


RESEARCH

Open Access



TIM: threat context-enhanced TTP intelligence mining on unstructured threat data

Yizhe You^{1,2}, Jun Jiang^{1,2}, Zhengwei Jiang^{1,2*} , Peian Yang¹, Baoxu Liu^{1,2}, Huamin Feng⁴, Xuren Wang³ and Ning Li^{1,2}

Abstract

TTPs (Tactics, Techniques, and Procedures), which represent an attacker's goals and methods, are the long period and essential feature of the attacker. Defenders can use TTP intelligence to perform the penetration test and compensate for defense deficiency. However, most TTP intelligence is described in unstructured threat data, such as APT analysis reports. Manually converting natural language TTPs descriptions to standard TTP names, such as ATT&CK TTP names and IDs, is time-consuming and requires deep expertise. In this paper, we define the TTP classification task as a sentence classification task. We annotate a new sentence-level TTP dataset with 6 categories and 6061 TTP descriptions from 10761 security analysis reports. We construct a threat context-enhanced TTP intelligence mining (TIM) framework to mine TTP intelligence from unstructured threat data. The TIM framework uses TCENet (Threat Context Enhanced Network) to find and classify TTP descriptions, which we define as three continuous sentences, from textual data. Meanwhile, we use the element features of TTP in the descriptions to enhance the TTPs classification accuracy of TCENet. The evaluation result shows that the average classification accuracy of our proposed method on the 6 TTP categories reaches **0.941**. The evaluation results also show that adding TTP element features can improve our classification accuracy compared to using only text features. TCENet also achieved the best results compared to the previous document-level TTP classification works and other popular text classification methods, even in the case of few-shot training samples. Finally, the TIM framework organizes TTP descriptions and TTP elements into STIX 2.1 format as final TTP intelligence for sharing the long-period and essential attack behavior characteristics of attackers. In addition, we transform TTP intelligence into sigma detection rules for attack behavior detection. Such TTP intelligence and rules can help defenders deploy long-term effective threat detection and perform more realistic attack simulations to strengthen defense.

Keywords: TTPs, Threat intelligence, Natural language processing (NLP), Advanced persistent threat (APT)

Introduction

Cyber threat intelligence (CTI) is the information and knowledge used for defense and detection in cyber warfare. Traditional threat protection uses IOC (indicator of compromise) intelligence, such as IP, domain name, and malicious file hash to generate detection rules. However, there are many problems with IOC intelligence-based

protection. Zhu and Dumitras (2018) mentioned that, although intelligence sharing standard STIX 2.1 (OASIS 2021) includes an attack pattern field, most open-source intelligence feeds do not provide IOC-related attack patterns or attack technique information to enrich IOC intelligence (e.g., IP threat type: a brute-force IP or a command & control server IP). This makes IOC-based protection more blind. When an IOC-based protection device generates an alert, the defender does not know what kind of attack is going on behind this IOC alert and cannot respond effectively because the IOC does not have an attack technique description. Nation-state APT

*Correspondence: jiangzhengwei@iie.ac.cn

¹ Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

Full list of author information is available at the end of the article

(advanced persistent threat) groups can also easily evade this IOC protection mechanism by shifting their attack infrastructures, such as changing attack IPs, phishing domain names, and modifying malicious codes. Thus, the Pyramid of Pain (David)Bianco 2021) considers IOC as low-value and easily accessible intelligence.

TTPs (Tactics, Techniques, and Procedures) can describe the long-term behavior and essential features of an attacker. MITRE constructs a TTP knowledge base named ATT&CK (MITRE 2021) to provide TTP unified names and procedure examples of attack techniques. They use the tactic to represent the goals of each attack campaign stage and techniques to describe how attackers accomplish these goals. TTP intelligence can thus be used in penetration tests of enterprises to compensate for defense deficiencies.

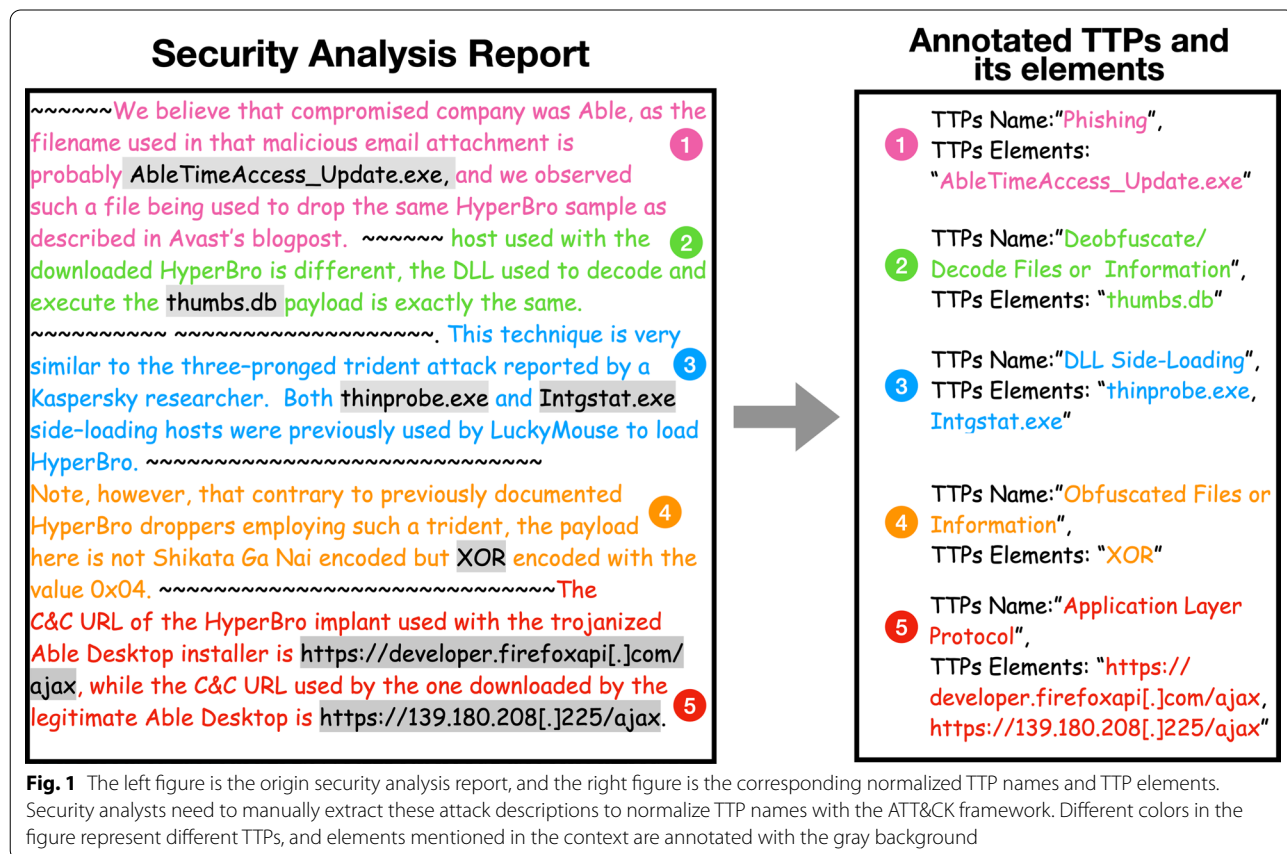
However, high-value TTP intelligence is difficult to obtain directly. Cybersecurity analysts generally use natural language to describe TTP intelligence in security analysis reports (Tartare 2021).

Figure 1 shows the TTP description examples in a security analysis report. The left represents the report text, and we use different colors to annotate the different TTP descriptions. The left also shows that the TTP

description contains a great deal of IOC information and other security terms. We use the gray background to annotate them. This paper refers to these IOCs and other types of security terms in the TTP description as TTP elements. The right side shows the ATT&CK TTP name and TTP elements obtained from the left description text.

Manually converting these TTP descriptions into ATT&CK standard names is very time-consuming and requires in-depth expert knowledge. The existing NLP (natural language processing) methods (training on SST, AG News, DBpedia corpus, etc.) cannot be directly used on cybersecurity-related text because there is no available security corpus to train a TTP classification model.

Several existing TTP classification methods (Ayoade et al. 2018; Legoy 2019; Li et al. 2019; Niakanlahiji et al. 2018) are at the document level, which may cause low-accuracy problems since the articles may consist of different kinds of TTPs. These methods also have limitations that can only provide static names with a confidence coefficient without providing more details, such as related TTP elements, which are also significant to cyber defenders. The static bag-of-words method proposed by Husari et al. (2017) is not robust enough to classify



TTP intelligence in complex security analysis articles and needs a prebuild knowledge base. The subsequent work of Husari et al. (2018) tries to find threat behavior, consisting of one verb and one noun object. These verb-object pairs are too simple to describe complete TTPs. However, none of the existing TTP classification methods have evaluated the performance of their method on few-shot training sample cases, and many TTPs have only a few description texts.

In addition, previous methods have taken static TTP names as the final outcome and lacked TTP description details. As a result, this TTP information is difficult for defenders to use.

