Computers & Security 120 (2022) 102763

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Vulcan: Automatic extraction and analysis of cyber threat intelligence from unstructured text

Hyeonseong Jo[a], Yongjae Lee[b], Seungwon Shin[c,*]

[a] Graduate School of Information Security, School of Computing, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon 34141, Republic of Korea
[b] S2W Inc., 12, Pangyoyeok-ro 192beon-gil, Bundang-gu, Seongnam-si, Gyeonggi-do, 13524 Republic of Korea
[c] School of Electrical Engineering, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon 34141, Republic of Korea

ARTICLE INFO

Article history:
Received 10 January 2022
Revised 23 April 2022
Accepted 17 May 2022
Available online 25 May 2022

Keywords:
Cyber threat intelligence
CTI
Cybersecurity
Information extraction
Language model

ABSTRACT

To counteract the rapidly evolving cyber threats, many research efforts have been made to design cyber threat intelligence (CTI) systems that extract CTI data from publicly available sources. Specifically, indicators of compromise (IOC), such as file hash and IP address, receives the most attention among security researchers. However, the ability of IOC-centric CTI systems to understand and detect threats remains questionable for two reasons. First, IOCs are forensic artifacts that indicate that an endpoint or network has been compromised. They cannot depict the technical details of threats. Second, attackers frequently change infrastructure and static indicators, which makes IOCs have a very short lifespan. Therefore, when designing a CTI system, we should turn our attention to other types of CTI data that are helpful in threat understanding and detection (e.g., attack vector, tool).

In this work, we propose Vulcan, a novel CTI system that extracts descriptive or static CTI data from unstructured text and determines their semantic relationships. To do this, we design a neural language model-based named entity recognition (NER) and relation extraction (RE) models tailored for cybersecurity domain. The experimental results confirm that Vulcan is highly accurate with an average F_1-score of 0.972 and 0.985 for NER and RE tasks, respectively. Vulcan also provides an environment where security practitioners can develop applications for threat analysis. To prove the applicability of Vulcan, we introduce two applications, evolution identification and threat profiling. The applications save time and labor costs to analyze cyber threats and show the detailed characteristics of the threats.

1. Introduction

In these days, cyber threats are becoming more complex and sophisticated (EMBROKER, 2022). Attackers continuously seek possible attack vectors, from persuasion techniques that make users voluntarily execute malicious files to zero-day attacks that exploit previously unknown vulnerabilities. Furthermore, a stealthy technique that hides an attacker's identity in trusted benign processes is being actively developed and employed (Wang et al., 2020). To counteract those ever-evolving threats, security researchers collect massive information of threats mentioned in the Web and analyze them to understand current and emerging attack trends, which is called cyber threat intelligence (CTI). The accumulated CTI data from various sources enables security practitioners to have visibility into the threat landscapes and find the clues for threat detection that are not seen in isolated sources.

Given its importance, many research efforts are being made to develop CTI systems that collect, and analyze CTI data (Liao et al., 2016; Zhao et al., 2020a; Zhu and Dumitras, 2018). Liao et al. proposed iACE that collects indicators of compromise (IOC), one type of CTI data, and revealed the relationships across seemingly unrelated cyber threats (Liao et al., 2016). Zhu et al. suggested ChainSmith to extract IOCs from security articles and classify them into four malware delivery stages (Zhu and Dumitras, 2018). Zhao et al. presented TIMiner, a novel framework for CTI extraction, that gathers IOCs and identifies their target domains (e.g., finance, education) Zhao et al. (2020a).

However, while many CTI systems have been proposed in academia (Kurogome et al., 2019; Liao et al., 2016; Zhao et al., 2020a; Zhu and Dumitras, 2018), they give all their attention to IOC (e.g., file hash, IP address) that is one of the types of CTI data. The problem is that IOCs are not effective CTI data to understand and detect threats for the following two reasons. First, IOCs are basically used as signs to determine whether devices are infected by threats. They cannot describe the technical details of

* Corresponding author.
E-mail address: claude@kaist.ac.kr (S. Shin).

https://doi.org/10.1016/j.cose.2022.102763
0167-4048/

threats, which makes it impossible for defenders to gain a holistic view of threats (Zhao et al., 2020b). Second, attackers frequently change their indicators, such as MD5 hashes of malicious files and IPs of botnets, to evade detection. Consequently, IOCs are valid for a very short period of time within 2 days (Iklody et al., 2018; Liao et al., 2016). However, it may take up to several months to perform the tasks from IOC collection to distribution (Radar, 2022), which makes threat detection difficult. To understand and detect fast-evolving threats, CTI systems should cover diverse types of CTI data other than IOCs, which describe the detailed characteristics of threats or are difficult for attackers to change (e.g., attack vector, tool) Cybereason (2022).

