher-group that performed the studying of threat actors. The label *Malware used* on the other hand works well, having most sentences naming specific malware or describing how they work. The last label, *Vulnerability targeted*, shows an increased occurrence of specific Common Vulnerabilities and Exposuress (CVEs) (*e.g.*, sentences 1, 7), which describe vulnerabilities, and words like "execution" and "bug" (*e.g.*, sentences 1, 2). That shows, that while the different topics of each label can be identified by the top sentences, an overlap is noticeable. Fig. 8 shows the gradient of ground truth vectors during training, displaying the stabilization of the vectors after 10 document.

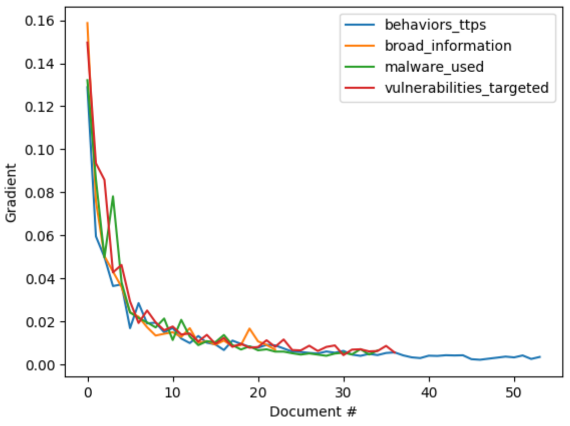

Figure 8: The ground truth gradients.

Tab. 3 shows the result of the K-fold cross-validation for 6 different sentence configurations as well as their runtime. The adaptive amount of sentences configuration exits at 11 sentences when using 0.02 as gradient limit. However, when comparing that configuration with the one using 50 sentences, the latter outperforms it for almost every label, showing that 0.02 as gradient limit was to high and should have been lower. Compared to the configuration using 100 sentences on the other hand, the one with 50 sentences still performs better. A reason is overfitting, which is confirmed by the model using all sentences, which performs even worse.

The label performing the worst over all tested configurations is *Malware used*, with an average F1 score of 0.3, which is contrary to the clear vocabulary of this label. It might be caused by the fact that the used malware is also mentioned in documents of other labels.

Another statistic that stands out is the fact, that the label *Vulnerability targeted* has a high recall, but low precision. This might be caused by the relatively high allowed distance for this label, which could also be a influential factor for the aforementioned low values for *Malware used*.

The very high recall and low precision for *relevant* is interesting as well. The reason for that is the calculation of the allowed distance. The first 5 configurations use the graceful max-distance measurement as the allowed distance. This is different in the last configuration, where the much lower, average distance of all documents per label is used as allowed distance. It reduces the number of *relevant* documents as well. Lastly it is important to note, that the range of all values between different configurations is very high, which is a sign of a dataset that might be too small. But while the values for the 4 labels *Tactics, Techniques & Procedures*, *Broad information*, *Malware used*, and *Vulnerability targeted* are not optimal, the metrics for the label *Relevance* are still in an acceptable range with a precision of 0.77, a recall of 0.99 and a resulting F1 score of 0.86 in the best configuration.

In Tab. 4 the comparison of different models is shown. Surprisingly, while the fine-tuned CySecBERT gets similar values as the S-BERT model, the more general BERT model outperforms both