Motivation & Challenges

Based on the above discussion, our motivation is to automate obtaining long-term valid and more essential features of attackers, such as the TTPs from unstructured threat data. Specifically, this paper uses the text classification method to find and classify TTP descriptions from security analysis reports to represent such features and use them for better protection. Therefore, our work mainly faces two major challenges: the lack of available sentence-level TTP dataset and the need to accurately find and classify TTP-related descriptions, even in the case of few-shot training samples.

Our study

We define TTP intelligence as the detailed description and elements of TTPs in unstructured threat data. To obtain TTP intelligence from unstructured threat data, we built a threat context-enhanced TTP intelligence mining framework named TIM that crawls analysis reports from security websites and mines the TTPs intelligence.

To solve the challenge of no available dataset, we build a new TTP dataset at the sentence level. We use the MITRE ATT&CK framework to define TTPs rather than the incomplete and simple threat action used by Husari et al. (2018). Based on the suggestions of frontline threat intelligence analysts, we selected 6 categories of TTPs as classification targets to validate the feasibility of the TTP intelligence mining framework in this paper. These TTP categories cover the 7 major tactics of ATT&CK and are very representative. We annotate 6061 TTP descriptions from 10761 security reports.

To solve the low-accuracy problem of the previous document-level TTP classification methods, we design and propose the TCENet (Threat Context Enhanced Network) model to classify TTPs at the sentence level. We define the threat context as the TTP description and the TTP elements. A TTP description consists of three continuous sentences and includes the textual information of the threat context. The TTPs element represents the

12 categories of security terms contained in the threat context, such as IPs, URLs, file hashes, CVE, protocols, encryption algorithms, etc. We design a TTP elements correlation coefficient calculation method to prove the rationality of adding TTP element features into the classification.

To demonstrate that the TCENet used by the TIM framework can accurately find and classify TTP descriptions in security analysis reports, we perform a variety of evaluations on our annotated dataset.

The evaluation results show that our TCENet is better than previous document-level methods and mainstream text classification methods. The evaluation results also show that the TTP element feature improves the classification performance. In the case of small samples, the TTP classification accuracy of TCENet is still better than that of the comparison model.

Finally, we organize the TTP descriptions and corresponding TTP elements into shareable intelligence in STIX 2.1 format as well as Sigma detection rules (MSigmaHQ 2021). Cyber defenders can obtain more valuable and direct intelligence about the attack and update their defense rules and mechanism without reading the whole security analysis report. TTP intelligence containing specific TTP elements allows for more realistic attack simulation. Sigma detection rules that include TTP names and TTP elements can provide richer threat context information, enabling defenders to make more targeted defenses.

Contributions

In general, our contributions are as follows:

- We annotate 10761 reports from 5 security vendor websites and build a TTP corpus containing 6 types of popular ATT&CK TTPs and 6061 descriptions. This will be the first study that builds a sentence-level TTP dataset.
- We propose a framework named TIM for mining TTP intelligence from unstructured threat data such as security analysis reports. The TIM framework uses our proposed TCENet to find and classify TTP descriptions from reports. This is the first work to perform TTP classification at the sentence level using pretrained language models and TTP element features. The TIM framework eventually generates shareable TTP intelligence in STIX 2.1 format as well as Sigma detection rules for better protection.
- The final experimental results show that the sentence-level TCENet in our TIM framework achieves better performance on precision, recall, and F1 than previous document-level TTP classification work and mainstream text classification methods, even in the case of few-shot training samples. The experimental

results also demonstrate that our work can be generalized for mining TTP intelligence in most categories; in particular, some TTPs only have a few description texts.

Related works

Ayoade et al. (2018) used a TF-IDF with the SVM classifier to classify TTPs at the document level. Legoy (2019) used TF-IDF and Word2Vec to represent the whole security analysis report. They leverage Adaboost, linear SVC, decision tree, etc., to classify TTPs at the document level. The linear SVC with the TF-IDF article vector achieved the best performance in their experiments. Li et al. (2019) used latent semantic analysis to generate topics of targeting articles and compared the topic vectors with the TF-IDF vectors of ATT&CK description pages to obtain cosine similarity. Then, they used these similarity vectors with naive Bayes and decision trees to classify TTPs. Niakanlahiji et al. (2018) used the TF-IDF score of the independent noun phrase in security analysis articles to find the keywords to represent the TTPs. They used these keywords to query analysis articles in their corpus.

The above document-level methods can only output several static TTP names and cannot provide more detailed and specific information about an attack. Our work uses regex and gazetteer to extract TTP elements from the TTP description text, making the result more concrete than a static TTPs name. The IOC intelligence extracted in the TTP context can be used for intelligence sharing and detection rule generation, which has more security value than the general IOC without TTP information.

Other previous methods tried to find more atomistic descriptions of TTP, such as a verb-noun phrase, which they defined as threat action. Husari et al. (2017) used TF-IDF with an enhanced BM25 weight function to generate a word bag of candidate threat action text and compared it with the word bags in their knowledge base to obtain threat action in the text. The subsequent work of Husari et al. (2018) used entropy and mutual information

to find the object-verb pairs of high mutual information in the malicious software-related Wikipedia and used these object-verb pairs to find the threat actions of equal mutual information in the security analysis report.

Compared with the above two works, our work does not need a heavy prebuild knowledge base. Unlike the bag-of-words method used by Husari et al. (2017), our work uses a pretrained language model as our word embedding model, which gives more context features than bag-of-words methods or static word-vector models. Therefore, compared with bag-of-words and static word embedding, the pretraining model can improve classification accuracy. Our work uses 6061 TTP descriptions from 10761 reports for TCENet training, which is more robust than the word-bag method in Ghaith’s work (Husari et al. 2017). We classify TTPs at the ATT&CK technique level rather than the threat action level in Husari et al. (2018) because the threat action is too simple to represent the complete TTPs.

None of the current work evaluates the performance of TTP classification in few-shot training sample cases. However, many TTPs only have a few description texts, which makes the existing work not generalizable in the TTP classification task.

Our work uses both textual and TTP element features to enhance our TCENet. The evaluation result also shows that the classification accuracy of our TCENet model is better than that of other methods, even in the case of few-shot training samples. This means that our method can be generalized to most TTP classification tasks.

Preliminaries

TTPs

Tactics, Techniques, and Procedures are three different levels of the cyberattack campaign derived from military terminology. In this work, we select the most popular 5 techniques and 1 tactic from ATT&CK as our TTP classification targets, as shown in Table 1.

Tactics represent multiphases objectives of an attack campaign, such as initial access, persistence, and privilege escalation. Techniques represent the method to accomplish the stage objective, such as using phishing or

Table 1 TTPs categories and TTPs description text examples

TTPs	Description example
Phishing	Dragonfly has used spearphishing campaigns to gain access to victims.
Scheduled Task/Job	Remsec schedules the execution one of its modules by creating a new scheduler task.
Obfuscated Files or Information	Agent Tesla has had its code obfuscated in an apparent attempt to make analysis difficult.
Deobfuscate/Decode Files or Information	Carbon decrypts task and configuration files for execution.
Collection*	The jar file contains various classes for platform-specific implementations for capturing screenshots, capturing audio, logging keystrokes, among others.
Application Layer Protocol	Carbon can use HTTPs in C2 communications.

drive-by compromise to enter the victim’s network; procedures represent the specific implementation instance of a technique.

TTP intelligence

Previous TTP intelligence mining works only output static TTP names as results. These results lack description details about the TTPs and are difficult to use for defense. We define TTP intelligence as the detailed description and elements of TTPs in unstructured threat data.

Threat context

In this paper, the threat context of TTP intelligence is defined as two parts: TTP description and TTP elements.

TTP description

Rather than supposing a whole paragraph as a TTP description, as in the work of Li et al. (2019), we focus on TTPs at the sentence-level to achieve a more accurate classification. We define three continuous sentences as a TTP description.

TTP element

As shown in Fig. 1, the TTP description contains a number of terms that are closely related to a particular TTP: IP, domain name, URL, CVE, and security terms. We refer to these terms as elements. In this paper, we have defined 12 types of TTP elements: IPv4, domain, email, filename, URL, file hash, file path, regkey, CVE, encode&encryption algorithm, communication protocols and the data object keyword (e.g., clipboard, screen, snap-shot, keylogging, password, outlook, etc.), as shown in Table 2.

TTP description example

Gorgon Group uses Microsoft Word documents with CVE-2017-0199 to implement the phishing attack. And the Patchwork group uses the MS PowerPoint document, which exploits CVE-2014-6352, to implement the phishing attack. APT groups frequently use this method to gain initial access to victims’ networks.

This is an example of a TTP description that describes the procedure of two different APT groups. This description belongs to the phishing TTP category. CVE-2017-0199 and CVE-2014-6352 are the TTP elements of the above two TTP procedures examples. Based on the TTP elements in intelligence, defenders can perform different attack simulations and generate threat detection rules to improve the effectiveness of the defense.