To derive descriptive or static CTI data from unstructured text, there are several information extraction (IE) methods to choose from (e.g., dictionary-based Cook and Jensen, 2019, rule-based Liao et al., 2016; Zhu and Dumitras, 2018, deep learning-based Dong et al., 2019; Zhao et al., 2020a). However, there are several challenges to utilize these IE methods to extract the CTI data of our interest. First, since dictionary-based methods use a list of predefined terms, they cannot capture previously unseen CTI data from unstructured text, which yields high false positive and negative rates (Cook and Jensen, 2019). Second, unlike IOCs, the CTI data of our interest are represented by any combination of letters and numbers. It is thus challenging to design patterns or rules for the CTI data. Third, deep learning-based methods could give a satisfactory performance when enough training data is available. Thus, it is time-consuming and labor-intensive to cover multiple types of CTI data.

Vulcan To address the challenges mentioned earlier, we present a novel CTI system, Vulcan, that is responsible for the entire process from CTI data collection to utilization. To extract CTI data and their semantic relationships from unstructured text, we fine-tune a pretrained language model for the IE task. The IE task is mainly performed by two components: (1) threat entity identifier (TEI) and (2) threat relation identifier (TRI). First, the named entity recognition (NER) model, TEI, seeks to locate and classify CTI data mentioned in unstructured text into predefined categories (e.g., attack-vector, tool). Then, the relation extraction (RE) model, TRI, determines the semantic relationships between the CTI data identified by TEI. The goal of these models is to learn contextual information of words that correspond to CTI data from sentence structures and extract previously unseen CTI data and their relationships.

Vulcan stores the extracted CTI data in a graph database and provides security practitioners with search APIs to access the database, which makes it easy for them to develop applications for threat analysis. Currently, we support several basic search functionalities such as finding the entities that have a *y*-relationship with an entity *x*, but plan to support complex search functionalities in the near future.

Evaluation Vulcan collects a massive amount of unstructured text from publicly available sources including ThreatPost, and Malwarebytes. The entire dataset consists of 540 K articles (roughly 11 M sentences) about cyber threats. For evaluation of Vulcan, we manually annotate 6791 CTI data and 4323 relationships between the CTI data. The experimental result shows that Vulcan is highly accurate with an average F_1-score of 0.972 in identifying CTI data from unstructured text and 0.985 in determining the relationships between the CTI data. Besides, Vulcan achieves better performance compared to existing NER and RE systems. To prove the applicability of Vulcan, we design two applications using the search APIs provided by Vulcan, evolution timeline and threat profiling. Implementing an application to understand the evolution of a ransomware family requires only 24 lines (see Fig. 9 for details). The applications save a lot of time and labor costs to analyze the characteristics of threats and uncover the hidden facts about cyber threats that were not revealed by pieces of information located in different sources.

Contribution In short, the contributions of this work are summarized as follows:

- We present Vulcan to identify various types of CTI data and their semantic relationships from unstructured text. More specifically, we introduce customized designs to language model-based NER and RE models.
- For the collected CTI data, Vulcan provides several kinds of search APIs to security practitioners, which enables them to utilize CTI data to analyze current and emerging threats from many perspectives. To prove the applicability of Vulcan, we develop two applications based on the search APIs, evolution identification and threat profiling.

2. Background

2.1. Cyber threat intelligence

As mentioned earlier, CTI is a collection of data that delineates the characteristics of cyber threats, which enables security practitioners to understand cyber threats and establish defense strategies against the threats. Among many types of CTI data, indicators of compromise (IOCs) have gained the most attention from CTI related researches. As a kind of fingerprint, IOCs are used to identify malicious activity on a system or network. For example, IOCs include IP addresses used by malicious actors and MD5 hashes of malware files. These CTI data could be collected from diverse sources, such as private corporations, public technical blogs, or online forums. Examples of public sources for CTI collection include the sites of MalwareBytes, AlienVault, and FireEye.

The understanding of cyber threats could be made more accurate through CTI sharing between organizations since each of them cannot see all aspects of threats. To make this possible, several proposals have been made, such as STIX (2022); TAXII (2022). TAXII, defines protocols for exchanging CTI data, while STIX is a common language standard for that data, including cyber threat incidents. Additionally, some threat intelligence platforms, including MISP Wagner et al. (2016) and OpenCTI (2022), have been presented to effectively collect, store, and manage CTI data about cyber threats.