Rank	***Tactics, Techniques & Procedures***
1	Instead, the threat actor leveraged a misconfiguration in GitHub repositories to get code execution and initial access on thousands of hosts across what are likely multiple victim environments worldwide
2	Based on the overlaps between UNC2165 and Evil Corp, we assess with high confidence that these actors have shifted away from using exclusive ransomware variants to LOCKBIT—a well-known ransomware as a service (RaaS)—in their operations, likely to hinder attribution efforts in order to evade sanctions
3	Part of the group's success at achieving such a long dwell time can be credited to their choice to install backdoors on appliances within victim environments that do not support security tools, such as anti-virus or endpoint protection
11	As we detail the technical components of this attack, we can confirm that we have undertaken pre-release disclosure to the victims and provided all necessary content required to remove all known attack components from their environments
15	Both are sold as malware-as-a-service on specialized hacking forums, and both allow buyers to customize the malware with their own command and control (C&C) and obfuscation methods
	Broad information
1	The transaction details and monetization patterns of modern eCrime reveal critical insights for organizations defending against ransomware attacks
2	EMBER BEAR appears primarily motivated to weaponize the access and data obtained during their intrusions to support information operations (IO) aimed at creating public mistrust in targeted institutions and degrading government ability to counter Russian cyber operations
3	The top referenced vulnerabilities associated with cyberattacks in H1 2022 affected Apache's Log4J (Log4Shell), Microsoft Windows (Follina), Microsoft Exchange Server (ProxyShell), Atlassian's Confluence, and the Java Spring Framework
15	Trellix Threat Labs includes elite researchers who have been studying cybercriminal groups and nation-backed cyber activity for years, independently and in collaboration with global government agencies
	Malware used
1	Industry reporting has claimed the Go-based ransomware dubbed PartyTicket (or HermeticRansom) was identified at several organizations affected by the attack,1 among other families including a sophisticated wiper CrowdStrike Intelligence tracks as DriveSlayer (HermeticWiper)
2	Once those two files have been written, one of the DLLs loaded by ProcessHacker has to be hijacked using a technique called a DLL search order hijacking
3	While it appears to primarily deliver commodity malware, Secureworks® Counter Threat Unit™ (CTU) researchers identified DarkTortilla samples delivering targeted payloads such as Cobalt Strike and Metasploit
4	We also provide a core set of detections designed to identify phishing campaign elements leveraged by both Caffeine-specific actors as well as more generalized phishing activity
15	Kraken's presence became more apparent at the end of September, when the security researcher nao_sec discovered that the Fallout Exploit Kit, known for delivering GandCrab ransomware, also started to deliver Kraken
	Vulnerability targeted
1	At the start of 2022, CrowdStrike Intelligence and CrowdStrike Services investigated an incident in which PROPHET SPIDER exploited CVE-2021-22941 a remote code execution (RCE) vulnerability impacting Citrix ShareFile Storage Zones Controller â to compromise a Microsoft Internet Information Services (IIS) web server
2	Our product security (PSIRT) team collaborated with our threat intelligence team, and we quickly realized that some attempts to exploit Log4j across our customer networks manifested as illegal DNS queries
3	Duo Authentication integrates with Microsoft Windows and Active Directory (AD) to support multi-factor authentication (MFA) for both remote desktop and local logons
7	This post has been updated to reflect a new CVE (CVE-2021-45105) and associated patch (Log4j)
15	The impact of this vulnerability has the potential to be massive due to its effect on any product which has integrated the Log4j library into its applications, including products from internet giants such as Apple iCloud, Steam, Samsung Cloud storage, and thousands of additional products and services

Table 2: Most influential sentences per label given by their ranking (based on their angular similarity to the ground truth vector of that specific label)

models. This could be a sign that CySecBERT might be too specialized, while the task of classifying online content requires a more generalized model.

5.3 Evaluation of ThreatCrawl

Using the results from the previous evaluation, the model BERT with 50 sentences was used to evaluate the overall performance of ThreatCrawl. To calculate the allowed distance, the maximum distance per label was used. This leads to more false positives, but also reduces the false negatives, which is considered the better trade-off for this usage.

To test ThreatCrawl, 1000 URLs were crawled, with a runtime of 19 minutes and 15 seconds. Out of these 1000 URLs, 512 were classified as relevant, which results in harvest rate of relevant documents of 51.2%. This is a huge improvement compared to Koloveas et al.'s crawler [15] with a harvest rate of 1%. When looking at the harvest rate, it is important to keep in mind that the ML model with more false positives was used and the fact, that the

#Sent.	10			50			100			all			adaptive			average distance		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
TTPs	0.46	0.64	0.53	0.7	0.44	0.5	0.5	0.22	0.2	0.46	0.18	0.17	0.42	0.67	0.5	0.81	0.35	0.46
Info.	0.45	0.25	0.32	0.77	0.48	0.53	0.73	0.45	0.52	0.48	0.28	0.24	0.4	0.35	0.36	0.47	0.45	0.44
Malw.	0.65	0.14	0.22	0.35	0.39	0.37	0.29	0.89	0.35	0.61	0.28	0.35	0.27	0.05	0.08	0.53	0.34	0.4
Vulns.	0.23	0.97	0.37	0.39	0.91	0.52	0.91	0.5	0.64	0.33	0.92	0.39	0.39	0.97	0.49	0.63	0.52	0.56
Rel.	0.61	0.98	0.76	0.77	0.99	0.86	0.66	0.97	0.78	0.62	0.99	0.76	0.62	0.99	0.76	0.97	0.62	0.75
Runtime	10m 34s			46m 1s			1h 1m 55s			2h 49m 21s			2h 48m 18s			2h 17m 59s		