TTP classification definition

In this paper, we define the TTP classification problem as a text classification task. Given a sentence S_n in an analysis report, we first obtain its context $C_n = \{S_{n-1}, S_n, S_{n+1}\}$ and sentence embedding CE_n using Sentence-BERT (Reimers and Gurevych 2019). We then use regex and gazetteer to extract TTP elements in the context C_n . We use the occurrence number of specific TTP element types in C_n to represent the TTP element features. The element feature vector represents $Elms_n = \{Elm_1, Elm_2, \dots, Elm_k, \dots, Elm_m\}$, where Elm_k represents the number of occurrences of the k -th type TTP element in the context C_n . The length of vector $Elms_n$ is 12 because we define 12 types of elements in this work. Our proposed method TCENet uses the description context embedding CE_n and the TTPs element type vector $Elms_n$ as input and classifies the TTP type TTP_i of sentence S_n , which can be denoted as Eq. 1.

Table 2 12 TTPs element types

TTPs Element	Example	Extract Method
IPv4	192.168.1.1	Regex
Domain	Example.com	Regex
Email	mail@example.com	Regex
Filename	example.vba	Regex
URL	http://example.com/project/example.php	Regex
File Hash	66efff4c945d3c3b87fc271b47d456db	Regex
File Path	/home/example/example.o	Regex
CVE	cve-2017-11882	Regex
Encode&Encryption Algorithm	Base64, XOR, etc.	Gazetteer
Communication Protocols	HTTP, SMTP, etc.	Gazetteer
Data Object	clipboard, screen, password, etc.	Gazetteer
Regkey	HKCU/Software/Microsoft/Windows/CurrentVersion/Run	Regex

$$TTP_i = TCENet(CE_n, Elms_n) \quad (1)$$

TTP intelligence mining

TTP intelligence mining is the process of finding TTP-related description texts from security analysis reports and organizing them into a shareable intelligence format (e.g., STIX 2.1). In this paper, we propose a threat context-enhanced TTP intelligence mining framework TIM that finds and classifies TTP descriptions from security analysis reports by using the TCENet proposed in this paper. The TIM framework then organizes the TTP descriptions and TTP elements into a shareable intelligence in STIX 2.1 format as well as Sigma detection rules.

Algorithm 1 describes the TTP intelligence mining process for one cybersecurity analysis report.

Algorithm 1 The TTPs intelligence mining process

Input:

- 1: Report $R = \{S_1, S_2, S_3, \dots, S_k, \dots, S_n\}$, $k \in [1, n]$, S_k is the k -th sentence of the report.
- 2: Model $TCENet = \{NN_1, NN_2, \dots, NN_i, \dots, NN_m\}$, $i \in [1, m]$, where NN_i is our proposed model for the i -th type TTP_i .

Output: TTP intelligence $TI = \{TI_1, TI_2, \dots, TI_l, \dots, TI_p\}$, $l \in [1, p]$, TI_l is the l -th TTP intelligence.

- 3: **for** $k = 1$ **to** n **do**
- 4: **if** $k = 1$ **then**
- 5: $S_{k-1} \leftarrow S_k$
- 6: **if** $k = n$ **then**
- 7: $S_{k+1} \leftarrow S_k$
- 8: $C_k = \{S_{k-1}, S_k, S_{k+1}\}$
- 9: $CE_k \leftarrow \text{Sentence} - \text{BERT}(C_k)$
- 10: $Elms_k \leftarrow \text{Regex} - \text{Gazetteer}(C_k)$
- 11: **for** $i = 1$ **to** m **do**
- 12: $T_p \leftarrow NN_i(CE_k, Elms_k)$
- 13: **if** $T_p = 1$ **then**
- 14: $TI_l \leftarrow \{TTP_i, C_k, Elms_k\}$
- 15: **if** $T_p = 0$ **then**
- 16: continue;
- 17: **return** TI

Dataset

Data source

A major contribution of our work is to annotate the first sentence-level TTP dataset. We build our dataset by crawling the security analysis reports from security vendor websites, including: **Malwarebytes** (Malwarebytes 2021), **Securelist** (Securelist 2021), **Welivesecurity** (ESET 2021), **Trendmicro** (Trendmicro 2021), and **Threatpost** (Threatpost 2021). We crawl security analysis reports using the category tag of the report (e.g., malware, analysis, apt, etc.). Therefore, it is possible to remove ads and other nonsecurity analysis reports directly. We finally acquired 10761 security analysis reports.

The statistics of reports used by previous document-level TTP mining methods (Ayoade et al. 2018; Legoy

Table 3 Report statistics of different researches

Dataset	The number of reports
Legoy (2019)	1490
Li et al. (2019)	55
Ayoade et al. (2018)	18257
Our sentence-level dataset	10761

Table 4 Report statistics of different security vendors

Vendor name	Number of reports
Malwarebytes	3047
Securelist	1299
Welivesecurity	675
Trendmicro	3951
Threatpost	1789

2019; Li et al. 2019) and our annotated dataset are shown in Table 3.

Our sentence-level dataset are more bigger than that of Legoy (2019) and Li et al. (2019). While 17600 reports of Ayoade et al. (2018) dataset from a single resource of Symantec, ours comes from five different vendors with more balanced distribution. Therefore, our dataset is more general.

The distribution of reports in our dataset according to different vendors is shown in Table 4.

Annotation

Our annotation work is done by three threat intelligence researchers.

Annotators first learn the specific concepts of the 6 ATT&CK TTPs used in this paper. Then, annotators start to read the 10761 security analysis reports collected. We split the report into sentences. The annotator manually extracts the TTP descriptions (three continuous sentences) from the security analysis report and saves them in a file of specific TTPs.

From these reports, we annotated 6061 TTP descriptions and used regexps with a gazetteer to obtain TTPs elements. The annotation number of each TTP category is shown in Table 5. The annotation results were revised by three other cybersecurity researchers.

Dataset validation

As mentioned above, our dataset has been annotated by threat intelligence researchers and revised by cybersecurity researchers with domain expertise. To objectively

Table 5 TTPs description annotation number in our dataset

TTPs	TTPs Description Annotation Number
Phishing	2599
Scheduled Task/Job	451
Obfuscated Files or Information	439
Deobfuscate/Decode Files or Information	475
Collection	1401
Application Layer Protocol	696

demonstrate the validity of our dataset, we use the TTP keyword matching method to evaluate the TTP keyword match rate of our dataset.

To obtain objective and accurate TTP keywords, we use the TTP procedure description instances from the ATT&CK website as the corpus. After removing the stop words, we calculate the TF-IDF score of terms in TTP description instances and select the top ten scored words. We use these keywords for queries in the dataset and calculate how many TTP descriptions are matched. The matching result shows that the keyword match reaches an average of 0.925 in the positive sample. This indicates that the vast majority of our labeled positive samples are consistent with the keywords mentioned in the TTP descriptions of the ATT&CK website. This means that our labeled dataset is valid and can be used for model training.

TTP correlation

The TTP description example in Fig. 1 shows the TTP elements that appear in the TTP threat context and correlate with specific TTPs.

Therefore, we designed a TTP elements correlation calculation method based on our dataset to evaluate the correlation between TTP elements and TTP categories.

We use **Cyobstrack** (cmu-sei 2021) and our TTP element **gazetteer** to extract TTP elements of each TTP description in our TTP dataset.

We first calculate the support coefficients of each TTP element category at different positions of the TTP description. This coefficient represents the distribution of TTP elements categories in our dataset, as shown in Eq. 2:

$$\begin{aligned}
 Support_p(Elms^i) &= \frac{\ln(|Elms_p^i|)}{\ln(|Elms_p|)}, \text{ where } p \in \{c, d\} \\
 Elms_{sup}^i &= \begin{cases} Support_c(Elms^i) & \text{if } p = c \\ Support_d(Elms^i) & \text{if } p = d \end{cases}
 \end{aligned}
 \tag{2}$$

where i denotes the i -th type of TTP element and p denotes the TTP element position in the TTP description. If an element is described in mid-sentence, it is as a direct element, represented as d in Eq. 2. Otherwise, it is a context element represented as c . $Elms_p^i$ denotes the i th type elements described in position p . We use a logarithmic fraction to measure the support coefficients of the $Elms^i$ in position p . $Elms_{sup}^i$ denotes two different position support coefficients of $Elms^i$.

For each element in a specific TTP description, we select three verbs closest to the target element and then compare these verbs to the verbs in the corresponding ATT&CK TTP description page with BERT (Devlin et al. 2018) embedding. We select ATT&CK description verbs by using TF-IDF. Then we calculate the cosine similarity between the candidate verb and the TTP description page verb to find the most similar verb in a TTP description. $SimV$ denotes the max cosine similarity, and the $Vmax$ denotes the most similar verb.