2.2. Information extraction

Articles posted on public sources are powerful information sources that describe various aspects of events. On the other hand, they are written in free text format and contain information that people are not interested in. Besides, a machine cannot process text data in raw form. Therefore, we need to extract structured data from unstructured textual data via a process called *information extraction (IE)*. The IE task consists of two subtasks: (1) named entity recognition (NER) and (2) relation extraction (RE). The NER model seeks to locate named entities mentioned in a sentence and classify them into predefined categories. Then, the RE model determines semantic relationships between the named entities recognized by the NER model. As shown in Fig. 1, the NER model assigns 'Jaff' and 'emails' to ransomware and attack-vector labels, respectively; and, the RE model assigns `spread_via` label to the relation between the words 'Jaff' and 'emails.' These kinds of structured data extracted by IE task enables various types of data driven analysis, such as hidden relationship detection and trend mining.

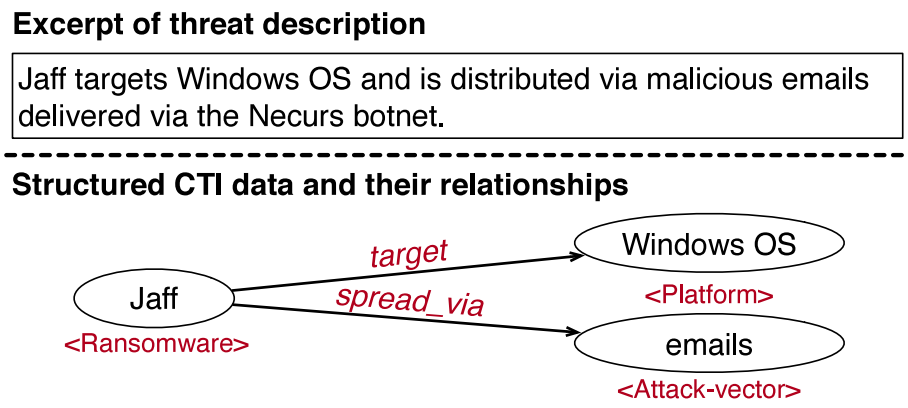

Fig. 1. The excerpt of threat description written in free text format and the structured CTI data extracted from the excerpt via Vulcan.

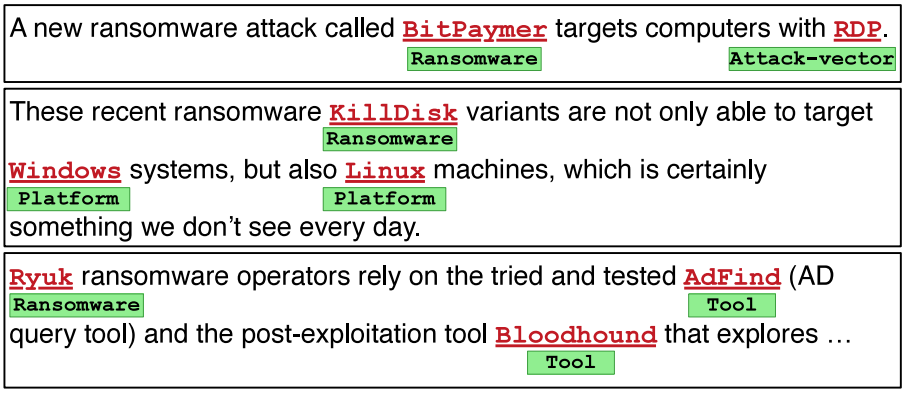

Fig. 2. Examples of sentences including multiple types of CTI data.

3. Problem statement

In this section, we first present the problem of existing CTI systems and the scope of our work. Then, we explain the technical challenges of extracting CTI data from unstructured text.

3.1. Research goal

As cyber threat intelligence (CTI) has been increasingly adopted by security practitioners to analyze and respond to emerging cyber threats, many CTI systems have been proposed to collect CTI data from unstructured text (e.g., threat reports, articles) Kurogome et al. (2019); Liao et al. (2016); Zhao et al. (2020a); Zhu and Dumitras (2018). Specifically, most of them focus on collecting a particular type of CTI data, indicators of compromise (IOCs) Sauerwein et al. (2017). However, there are still doubts about their effectiveness for threat understanding and detection. First, as indicators to identify whether attacks occurred or not, IOCs hardly depict a comprehensive view of threats. Second, since fast-evolving threats frequently change infrastructure and static indicators, IOCs have a very short useful lifespan within 2 days (Liao et al., 2016). It is thus difficult for IOC-centric CTI systems to detect sophisticated and fast-evolving threats.