Table 3: Comparison of Different Parameters. The abbreviations *TTPs, Info., Malw., Vulns.*, and *Rel.* stand for the labels *Tactics, Techniques & Procedures, Broad information, Malware used, Vulnerability targeted*, and overall *Relevance*, respectively.

	CySecBERT [2]			BERT [9]			SBERT [28]		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
TTPs	0.42	0.67	0.50	0.42	0.67	0.51	0.52	0.31	0.32
Info.	0.4	0.35	0.36	0.45	0.55	0.47	0.79	0.45	0.54
Malw.	0.27	0.05	0.08	0.43	0.17	0.22	0.39	0.16	0.22
Vulns.	0.39	0.97	0.49	0.41	0.91	0.56	0.33	0.92	0.4
Rel.	0.62	0.99	0.76	0.77	0.98	0.86	0.73	0.98	0.83

Table 4: Comparison of Different BERT Models

	URL	Label
1	www.fireeye.com/blog/…	*TTPs*
2	www.zscaler.com/blogs/…	*TTPs*
3	blog.talosintelligence.com/…	*Vulns.*
4	blog.talosintelligence.com/…	*Vulns.*
5	blog.talosintelligence.com/…	*TTPs*
6	blog.talosintelligence.com…	*Vulns.*
7	www.proofpoint.com/…	*TTPs*
8	research.nccgroup.com/…	*Vulns.*
9	www.proofpoint.com/…	*TTPs*
	…	
507	docs.github.com/…	*Vulns.*
508	support.mozilla.org/…	*Vulns.*
509	support.mozilla.org/…	*Vulns.*
510	support.mozilla.org/…	*Vulns.*
511	attack.mitre.org/…	*Vulns.*

Table 5: Sorted ThreatCrawl results ranked by their relevance with the according label.

crawler changes its path dynamically, not following routes with non-relevant documents.

Tab. 5 shows an excerpt of the top entries of the list of relevant URLs and their label given by ThreatCrawl sorted by the lowest relative distance (*cf.* § 3.4). Except number two, three, and four, all of the shown top documents are blog posts or news articles. The lower ranked URLs are somewhat related to the CTI domain and describe content like vulnerabilities or Advanced Persistent Threats (APTs), but they do not contain the specific type of content that is considered relevant for this work. This shows, that the usage of the relative distance as an effective way to rank the downloaded documents.

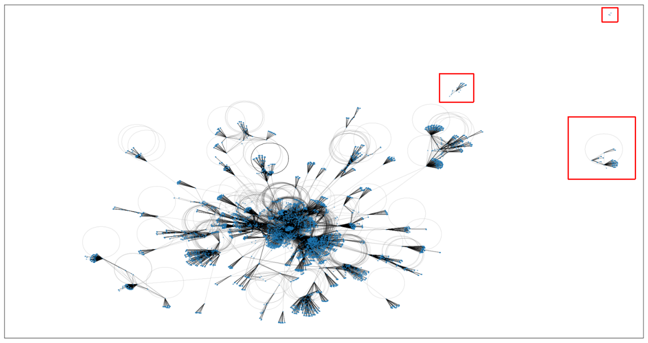

Figure 9: Multigraph of the crawled webpages.

Overview pages of news sites or blogs that contain a list of all recent articles do not get consistently classified as relevant. This can lead to situations, where documents can not get reached by ThreatCrawl because it stops one hop before those pages. The reason for this behavior is the fact, that overview pages do not contain a lot of information. When extracting the main content of those pages, not many sentences get extracted which leads to very unstable embeddings.