We also take the distance factor between the verb $Vmax$ and element into account, denoted as $dist(Vmax, Elms)$. Thus, the text relevance about textual and lexical features of $Elms^i$ computed from:

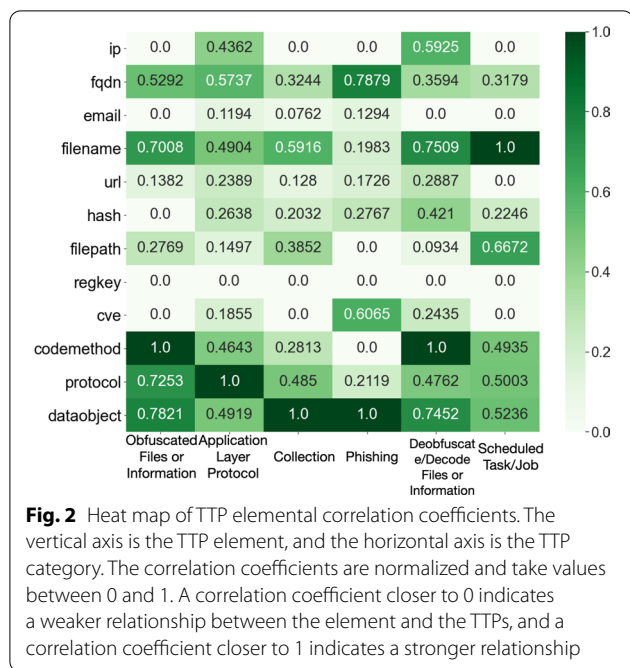
$$R_{text}(Elms^i) = \sum_{j=1}^k \exp\left(\frac{SimV_j^i}{dist(Vmax_j^i, Elms^i)}\right)
 \tag{3}$$

j in Eq. 3 is the j th instance of the TTP description. With the data distribution, textual, and lexical features, we calculate the average correlation coefficient of the i th element type to the specific TTPs using Eq. 4, where $|Elms_i|$ is the amount of the i th element type.

$$coefficient(Elms^i) = \frac{Elms_{sup}^i \cdot R_{text}(Elms^i)}{|Elms^i|}
 \tag{4}$$

We then normalize each TTP correlation coefficient score and show the coefficient by using a heat map in Fig. 2. The normalized coefficient score takes values from 0 to 1. Scores close to 1 indicate strong relevance, and scores closer to 0 indicate weak relevance.

The result in Fig. 2 shows that there is a strong correlation between some elements and the particular TTPs. The *code method*, *protocol*, and *data object* TTP elements are the most correlated elements in the **obfuscate**, **C2 application layer protocol**, and **collection** TTPs. These elements can provide details to the TTP instance, so they frequently appear in the specific context. The *filename* element is the most correlated element in the **Scheduled Task/Job** TTPs because attackers would use some scripts (e.g., .bat or .vbs) and malicious files to create a scheduled task or create a scheduled task to execute other malicious files and scripts to perform a further attack. The *Data object* element is



the most correlated element type in the **Phishing** TTPs because email service names such as *Microsoft Outlook* and *Mailbox* (we defined as data objects) are frequently mentioned in the **Phishing** threat context. The *FQDN* and *CVE* are the second and third correlated elements in **Phishing** TTPs since many phishing attack descriptions would mention the phishing domain names and the *CVE* exploit by attackers to create malicious phishing documents.

It is worth noting that the *email* elements obtain a low correlation score because only a few reports disclose the attack email address or victim address. The *URL* elements also obtain a low correlation value because there are only a few **Phishing** description instances that mentioned the specific URL in our dataset.

Many URL phishing description texts would only mention the attacker performing the attack by using URLs without giving the URL details, or the phishing URLs are mentioned outside the context window, so they would not be considered TTP elements.

The *regkey* element encounters the same problem that it may be described outside the context window, so there are only a few instances in our dataset. This is the limitation of our work, and we will discuss how to solve it in the future work section.

Threat context-enhanced TTP intelligence mining framework

We designed a threat context-enhanced TTP intelligence mining framework named TIM, as shown in Fig. 3. The TIM framework consists of five modules: crawling,

preprocessing, feature embedding, TTP classification, and TTP intelligence generation.

Crawling and preprocessing

Crawling

We first crawl 10761 security analysis reports from 5 data sources using category tags such as **malware**, **threat analysis**, etc., to filter security-related articles.

Preprocessing

We use BeautifulSoup (Richardson 2021) to clean all HTML tags and continuous line breaks. As we defined above, the TTP description contains three continuous sentences. We then split these articles with a 3-size sliding window by NLTK tools in Python. Next, we extract the TTP elements from each TTP description. We use Cyobstruct (cmu-sei 2021) to extract TTP elements of the IOC type. The actual output of Cyobstruct is normalized format IOC, such as 192.168.1.1. Thus, we modify the output function of Cyobstruct to find the original IOC elements in the cyber analysis report, such as 192[.]168[.]1[.]1. We also construct a TTP element **gazetteer**, as shown in Table 6, to match non-IOC elements such as protocol names and encryption algorithms.

We then replace all found TTP elements with element holder $\$[Elms.]$ to avoid unexpected tokens when tokenizing the whole TTP description. We resume these elements after TTP classification and use them to generate TTP intelligence.

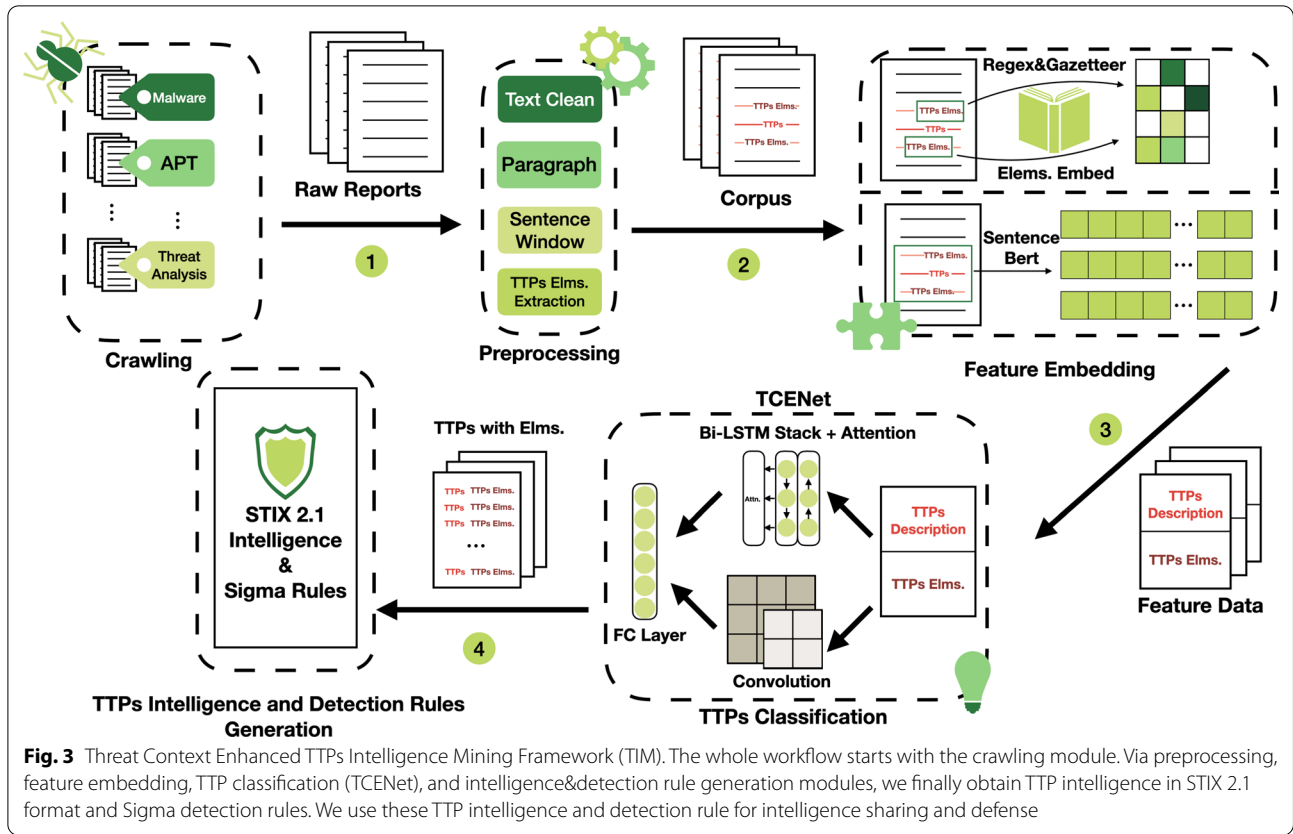
Feature embedding and TTP classification

Figure 4 shows the architecture of our TCENet. Based on the TTP elements correlation result in Fig. 2, the TCENet model consists of two paths: element feature extraction path (upper) and description feature extraction path (lower). A fully connected layer would jointly learn the feature extracted by these two paths for final classification. We use binary-relevance to train the TCENet model on 6 types of TTP data.

Elements feature path

We use regex and a TTP elements gazetteer to extract 12 types of TTP elements in the TTP description. We construct the number of occurrences of each element type as a TTP element feature vector *Elms*.