To address the shortcoming of existing CTI systems, we aim to collect diverse types of CTI data that describe the tactics, techniques, and procedures (TTP) of attacks, as shown in Fig. 2. The first sentence describes what the *target platform* of Killdisk is. And, the two subsequent sentences explain the *attack vector* and *tools* used for ransomware attacks, respectively. Unlike IOCs, since attackers have a limited number of available techniques, they cannot easily change the techniques they use (Cybereason, 2022). Besides, developing an entirely new technique requires considerable time and system knowledge. Therefore, these TTP-related CTI data can help us to detect and understand cyber threats. Fig. 3 shows CTI data and their relationships to be covered in this work. Among various types of threats (e.g., denial of service, account hijacking), we focus on extracting CTI data pertaining to ransomware attacks.

- **Ransomware.** A ransomware is a form of malware designed to encrypt all the files on a benign device and demand a ransom for decryption.
- **Attack-vector.** Attackers leverage multiple attack vectors to deliver ransomware into a benign device. Examples include drive-by download, phishing, and removable media.
- **Vulnerability.** A vulnerability is a publicly disclosed security flaw in softwares. Here, we cover both CVE identifiers and Microsoft Security Bulletins.
- **Platform.** A platform refers to various operating systems including Windows, Linux, and Android, and these are the main targets of ransomware attacks.
- **Algorithm.** Once a ransomware arrives at a benign device, it silently encrypts a set of files using encryption algorithms.
- **Tool.** Once a ransomware arrives at a benign device, it performs malicious behaviors (e.g., lateral movement) using multiple types of tools. Here, we address multiples types of tools from Windows built-in to open source.
- **Behavior.** A behavior refers to malicious techniques performed by malware including ransomware.
- **Date.** A date describes when a ransomware was first discovered and is used to analyze the characteristics of ransomware over time.

A relation is an ordered pair of CTI data where the type of relation is determined by the types of CTI data that have a relationship. Most of CTI data and their relationships addressed in this

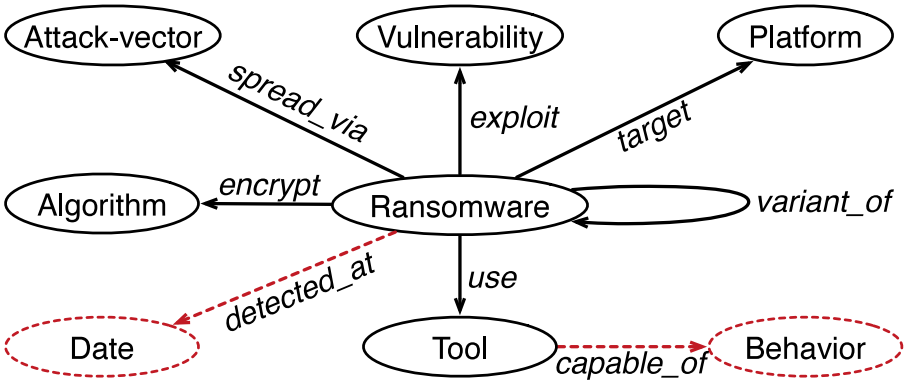

Fig. 3. Structured CTI data and their relationships covered in this work. The black nodes (entities) and edges (relations) are extracted by Vulcan, and the red dotted ones are imported from external sources. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

work are extracted by Vulcan. To improve situational awareness against cyber threats, some types of CTI data and their relationships are imported from external sources that provide structured CTI data (e.g., MITRE, 2022; NVD, 2022).

3.2. Technical challenges

Many previous CTI studies proposed their own methods to extract IOCs from unstructured textual data (e.g., dictionary-based Cook and Jensen, 2019, rule-based Liao et al., 2016; Zhu and Dumitras, 2018, deep learning-based Dong et al., 2019; Zhao et al., 2020a). However, there are several challenges to utilize these methods to identify the CTI data of our interest.

C-1) Since dictionary-based methods use a list of predefined terms, they cannot recognize previously unseen CTI data. Furthermore, they cannot distinguish between the CTI data that have the same name but mean different things (see Section 4.4). Therefore, it is inappropriate to use dictionary-based methods to extract the CTI data from unstructured text.