Multigraphs can be used to visualize the structure of all downloaded documents and their relationship to each other. In this graph, nodes represent documents and an edge between a document A and a document B represents a reference to B that appears in A. A multigraph of all explored documents is depicted in Fig. 9. As described in § 3.5, choosing the right seed is important to avoid getting stuck in isolated groups of documents. Even though a fixed set of pre-chosen URLs was used as a seed, the multigraph of all documents shows, that most documents can be reached over the available edges. Only three small isolated groups can be seen in the top right (marked red). This shows, that the seed is sufficient enough for this task. It is important to note, that this highly depends on the nature of the examined domain. News articles and blog entries in the CTIs domain are highly interconnected, which severely lowers the appearance of isolated groups of content. This does not have to be the case for content of other domains.

6 CONCLUSION

This section concludes the paper by outlining our contributions, answering our research questions (*cf.* § 1), presenting limitations, implications, and future work.

Contributions. ThreatCrawl uses a new approach to find and rank CTI-relevant information by downloading and classifying documents before references are extracted, which answers our main research question **(RQ)**.

It answers **RQ1** on how websites in CTI can be efficiently crawled and ranked (*cf.* § 1) by using dynamic pathing, where non-relevant documents are ignored to avoid whole clusters of non-relevant documents. This leading to a harvest rate of 52% for relevant documents. Other approaches use a two-step approach by downloading huge amounts of content first and filter non-relevant documents out later. Contrary to that, the design presented manages to perform both tasks at the same time. Thus, lots of resources like disk space, network resources and server capacities can be saved.

RQ2 *"How can websites be categorized automatically as CTI-relevant?"* is answered by using contextualized document embeddings. Those are calculated over the training set by averaging generated contextualized document embeddings for each document in the set. Document embeddings are generated by averaging sentence embeddings which are generated by averaging contextualized embeddings for each token in a sentence. During the crawl process, the document is accordingly labeled. The evaluation in § 5 shows good results for the implemented models. Surprisingly, the general BERT model [9] outperforms the more specialized CySecBERT [2] and S-BERT model [28].

The generated ground truth vectors can be seen as the most *relevant* documents. The closer the embedding of a document, the more *relevant* it is. Therefore, documents can be sorted based on their distance from the according ground truth vector. To make those distances comparable over all labels, those distances get converted into relative distances. Hence, the final list is a list of all relevant documents sorted by their relative distance from their ground truth vector, answering **RQ3** on how the crawling and and labeling process can be combined.

Implications. ThreatCrawl can be trained with a labeled dataset and then started to crawl any amount of URLs. Due to its flexible structure, every module can be changed and optimized on its own. The used one-step approach, its dynamic pathing and the implemented embedding and classification approach can be used to refine the search for CTI-relevant content as well as foundation for other focused crawler and the search for content in other domains.

Limitations and Future Work. Unfortunately, some limitations occurred during the development of ThreatCrawl. The high range of metrics in the evaluation (*cf.* § 5) showed, that the created dataset is too sparse and the labels chosen overlap. Future work could therefore start to define more specific labels and a build bigger dataset to generate better performing models. Another limitation of this work is the small seed. As mentioned by Vieira et al. [35], choosing the right seed is essential for focused crawlers. Methods used by Zhang et al. [37] might show further improvements in this regard. And even though Fig. 9 shows very few isolated components, an optimized seed could reveal even more unique parts of the CTI-domain. One last proposal for future work is to implement solutions to avoid the problem of overview pages as described in § 5.3. This could be for example done by detecting those pages using other NLP techniques or by implementing a two-hop approach, that checks references of non-relevant documents for one more hop before stopping to follow them. That way, the amount of occurrences where one non-relevant page prevents the access to multiple potential relevant pages can be minimized.

ACKNOWLEDGMENTS

This work was supported by the German Federal Ministry for Education and Research (BMBF) in the project CYWARN (13N15407) and German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

REFERENCES

[1] Adrien Barbaresi. 2021. Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction. In *ACL/IJCNLP'21*. ACL, 122–131. https://doi.org/10.18653/v1/2021.acl-demo.15

[2] Markus Bayer, Philipp Kuehn, Ramin Shanehsaz, and Christian Reuter. 2024. CySecBERT: A Domain-Adapted Language Model for the Cybersecurity Domain. *ACM Transactions on Privacy and Security* (2024). https://doi.org/10.1145/3652594

[3] David J. Bianco. 2014. The Pyramid of Pain.