Each type of element corresponds to one dimension of the vector. For example, if there are 2 hashes, 3 email addresses and 1 CVE ID described in the spear-phishing TTP threat context, the element embedding vector *Elms* would be the vector as: [0, 0, 3, 0, 0, 2, 0, 0, 1, 0, 0, 0]. Elements are organized in the following order: [ip, fqdn, email, filename, url, hash, file path, regkey, cve, code method, protocol, data object]. We then normalize



these element vectors before training. The TTP element embedding process is shown in Algorithm 2.

Algorithm 2 TTPs element embedding process.

Input:
 1: TTP descriptions $D = \{D_1, D_2, \dots, D_k, \dots, D_n\}$, $k \in [1, n]$, and n is the number of TTP descriptions.
 2: TTPs elements types list $ElmT = \{ElmT_1, ElmT_2, \dots, ElmT_i, \dots, ElmT_{12}\}$, $i \in [1, 12]$. The TTP element vector is organized in the following order: ip, fqdn, email, filename, url, hash, file path, regkey, cve, code method, protocol, and data object.
 3: Match function M_f , which uses regex and gazetteer to match TTP elements in the TTP description and outputs the occurrence number Occ_i of each type of element.
Output: Normalized TTPs element embedding matrixs $Elms$
 4: $Elms \leftarrow []$
 5: **for** $k = 1$ to n **do**
 6: $Elms_k \leftarrow []$
 7: **for** $i = 1$ to 12 **do**
 8: $Occ_i \leftarrow M_f(D_k)$
 9: $Elms_k.append(Occ_i)$
 $Elms.append(Elms_k)$
 10: normalized $Elms \leftarrow normalize(Elms)$
 11: **return** $Elms$

Inspired by the malware analysis work of Nataraj et al. (2011) that transforms the binary file to a 2D grey-scale map, we resize the element vector to a 4*3 matrix and use two different CNN filters to extract element features $Elmf_n$.

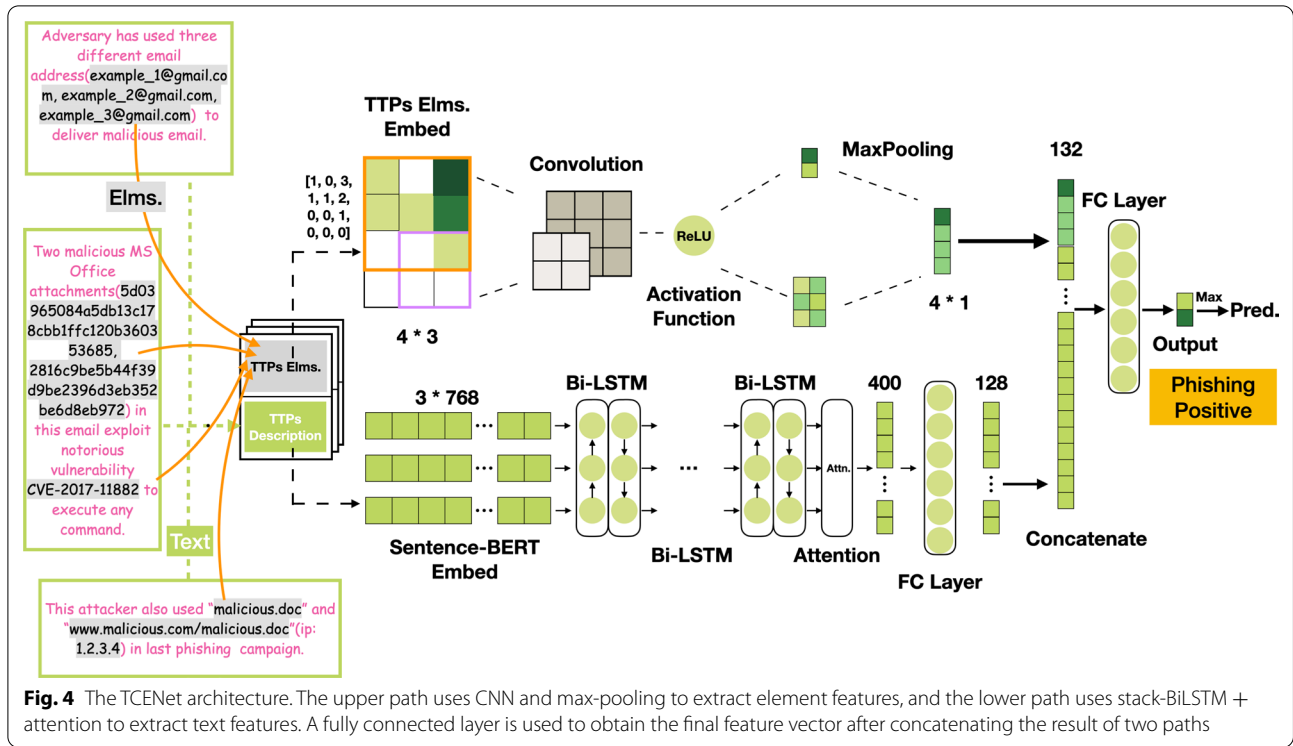
Table 6 TTPs element gazetteer

Element	Gazetteer words
Encode&Encryption Algorithm	aes,xor,ror,base64,rc4,des,lznt1 cast,3des,lzo
Communication Protocols	http,https,ftp,smtp,pop3,dns.
Data Object	desktop,clipboard,directory, exchange,gmail,outlook,mailbox, keystroke,keylogger,password.

In the TTPs element correlation section, we have proven that some types of elements may co-occur in the specific TTP description context. Therefore, transforming the TTP element vectors into 2D matrices can express the spatial relationship of co-occurring TTPs elements in the matrix. We use CNN to obtain the spatial features of TTP elements in the matrix, which cannot be obtained by 1D TTP element vectors and the fully connected layer. The $Elmf_n$ is computed from:

$$Elmf_n = \sigma(W_k \cdot Elms_n + b) \tag{5}$$

where σ is the ReLU activation function and W_k denotes different filters.



We compare the feature extraction accuracy, recall, and F1 of using the fully connected layer or the CNN layer in the evaluation section. The result shows that the 2D element matrix with CNN performs better than the 1D vector with a fully connected layer. The element correlation heat map (Fig. 2) also proves that there is a co-occurrence relationship among TTP elements.

At the end of the element feature path, we use two different max pooling to handle the feature vector from two different CNN filters. The max-pooling layer lowers the feature dimension and reserves the main features. This path finally outputs a $4 * 1$ dimension vector $Elmfp_n$.

Description Feature Embedding. We use Sentence-Bert (Reimers and Gurevych 2019) to embed the description text into three 768-dimensional vectors CE_n . These vectors capture the word features inside the sentence, and with the sentence-BERT mean-pooling embedding, the embedding vector could represent the sentence and be used for downstream tasks.

Then, we feed the sentence-embedding into stack Bi-LSTM. The output of the stack Bi-LSTM layers computed from:

$$\begin{aligned}
 \vec{h}_j &= LSTM(x_j, \vec{h}_{j-1}) \\
 \overleftarrow{h}_j &= LSTM(x_j, \overleftarrow{h}_{j-1}) \\
 h_j &= [\vec{h}_j, \overleftarrow{h}_j], j \in [1, n]
 \end{aligned}
 \tag{6}$$

where \vec{h}_j and \overleftarrow{h}_j represent the j -th state produced by LSTM from two directions; x_j is the j -th input vector; h_j is the j -th state, and $[\cdot, \cdot]$ represents the concatenation operation. After that, the attention mechanism (Shen and Lee 2016) outputs the weighted summing of the Bi-LSTM output sequence $H = [h_1, h_2, \dots, h_n]$, which is computed from:

$$\begin{aligned}
 \vec{a} &= \sigma(W_2(\tanh W_1 H + B_1) + B_2) \\
 a &= \frac{\vec{a}}{\sum_{j=1}^T \vec{a}_j} \\
 Z &= Att([h_1, h_2, \dots, h_n])
 \end{aligned}
 \tag{7}$$

where Z is the TTPs description representation. Next, we use a fully connected layer to lower the dimension of the attention layer output. The final output of the description feature path is a 128-dimensional vector Z_l .

TTPs classification

After TTP elements feature embedding and TTPs description feature embedding, we obtain the element features $Elmfp_n$ and textual features Z_l . At the end of our TCENet architecture, we concatenate these two feature vectors into a 132-dimensional vector Z_c and use a fully connected layer to output the final vector Z_f . We use the position in the final two-dimensional vector as the class label. If the max

value appears in the first dimension, the prediction result is negative; otherwise, it is positive, which is computed from:

$$pred = \arg \max(Dense([Elmfp_n, Z_l])) \tag{8}$$

We use the cross-entropy as the loss function and use binary relevance to train six different TTP classification models. The loss L is computed from:

$$L = -(\alpha y \log(pred) + \beta(1 - y) \log(1 - pred)) \tag{9}$$

where y is the true label of the TTP description, and $pred$ is the predicted result of our TCENet. α and β are cross-entropy weights used to balance positive and negative train samples. We minimize the loss L to train the TCENet. Algorithm 3 summarizes the training process.