C-2) Unlike IOCs that tend to follow fixed patterns, the CTI data of our interest are expressed by any combination of characters and digits. Besides, some of them consist of multiple words, such as DMA Locker and Cobalt Strike. Consequently, it is infeasible to design patterns like regular expressions for the CTI data of our interest.

C-3) While deep learning-based methods can learn the contextual information of words, its learning process requires a large amount of labeled data to train deep neural networks. Therefore, it is inappropriate to cover multiple types of CTI data.

C-4) Most existing methods focus on extracting isolated IOC without the consideration of their relationships. However, to understand the characteristics of threats in-depth, the semantic relationships between CTI data should also be considered.

4. The design of Vulcan

To address the challenges mentioned earlier, we develop a novel CTI system, Vulcan, that performs the entire process from gathering unstructured data to utilizing the extracted CTI data to analyze the technical details of threats. Fig. 4 shows Vulcan's overall architecture and workflow. We first present the overall workflow of Vulcan and then explain in detail how each component works.

4.1. Overall workflow

As shown in Fig. 4, Vulcan is divided into two parts: CTI data collection, and CTI data usage. The data collection part consists of five components: ❶ data scraper, ❷ pre-processor, ❸ threat entity identifier, ❹ entity linker, and ❺ threat relation identifier. Each component operates sequentially by taking the output of the previous one as input.

The operation of Vulcan starts with the collection of textual data that is likely to include CTI data from multiple sources (e.g., threat report, article) via the data scraper. Second, the pre-processor filters the collected data to select only those related to ransomware attacks. Third, the threat entity identifier (TEI) extracts the entities from textual data. For example, let us consider the following sentence: "*Revil has been distributed via malicious e-mail attachment.*" TEI assigns two words, Revil and e-mail, as entity types, `RANSOM` and `VECTOR`, representing ransomware and attack-vector, respectively. Fourth, the entity linker performs two functions, entity segmentation and entity integration, to assign unique identities to the entities recognized by TEI. The entity segmentation step decouples the entities with the same name that refer to different things. For example, although the Magniber ransomware family has multiple variants, they are all referred to by the same name, 'Magniber.' Thus, the Magniber entity is separated into different things based on the unique characteristics of the variants. The entity integration step, on the other hand, connects the entities with different names that refer to the same thing (e.g., MailTo, NetWalker, and Koko). Finally, the threat relation identifier (TRI) determines the semantic relationships of the entities within the same sentence. As shown in Fig. 4, the relation between the entity pair (Revil, e-mail) is assigned with `spread_via` type.

Vulcan stores the collected CTI data and their relationships in a graph database and provides security practitioners with search APIs to access the database. These APIs enable them to develop various types of applications for threat analysis. Here, we introduce two applications built with search APIs, evolution identification, and threat profiling.

4.2. Data scraper & pre-processor

Data scraper The data scraper is a kind of crawler that collects threat-related textual data written in natural language from multiple websites. Specifically, the data scraper starts with collecting the links of articles until there is no more link for each site. For each article link, it then extracts the headline, date of writing, content, and domain name. In the case of Twitter, the data scraper collects the tweets including the 'Ransomware' hashtag. Some kinds of websites provide structured threat summary before describing detailed explanation The data scraper also collects these data as a resource for threat analysis.

Pre-processor For the collected articles via the data scrapper, the pre-processor performs two text processing tasks, topic filter-

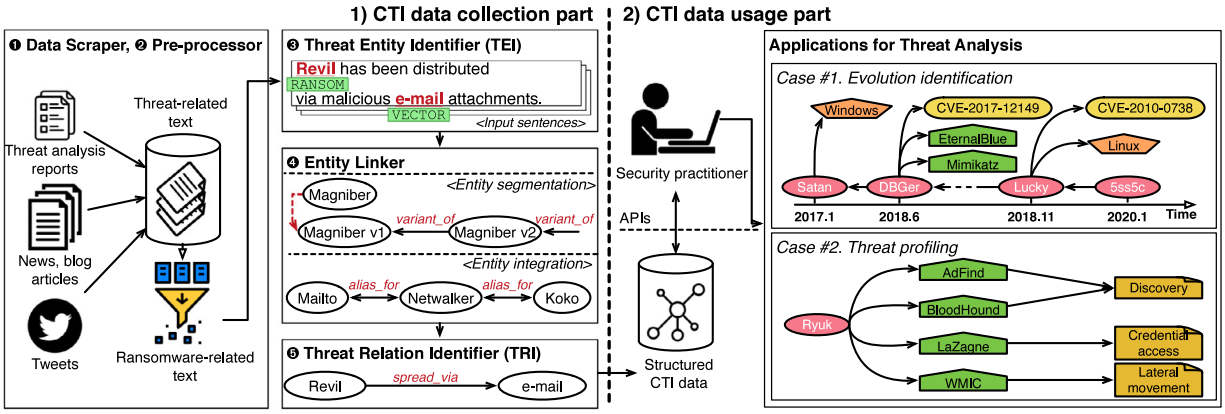

Fig. 4. Vulcan's overall architecture and workflow.