[4] Rebekah Brown and Robert M. Lee. 2021. *SANS Cyber Threat Intelligence Survey 2021.* Technical Report. SANS.

[5] Rebekah Brown and Pasquale Stirparo. 2022. *SANS Cyber Threat Intelligence Survey 2022.* Technical Report. SANS.

[6] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. 1999. Focused crawling: A new approach to topic-specific Web resource discovery. *Computer Networks* 31 (1999), 1623–1640. https://doi.org/10.1016/S1389-1286(99)00052-3

[7] Chris Sanders and Jason Smith. 2013. *Applied Network Security Monitoring* (1 ed.). Elsevier.

[8] Kai Denker, Marcel Schäfer, and Martin Steinebach. 2019. Darknets as Tools for Cyber Warfare. In *Information Technology for Peace and Security: IT Applications and Infrastructures in Conflicts, Crises, War, and Peace*, Christian Reuter (Ed.). Springer Fachmedien, 107–135. https://doi.org/10.1007/978-3-658-25652-4_6

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NORD'19* 1 (2018), 4171–4186. https://doi.org/10.18653/v1/N19-1423

[10] Clément Elbaz, Louis Rilling, and Christine Morin. 2020. Fighting N-day vulnerabilities with automated CVSS vector prediction at disclosure. In *Proceedings of the 15th International Conference on Availability, Reliability and Security.* ACM, 1–10. https://doi.org/10.1145/3407023.3407038

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning.* Springer. https://doi.org/10.1007/978-0-387-84858-7

[12] Eoin Hinchy. 2022. *Voice of the SOC Analyst.* Technical Report. Tines. 39 pages.

[13] Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. 2017. TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources. In *Proceedings of the 33rd Annual Computer Security Applications Conference.* Association for Computing Machinery, 103–115. https://doi.org/10.1145/3134600.3134646

[14] Paris Koloveas, Thanasis Chantzios, Sofia Alevizopoulou, Spiros Skiadopoulos, and Christos Tryfonopoulos. 2021. inTIME: A Machine Learning-Based Framework for Gathering and Leveraging Web Data to Cyber-Threat Intelligence. *Electronicsweek* 10 (2021), 818. https://doi.org/10.3390/electronics10070818

[15] Paris Koloveas, Thanasis Chantzios, Christos Tryfonopoulos, and Spiros Skiadopoulos. 2019. A Crawler Architecture for Harvesting the Clear, Social, and Dark Web for IoT-Related Cyber-Threat Intelligence. In *2019 IEEE World Congress on Services (SERVICES).* IEEE, 3–8. https://doi.org/10.1109/SERVICES.2019.00016

[16] Vlad Krotov, Leigh Johnson, and Leiser Silva. 2020. Legality and Ethics of Web Scraping. *Communications of the Association for Information Systems* 47 (2020), 539–563. https://doi.org/10.17705/1CAIS.04724

[17] Philipp Kuehn, Julian Bäumler, Marc-André Kaufhold, Marc Wendelborn, and Christian Reuter. 2022. The Notion of Relevance in Cybersecurity: A Categorization of Security Tools and Deduction of Relevance Notions. In *Workshop-Proceedings Mensch und Computer.* Gesellschaft für Informatik. https://doi.org/10.18420/muc2022-mci-ws01-220

[18] Philipp Kuehn, Markus Bayer, Marc Wendelborn, and Christian Reuter. 2021. OVANA: An Approach to Analyze and Improve the Information Quality of Vulnerability Databases. In *ARES '21: Proceedings of the 16th International Conference on Availability, Reliability and Security*. ACM, 11. https://doi.org/10.1145/3465481.3465744

[19] Quentin Le Sceller, ElMouatez Billah Karbab, Mourad Debbabi, and Farkhund Iqbal. 2017. SONAR: Automatic Detection of Cyber Security Events over the Twitter Stream. In *ARES'17*. ACM, 1–11. https://doi.org/10.1145/3098954.3098992

[20] Robert M Lee. 2020. 2020 SANS cyber threat intelligence (CTI) survey. (2020), 17.