Algorithm 3 TCENet training

Input: Train dataset $TD_i = \{CE_k, Elms_k, y_k\}, k \in [1, n]; n$ is the number of training datasets; i is the i -th type TTPs, and y_k is the true label of the k training sample.
Output: The TCENet model NN_i of the i -th TTPs

- 1: for $k = 1$ to n do
- 2: $Elmf_k \leftarrow CNN(Elms_k)$
- 3: $Elmfp_k \leftarrow maxpooling(Elmf_k)$
- 4: $H_n \leftarrow stack - BiLSTM(CE_k)$
- 5: $Z_k \leftarrow Attention(H_k)$
- 6: $Zl_k \leftarrow FC_l(H_k)$
- 7: $Zc_k \leftarrow Concat(Zl_k, Elmfp_k)$
- 8: $Zf_k \leftarrow FC_f(Zc_k)$
- 9: $pred_k \leftarrow \arg \max(Zf_k)$
- 10: $L \leftarrow WeightedCrossEntropy(pred_k, y_k)$
- 11: Algorithm convergence.
- 12: return NN_i

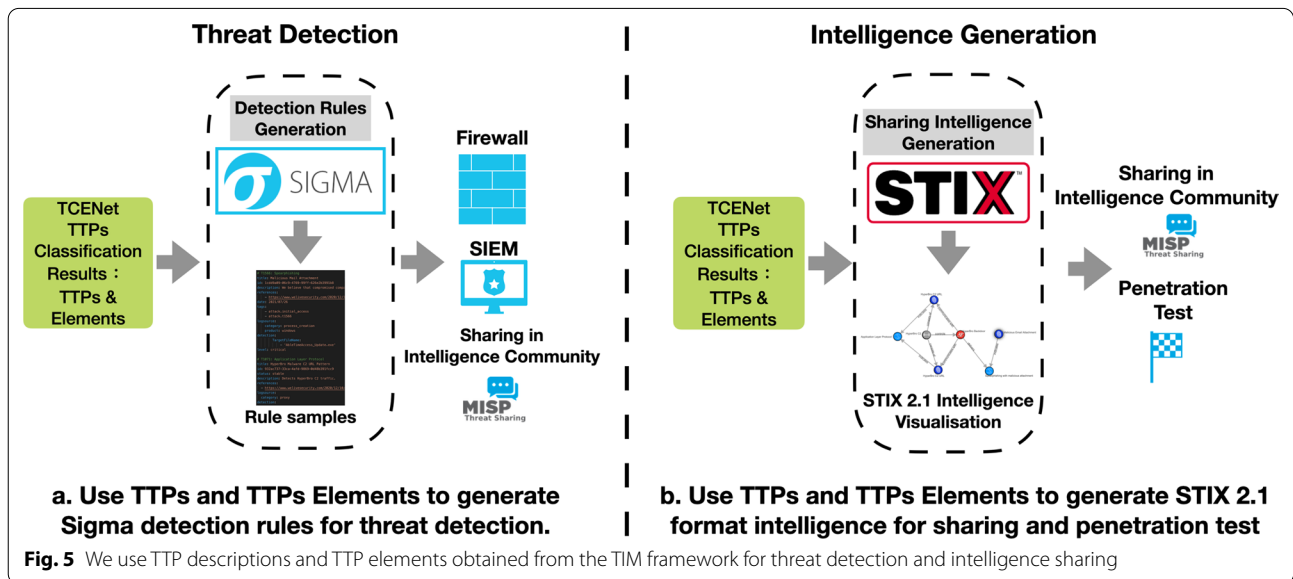
TTP intelligence generation

Based on our proposed TIM framework, we organize the TTP descriptions and TTP elements into Sigma (MSigmaHQ 2021) attack detection rules and shareable intelligence in STIX 2.1 format, as shown in Fig. 5.

Sigma is a generic and open signature format that allows defenders to describe cyber-attack log events. Sigma rules can be used to transform TTPs into search criteria for system logs and SIEM alert events, as well as detection rules for defensive devices such as firewalls to detect threats in the system. Sigma rules can also be used for direct sharing, such as in the MISP intelligence community.

STIX 2.1 (OASIS 2021) is a language and a serialization format used to exchange cyber threat intelligence (CTI). Defenders can also use STIX 2.1 TTP intelligence for penetration testing to simulate attack methods and optimize protection strategies.

As shown in Fig. 5, we organize the TTP description and TTP element information obtained from the TCENet into STIX 2.1 intelligence and Sigma rules for querying relevant threats in the log data of multiple protection devices. The defender can also better grasp the long-period and more essential attack characteristics of the attacker by using TTP intelligence. At the same time, we share the TTP intelligence and Sigma rules in the intelligence community, so that defenders can defend against threats more timely and effectively.



Examples of TTP intelligence in STIX 2.1 format and Sigma detection rules can be found in our anonymous Github repository (TCENet 2021).

Evaluation

In this section, we evaluate the proposed TCENet using our labeled dataset.

Metrics

We evaluate the **precision**, **recall**, and **F1** metrics of the TCENet and other models in comparison experiments and ablation experiments.

TP (True Positives) and TN (True Negatives) denote correctly classified data, while FP (False Positives) and FN (False Negatives) denote misclassified data.

Accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (10)$$

Precision

$$Precision = \frac{TP}{TP + FP} \quad (11)$$

Recall

$$Recall = \frac{TP}{TP + FN} \quad (12)$$

F1

$$F1 = \frac{2 * (Precision * Recall)}{Precision + Recall} \quad (13)$$

Evaluation data

Since the model in this paper uses a binary-relevance method, we construct a negative sample set for each TTP category.

Negative samples

The negative samples consist of the non-TTP descriptions, which are also annotated by annotators, and the TTPs of the other categories. For model training, we use non-TTP descriptions equal to the number of positive samples and other TTP descriptions as negative samples. The negative sample composition is shown in Eq. 14:

$$NSam_j = Non_{TTP} + \sum_{i=1}^{m-1} Other_{TTP}^i \quad (14)$$

The $NSam_j$ in Eq. 14 denotes the negative sample numbers of the j_{th} type TTPs. Non_{TTP} denotes the number of non-TTP descriptions, which is equal to the number of positive samples of the j_{th} type TTPs. $Other_{TTP}^i$ denotes the number of positive samples of the i_{th} type TTP descriptions, where $i \neq j$. The m denotes all 6 categories of TTPs. Since we do not use positive samples of the j_{th} type TTPs as its negative samples, only positive samples of other $m-1$ TTP categories are used as negative samples. Table 7 shows the number of positive and negative samples for the six types of TTP.

Dataset validation result

In the dataset section, we propose using the TTP keyword matching method to validate our dataset. Table 8 shows the matching rate of TTP keywords in both positive and negative samples. Table 8 also shows the accuracy of classifying TTP descriptions directly by TTP keywords.

The results show that the average matching rate of TTP keywords in the positive sample is 92.5%. This indicates that the vast majority of our labeled positive samples are consistent with the keywords mentioned in the TTP descriptions of the ATT&CK website. This means that our dataset is valid and can be used for model training.

However, TTP keywords cannot be directly used to classify TTP descriptions. The matching result also shows TTP keywords can also match many TTP description negative samples. Meanwhile, due to the limited nature of keyword enumeration, not all samples of TTP

Table 7 The positive and negative sample number of six TTPs

TTPs	Positive Samples	Negative: Non TTPs Samples	Negative: Other TTPs Samples
Phishing	2599	2599	3462
Scheduled Task/Job	451	451	5610
Obfuscated Files or Information	439	439	5622
Deobfuscate/Decode Files or Information	475	475	5586
Collection	1401	1401	4660
Application Layer Protocol	696	696	5365

Table 8 Matching rate of TTP keywords in the dataset

TTPs	Pos-hit	Neg-hit	Accuracy
Phishing	0.91	0.17	0.873
Scheduled Task/Job	0.988	0.22	0.884
Obfuscated Files or Information	0.902	0.529	0.6865
Deobfuscate/Decode Files or Information	0.92	0.33	0.795
Collection	0.879	0.351	0.764
Application Layer Protocol	0.945	0.102	0.9215
Average	0.925	0.283	0.820

descriptions can be covered by keywords. Moreover, if we directly classify the TTP descriptions using TTP keywords, it would introduce 28.3% false positives.

Therefore, training a deep learning model for TTP classification can address the limitations of TTP keyword enumeration and also identify the false positive samples that are easily confused by the keyword matching approach. Our subsequent experiments show that our TCENet achieves an accuracy of 0.94 on 6 TTP classifications, which is much higher than the accuracy of keyword-based TTP classification (0.82).