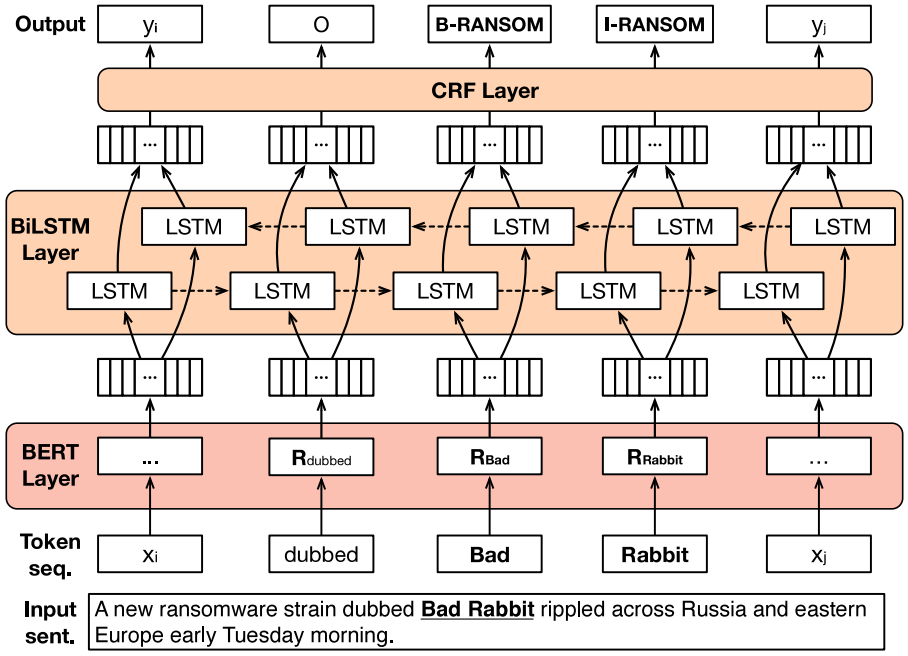

Fig. 5. The architecture of the threat entity identifier.

ing and text sanitization. First, it sort out the articles dealing with ransomware attack. To do this, the collected articles pass through a filter with a set of ransomware-related keywords. Examples of keywords include 'encrypt', 'decrypt', 'demand', and 'ransom.' Second, since text processing is largely dependent on the quality of input text (Roy et al., 2020), we need to tidy up the articles to get a clean text. To do this, the pre-processor removes noisy characters, such as HTML tags and white spaces, from the articles using a set of regular expressions. Finally, sentence tokenization is applied to each article to divide the articles into lists of sentences.

4.3. Threat entity identifier (TEI)

Many previous studies on named entity recognition (NER) have been proposed in natural language processing (NLP) community. Habibi et al. (2017); Ju et al. (2018); Lample et al. (2016). Besides, there are multiple open source NER tools available (e.g., CoreNLP Manning et al., 2014, SpaCy Honnibal, 2017). However, since the NER task is highly dependent on the target domain, it is difficult for NER models designed for one domain to work well in other domains (Settles, 2005). Besides, the entities of our interest do not follow any patterns, and previously unseen entities, such as names of ransomware and tool, frequently emerge. Thus, existing NER techniques based on dictionaries or rules cannot capture the ransomware-related entities from textual data. To solve these issues, we design a language model-based NER model, threat entity identifier (TEI), tailored for ransomware attacks. Specifically, we develop TEI by fine-tuning a well-known language model BERT (Devlin et al., 2018). The goal of TEI is to learn contextual information of words using information from both sides of words in a text.