[21] Xiaojing Liao, Kan Yuan, Xiaofeng Wang, Zhou Li, Luyi Xing, and Raheem Beyah. 2016. Acing the IOC game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *CSS'16*, Vol. 24-28-Octo. ACM Press, 755–766. https://doi.org/10.1145/2976749.2978315

[22] Wenjun Liu, Yu He, Jing Wu, Yajun Du, Xing Liu, Tiejun Xi, Zurui Gan, Pengjun Jiang, and Xiaoping Huang. 2023. A focused crawler based on semantic disambiguation vector space model. *Complex & Intelligent Systems* 9 (2023), 345–366. https://doi.org/10.1007/s40747-022-00707-8

[23] Tomas Mikolov, Kai Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations*.

[24] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL'18*. Association for Computational Linguistics, 2227–2237. https://doi.org/10.18653/v1/N18-1202

[25] Davy Preuveneers and Wouter Joosen. 2022. Privacy-Preserving Polyglot Sharing and Analysis of Confidential Cyber Threat Intelligence. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*. Association for Computing Machinery, 1–11. https://doi.org/10.1145/3538969.3538982

[26] Roshni R. Ramnani, Karthik Shivaram, Shubhashis Sengupta, and Annervaz K. M. 2017. Semi-Automated Information Extraction from Unstructured Threat Advisories. In *Proceedings of the 10th Innovations in Software Engineering Conference*. Association for Computing Machinery, 181–187. https://doi.org/10.1145/3021460.3021482

[27] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. 45–50. https://doi.org/10.13140/2.1.2393.1847

[28] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 3982–3992. https://doi.org/10.18653/v1/D19-1410

[29] Thea Riebe, Tristan Wirth, Markus Bayer, Philipp Kuehn, Marc-André Kaufhold, Volker Knauthe, Stefan Guthe, and Christian Reuter. 2021. CySecAlert: An Alert Generation System for Cyber Security Events Using Open Source Intelligence Data. In *Information and Communications Security (ICICS)*. 429–446. https://doi.org/10.1007/978-3-030-86890-1_24

[30] Dave Shackleford and Stephen Northcutt. 2015. *Who's using cyberthreat intelligence and how.* Technical Report. 24 pages.

[31] Hyejin Shin, WooChul Shim, Saebom Kim, Sol Lee, Yong Goo Kang, and Yong Ho Hwang. 2021. #Twiti: Social Listening for Threat Intelligence. In *Proceedings of the Web Conference 2021*. Association for Computing Machinery, 92–104. https://doi.org/10.1145/3442381.3449797

[32] Yahya Tawil and Saed Alqaraleh. 2021. BERT Based Topic-Specific Crawler. In *2021 Innovations in Intelligent Systems and Applications Conference (ASYU)*. 1–5. https://doi.org/10.1109/ASYU52992.2021.9599076

[33] Andrea Tundis, Samuel Ruppert, and Max Mühlhäuser. 2020. On the Automated Assessment of Open-Source Cyber Threat Intelligence Sources. In *Computational Science – ICCS 2020*, Valeria V. Krzhizhanovskaya, Gábor Závodszky, Michael H. Lees, Jack J. Dongarra, Peter M. A. Sloot, Sérgio Brissos, and João Teixeira (Eds.). Springer International Publishing, 453–467. https://doi.org/10.1007/978-3-030-50417-5_34

[34] Andrea Tundis, Samuel Ruppert, and Max Mühlhäuser. 2022. A Feature-driven Method for Automating the Assessment of OSINT Cyber Threat Sources. *Computers & Security* 113 (2022), 102576. https://doi.org/10.1016/j.cose.2021.102576

[35] Karane Vieira, Luciano Barbosa, Altigran Soares Silva, Juliana Freire, and Edleno Moura. 2016. Finding seeds to bootstrap focused crawlers. *World Wide Web* 19 (2016), 449–474. https://doi.org/10.1007/s11280-015-0331-7

[36] Wenxian Wang, Xingshu Chen, Yongbin Zou, Haizhou Wang, and Zongkun Dai. 2010. A Focused Crawler Based on Naive Bayes Classifier. In *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*. 517–521. https://doi.org/10.1109/IITSI.2010.30

[37] Haoxiang Zhang, Aécio Santos, and Juliana Freire. 2021. DSDD: Domain-Specific Dataset Discovery on the Web. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. Association for Computing Machinery, 2527–2536. https://doi.org/10.1145/3459637.3482427