Baseline model

The models we chose for comparison can be divided into four categories: document-level methods from previous work, machine learning methods based on static word embeddings, deep learning methods based on static word embeddings, and deep learning methods based on pre-trained models. Ayoade et al. (2018) and Legoy (2019) both use TF-IDF with an SVM classifier to classify TTPs at the document level. Li et al. (2019) leverage latent semantic indexing to compare the targeting analysis articles with ATT&CK description articles and use SVM with the cosine similarity to classify TTPs. Machine learning methods based on static word embedding include: Doc2Vec (Le and Mikolov 2014) with Linear SVC, Doc2Vec with Decision Tree (DT), Doc2Vec with random forest (RF). Deep learning methods based on static word embedding include: FastText (Joulin et al. 2016), TextCNN (Rakhlin 2016) with GloVe word embedding (Pennington et al. 2014), Bi-LSTM + Attention with GloVe word embedding. Methods based on pre-trained models include Bert-CLS and our proposed TCENet.

Train settings

We grid search for the best performance hyperparameter of our TCENet and other baseline models. Table 9 shows the results of our experiments on the hidden layer size and layer number of the Bi-LSTM network.

Based on the experimental results, we finally used a 3-layer Bi-LSTM and a hidden layer size of 200. The other hyperparameters are shown in Table 10. For cross-entropy weights α and β in Eq. 9, we use the inverse ratio of positive and negative samples as the weight to train the model.

Overall results

We evaluate the overall accuracy of our TCENet on all six TTPs. We divide each TTP-labeled dataset into training, validation, and testing sets according to a 7:1:2 ratio. We train each model for 80 epochs. Table 11 shows the overall accuracy on six TTPs by using TCENet. The **phishing** classification model achieves the best performance because it has the largest dataset (2599 positive samples). The accuracies of **obfuscated files or information** and **deobfuscate/decode Files or information** are 0.92 and 0.916, respectively, because they have a smaller annotated dataset (439, 475, respectively).

Comparison evaluation

Table 12 shows the **precision, recall, and F1** of TCENet compared to the three previous methods and six baseline models. Comparison evaluation is performed on the **Phishing** TTP data.

The result shows that our TCENet method achieves the best performance on all three metrics, and sentence-level

Table 9 LSTM hidden size and layer number evaluation experiment on TTPs Phishing, using F1 score

Hidden Node Size	LSTM layer number		
	1	2	3
50	0.939	0.941	0.944
100	0.939	0.970	0.944
200	0.950	0.941	0.971
300	0.968	0.969	0.950

Table 10 Model parameter settings

Parameters	Setting
Sentence Length	300
SentenceBert Vector Size	768
LSTM Hidden Layer Size	200
LSTM layers	3
Batch size	32
Epochs	80
Learning rate	1e ⁻³
Cross-Entropy Weight	Inverse ratio of positive and negative samples

Table 11 Overall TTPs classification accuracy by using our TCENet on six different TTPs

TTPs	Accuracy
Phishing	0.972
Scheduled Task/Job	0.934
Obfuscated Files or Information	0.920
Deobfuscate/Decode Files or Information	0.916
Collection	0.964
Application Layer Protocol	0.943
Average Accuracy	0.941

The average accuracy is noted in bold font

methods are obviously more accurate than document-level methods. The results also indicate that methods based on the pretrained language model perform better than static word embedding methods such as GloVe and fastText. Language models such as BERT and its variant Sentence-BERT consider the context features of a word and generate dynamic word embedding compared to static word embedding methods using the co-occurrence matrix.

Three Doc2Vec based baseline models achieve approximate results. The method using the random forest (RF) classifier performs better than the linear SVC and decision tree (DT).

FastText considers the n-gram features of words and achieves the best precision among the three static word embedding models. BiLSTM+Attention considers the temporal features of text and uses the attention mechanism to determine the weights of context and achieves the best recall among the three static word vector models. TextCNN uses multiple convolution kernels to capture the spatial features of the text and achieves the best F1 score among the three static word embedding models.

Our TCENet and the mainstream BERT-CLS model performed better than the above baseline. TCENet outperforms BERT by 3-4% on three metrics. TCENet uses a pretrained model, considers the differences between contextual sentences, and assigns weights to contexts using bidirectional LSTM and attention. Additionally, TCENet uses TTP element features to enhance the classification effect.

We then conducted ablation experiments to explore the effects of text features and TTP element features on the final classification results.

Ablation experiment

To demonstrate the effectiveness of each component of TCENet, we perform an ablation experiment. We evaluate the classification accuracy results using only TTP element features, only text features, and TCENet variants, as shown in Table 13. For TCENet variants, we change or remove different parts of TCENet to prove the validity of each part, e.g., using different neural networks to extract the text or the TTP element features, or not using TTP elements.

We first evaluate the TTP classification performance of the TCENet model without considering text features using only TTP element features and CNN. We denote this model as **Only TTPs Elms.** in Table 13.

The **TCENet w/o Elms with CNN** model in Table 13 uses Sentence-BERT for text embedding and CNN to extract context textual features without any elements features. The **TCENet w/o Elms with BiLSTM** model uses BiLSTM to extract context textual features without elements features. The **TCENet with FC_E** (TCENet with a fully connected layer for element features) uses contextual text features and TTP element features for TTP classification. It uses FC for TTP element features extracting. The **TCENet with FC_C** (TCENet with a fully connected

Table 12 The comparison evaluation of the TCENet on the phishing TTP

Text level	Model	Precision	Recall	F1
Document-level	Ayoade et al. (2018) & Legoy (2019)	0.437	0.500	0.608
	Li et al. (2019)	0.444	0.509	0.547
Sentence-level	Doc2Vec + linear SVC	0.859	0.881	0.870
	Doc2Vec + DT	0.853	0.857	0.855
	Doc2Vec + RF	0.895	0.902	0.899
	Bi-LSTM + Attention (GloVe)	0.871	0.923	0.896
	TextCNN (GloVe)	0.913	0.914	0.927
	fastText	0.936	0.895	0.915
	BERT-CLS	0.940	0.935	0.935
	TCENet	0.970	0.973	0.971

The best classification result is shown in bold font

Table 13 Ablation experiments on the phishing TTP

Model	Component	Precision	Recall	F1
Only TTPs Elms.	CNN	0.566	0.497	0.339
TCENet w/o Elms. with CNN	Sentence-BERT + CNN	0.915	0.954	0.934
TCENet w/o Elms. with BiLSTM.	Sentence-BERT + BiLSTM	0.941	0.950	0.945
TCENet with FC_E	Sentence-BERT + FC + BiLSTM	0.940	0.965	0.952
TCENet with FC_C	Sentence-BERT + CNN + FC	0.949	0.953	0.951
TCENet	Sentence-BERT + CNN + BiLSTM	0.970	0.973	0.971

The best result is noted in bold font

layer for context features) uses FC rather than BiLSTM to capture the TTP description context feature and uses CNN for the TTP elements feature.

Our **TCENet** uses both context textual features and TTP element features. It leverages the Bi-LSTM to capture description context features and uses CNN to extract TTP element features. Inspired by Nataraj et al. (2011), who transformed binary files into matrices, the TCENet transforms 1D TTP element vectors into 2D TTP element matrices and uses CNN to extract the spatial features of TTP element co-occurrences implicitly in the matrices.

The results show that both context description features and TTP element features improve the TTP classification performance. The result also shows that the model cannot perform effective classification when using only TTP element features. Without TTP elements features, the two TCENet variants (w/o Elms. models in Table 13) drops 3-4 % compared with TCENet. The TCENet with FC_E and the TCENet with FC_C leverage FC to extract TTP element features and context features. These two TCNet variants cannot better capture the text and TTP element features using FC than TCENet.

Therefore, TCENet obtains the best evaluation results using CNN to extract elemental features and BiLSTM to extract contextual features.

Few-shot evaluation

Some ATT&CK TTPs may have only a small amount of description text. Therefore, we performed a few-shot evaluation on the **obfuscated files or information** TTPs dataset, which has the least data.

In this experiment, we divided the positive sample data into training and test data a ratio of 8:2. We then keep reducing the positive sample training data (from 350 to 50) to evaluate the performance of different models in the case of few-shot training samples. The results are shown in Fig. 6.

From 350 to 50 training samples, Doc2Vec+RF and FastText’s performance drops sharply on 200 samples.

TextCNN also drops sharply on 50 samples, which obtains only a 0.638 accuracy score.

In this experiment, the results of BERT-CLS and the TCENet w/o Elms are similar. Without element features, the performance of TCENet w/o Elms, is also influenced by the number of training samples when it drops to 100.

Our TCENet method achieves the most stable performance on small training sets and achieves **0.857** accurate performance even when the training dataset drops to 50 samples.

The results demonstrate that our TCENet can be generalized to most TTP classification tasks, even in the few-shot training data case.

Annotation cost reduction

TTP data annotation requires expert knowledge and is time-consuming. However, there is no available dataset for sentence-level TTP description, which also hinders research in TTP classification. Therefore, our dataset is necessary and valuable.