Fig. 5 shows the architecture of TEI. On top of the BERT model, we stack a bi-directional long short-term memory (BiLSTM) layer and a conditional random field (CRF) layer. Before training TEI, we convert an input sentence into a sequence of tokens that is the format of the BERT model. Then, the final hidden state vectors from the BERT model for the tokens are fed into the BiLSTM layer. The BiLSTM layer extracts the features from both directions, forward and backward, of the token sequence. Next, the outputs of the BiLSTM layer are fed into the CRF layer to decode the best label sequence. Finally, given an input sequence of tokens, $(x_1, x_2, \ldots, x_n)$, TEI outputs the sequence of labels $(y_1, y_2, \ldots, y_n)$, where y_i corresponds to one of the following labels: ❶ ransomware, ❷ attack-vector, ❸ vulnerability, ❹ platform, ❺ algorithm, ❻ tool, and ❼

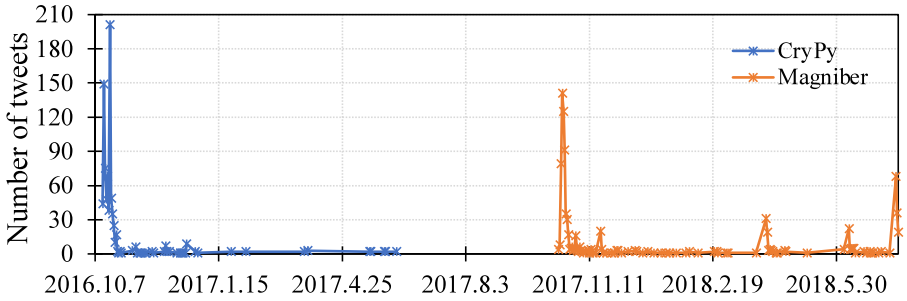

Fig. 6. The number of tweets containing the keywords, CryPy or Magniber, over time.

other. Note that *other* denotes that the given token does not belong to any of the preceding 6 ransomware-related entity types.

4.4. Entity linker

TEI can identify named entities related to ransomware attacks from unstructured text. However, NER models including TEI have a common limitation that they cannot understand the meanings beyond literal representations of entities. As a result, the following two problems may occur: (*i*) a single entity often refers to different things, or (*ii*) multiple entities with different names indicate the same thing. To overcome these problems, the entity linker conducts two types of operations, entity segmentation, and entity integration.

Entity segmentation The first problem occurs when entities are not *clearly* mentioned in textual data. For example, consider the following sentence: *"Magniber ransomware came back stronger in July 2018."* From the sentence, TEI identifies the word 'Magniber' as 'ransomware.' However, the entity Magniber corresponds to the second version of the Magniber ransomware family. Its first version was first discovered in Oct. 2017. Although these two versions of ransomware have the same name, they are technically distinct from each other. Therefore, we should find such entities and separate them into different things.

To reduce the burden of entity segmentation, we first pick out candidate entities that are likely to mention several ransomware variants based on bursts of tweets. The number of tweets tends to increase rapidly at the time the events occur (Mizunuma et al., 2014). As shown in Fig. 6, the number of tweets containing the keyword 'CryPy' follows a decreasing trend after a burst of tweets occurs once. By contrast, the number of tweets containing the keyword 'Magniber' soars several times over a long time span. It is thus likely that new variants with the same name as Magniber appeared. From this intuition, we consider the keywords showing two or more bursts as candidate entities. Here, the bursts correspond to local maxima whose values (i.e., the number of tweets) are more than 25.

For each candidate entity, we then analyze the differences of context specific terms mentioned in the tweets of burst points. The ransomware attacks are usually described with technical terms, such as tools and vulnerabilities (Shin et al., 2020). Thus, if the burst points deal with distinct ransomware variants, they are likely to contain different technical terms. To extract these terms from tweets, we utilize TEI designed in Section 4.3. Then, the terms that refer to the same thing are replaced with representative expressions. For example, 'e-mail', and 'attachment' are replaced by 'email.' We then check whether previously unseen technical terms are included in burst points. If so, we separate a given candidate entity into several different entities with version number based on the chronological order of burst points (e.g., Magniber v1.0, Magniber v2.0).

Entity integration The second problem occurs from the fact that some entities are mentioned with multiple names (i.e., alias) in the cybersecurity domain. For example, two ransomware entities with different names, Revil and Sodinokibi, indicate the same ransomware, and two different types of entities, EternalBlue and CVE-2017-0144, refer to the same vulnerability. These entities that refer to the same thing are used interchangeably across various sources (e.g., articles, blogs). However, since TEI distinguishes entities based on their names, it considers the entities with different names as different things. Therefore, we need to link these entities that indicate the same thing.