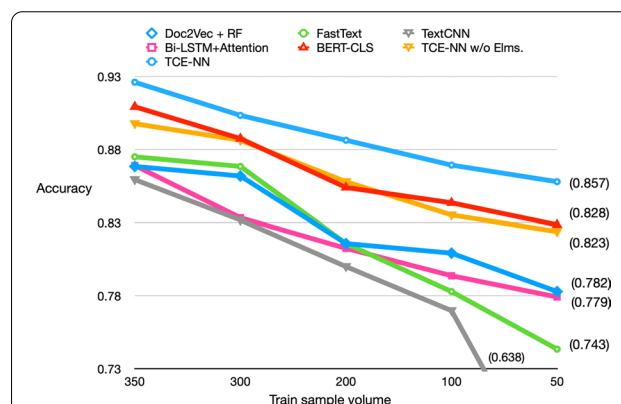


Fig. 6 Few-shot case evaluation on TTPs **Obfuscated Files or Information**. With the reduction of training samples, our TCENet still achieves considerable performance (0.875), even after training with 50 samples

We annotated a total of 10761 security articles with 6061 TTP descriptions in 6 TTP categories. Based on the experience of annotation and the results of our proposed method, we believe that we can reduce the annotation cost and extend the approach of this paper to other TTP annotation tasks in two ways.

To reduce the time cost of TTP annotation, the annotation process can utilize the aforementioned TTP keyword matching method to prioritize the annotation of sentences containing TTP keywords in the security analysis reports. The annotator only needs to confirm whether the matched descriptions are false positives. False alarm data can be used as negative sample data for model training data. Therefore, the annotator does not need to read the full analysis report to obtain the TTP description data.

The few-shot evaluation experiment (shown in Fig. 6) shows that our TCENet achieves an accuracy of 0.857 even for 50 training data samples and 0.93 for 350 training data samples. Therefore, we believe that the absolute number of data annotations can be reduced when the TCENet model is extended to other TTP classification tasks.

Limitations

In this paper, we use a sliding window of size 3 to obtain the TTP description, and the annotators keep only these three sentences in the final dataset when annotating the data. However, we find that some TTP elements may be outside the sliding window, so some elements in the TTP element association heat map show weak associations with TTPs, such as Phishing-URL and Scheduled Task/Job-Regkey. These TTP elements and TTPs that theoretically have strong correlations may also show weak correlations in the heat map (Fig. 2) due to insufficient data. In future work, we will retain longer contextual information to introduce more TTP element features in TTP classification.

Conclusions

In this work, we propose a threat context-enhanced TTP intelligence mining framework named TIM to mine TTP intelligence from unstructured threat data. This framework uses TCENet to classify sentences in security analysis reports for TTP intelligence by using threat context features consisting of TTP descriptions and TTP elements. TCENet achieve an average of 0.94 classification accuracies on 6 types of TTP data and achieves the best performance compared with previous document-level methods and mainstream text classification methods. The TTP element features promote overall performance by 2-3%. Our TCENet achieves

considerable performance (0.875) even in the case of few-shot training samples, which means our proposed method could be generalized to classify most ATT&CK TTPs with a few training data.

Our TIM framework finally organizes the TTP description and the TTP elements into STIX 2.1 intelligence format and Sigma attack detection rules. TTP intelligence and sigma detection rules can be used to attack simulation and threat detection and greatly benefit security defenders for better protection in enterprise security operations centers.

In the future, we will find the relationship of TTPs and their elements in the global document to solve the limitations of this work. We will also expand our dataset and use our proposed TCENet on all ATT&CK TTPs. With TTP intelligence and other cybersecurity entities, we will build a cyber threat knowledge graph to go deeper into APT attack campaigns in a more grand threat context.

Acknowledgements

The authors would like to thank all the annotators and anonymous reviewers for their useful comments and suggestions.

Authors' contributions

YY designed the data collection, model structure, experiments and drafted the manuscript. JJ, ZJ and XW made crucial contributions the experimental design and manuscript revised. PY, BL, HF and NL participated in problem discussions and manuscript revised. All authors read and approved the final manuscript.

Funding

Our research was supported by the National Key Research and Development Program of China (Grant No. 2018YFC0824801, No. 2019QY1302) and the National Natural Science Foundation of China (No. 61802404). This research was also partially supported by the Key Laboratory of Network Assessment Technology, the Chinese Academy of Sciences, and the Beijing Key Laboratory of Network Security and Protection Technology.

Availability of data and materials

We will open-source the full dataset after the paper is published. The demo model and data samples of the model can be found at the following anonymous GitHub address: <https://github.com/TCENet/TCENet>.

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

¹Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China. ²School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100029, China. ³College of Information Engineering, Capital Normal University, Beijing 100048, China. ⁴Beijing Electronic Science and Technology Institute, Beijing 102627, China.

Received: 30 August 2021 Accepted: 13 December 2021

Published online: 01 February 2022

References

Ayoade G, Chandra S, Khan L, Hamlen K, Thuraisingha, B (2018) Automated threat report classification over multi-source data. In: 2018 IEEE 4th

- international conference on collaboration and internet computing (CIC). IEEE, pp 236–245
- cmu-sei (2021) Cyobstrack github repository. [EB/OL]. <https://github.com/cmu-sei/cyobstrack> Accessed August 24, 2021
- DavidJBianco (2021) The Pyramid of Pain. [EB/OL]. <https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html> Accessed August 24, 2021
- Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint [arXiv:1810.04805](https://arxiv.org/abs/1810.04805)
- ESET (2021) Welivesecurity website. [EB/OL]. <https://www.welivesecurity.com/category/malware/> Accessed August 24, 2021
- Husari G, Al-Shaer E, Ahmed M, Chu B, Niu X (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, pp 103–115
- Husari G, Niu X, Chu B, Al-Shaer E (2018) Using entropy and mutual information to extract threat actions from cyber threat intelligence. In: 2018 IEEE international conference on intelligence and security informatics (ISI). IEEE, pp 1–6
- Joulin A, Grave E, Bojanowski P, Douze M, Jégou H, Mikolov T (2016) Fasttext.zip: Compressing text classification models. arXiv preprint [arXiv:1612.03651](https://arxiv.org/abs/1612.03651)
- Le QV, Mikolov T (2014) Distributed representations of sentences and documents. [arXiv:1405.4053](https://arxiv.org/abs/1405.4053)
- Legoy VSM (2019) Retrieving att&ck tactics and techniques in cyber threat reports. Master's thesis, University of Twente
- Li M, Zheng R, Liu L, Yang P (2019) Extraction of threat actions from threat-related articles using multi-label machine learning classification method. In: 2019 2nd international conference on safety produce informatization (IICSPI). IEEE, pp 428–431
- Malwarebytes (2021) Malwarebytes website. [EB/OL]. <https://resources.malwarebytes.com/#analyst-reports> Accessed August 24, 2021
- MITRE (2021) MITRE ATT&CK. [EB/OL]. <https://attack.mitre.or> Accessed August 24, 2021
- MSigmaHQ (2021) Generic Signature Format for SIEM Systems. [EB/OL] <https://github.com/SigmaHQ/sigma> Accessed August 24, 2021
- Nataraj L, Karthikeyan S, Jacob G, Manjunath BS (2011) Malware images: visualization and automatic classification. In: Proceedings of the 8th international symposium on visualization for cyber security, pp 1–7
- Niakanlahiji A, Wei J, Chu B-T (2018) A natural language processing based trend analysis of advanced persistent threat techniques. In: 2018 IEEE international conference on big data (Big Data). IEEE, pp 2995–3000
- OASIS (2021) Introduction to STIX. [EB/OL] <https://oasis-open.github.io/cti-documentation/stix/intro> Accessed August 24, 2021
- Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543
- Rakhlina A (2016) Convolutional neural networks for sentence classification. GitHub
- Reimers N, Gurevych I (2019) Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint [arXiv:1908.10084](https://arxiv.org/abs/1908.10084)
- Richardson L (2021) BeautifulSoup. [EB/OL]. <https://www.crummy.com/software/BeautifulSoup> Accessed August 24, 2021
- Securelist (2021) Securelist website. [EB/OL]. <https://securelist.com/category/apt-reports/> Accessed August 24, 2021
- Shen S-s, Lee H-y (2016) Neural attention models for sequence classification: Analysis and application to key term extraction and dialogue act detection. arXiv preprint [arXiv:1604.00077](https://arxiv.org/abs/1604.00077)
- Tartare M (2021) Operation StealthyTrident: corporate software under attack. [EB/OL]. <https://www.welivesecurity.com/2020/12/10/luckymouse-ta428-compromise-able-desktop/> Accessed August 24, 2021
- TCENet (2021) TCENet Repository. [EB/OL]. <https://github.com/TCENet/TCENet> Accessed August 24, 2021
- Threatpost (2021) Trendmicro website. [EB/OL]. <https://threatpost.com/category/malware-2/> Accessed August 24, 2021
- Trendmicro (2021) Trendmicro website. [EB/OL]. <https://blog.trendmicro.com/trendlabs-security-intelligence/category/malware> Accessed August 24, 2021
- Zhu Z, Dumitras T (2018) Chainsmith: automatically learning the semantics of malicious campaigns by mining threat intelligence reports. In: 2018

IEEE European symposium on security and privacy (EuroS&P). IEEE, pp 458–472

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen® journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)