To address this problem, we should construct a gazetteer (i.e., a dictionary consisting of alias pairs (a_1, a_2) where a_1 is an alias for a_2). Following the approaches utilized in several domains (Hu et al., 2019; Strobl et al., 2020), we collect threat-related terms and their aliases from multiple sources that provide structured CTI data (e.g., Jo et al., 2020; ThaiCERT, 2022). Then, for each alias pair (a_1, a_2), we add a `alias_for` relation between the entities with the same names as the names of the alias pair. In this work, we regard the entities connected by `alias_for` relations as the same thing. And, when we search for a specific entity from the extracted CTI data, the entities that have an alias relation with an input entity are also considered.

4.5. Threat relation identifier (TRI)

TEI identifies ransomware-related entities from input sentences and classifies them into predefined categories, such as ransomware and tool. For the identified entities, we then need to extract their semantic relationships via a process called relation extraction (RE). The RE task is formulated as a pair-wise classification problem that determines the categories of all possible pairs of entities (e_1, e_2) within the same sentence. To perform this task, we design a language model-based RE model, threat relation identifier (TRI). Specifically, we fine-tune the BERT model for the RE task (Wu and He, 2019).

As shown in Fig. 7, TRI takes a sentence and an entity pair as inputs and assigns the relationships between the entity pair with one of the following labels: ❶ variant_of(e_1, e_2), ❷ variant_of(e_2, e_1), ❸ spread_via, ❹ exploit, ❺ target, ❻ encrypt, ❼ use, and ❽ other. Note that since the `variant_of` type consists of two entities of the same type, (ransomware, ransomware), it is subdivided into two types according to the direction of the relation. And, *other* means that the relation between two given entities does not belong to any of the preceding 7 relation types. If three or more entities are contained in a single sentence, we compose all possible entity pairs that could establish semantic relations and consider them as different inputs. Before training TRI, we perform preprocessing on input sentences to make TRI better learn contextual information about the relationships between entities. First, we insert special markers (i.e., < e1 >, < /e1 >, < e2 >, and < /e2 >) at the beginnings and ends of two input entities, which could im-

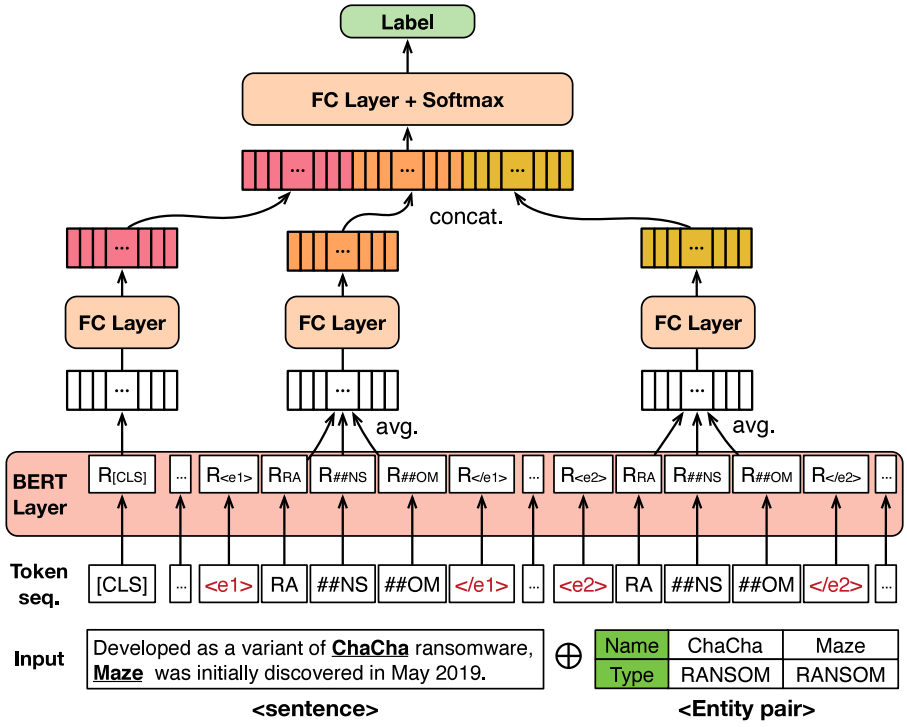

Fig. 7. The architecture of the threat relation identifier.

prove the performance of relation extraction (Soares et al., 2019). Second, two entities mentioned in the input sentences are masked with the names of their types to prevent overfitting of TRI. The preprocessed sentences are then transformed into the format used by the BERT model. Each sentence is tokenized into a sequence of tokens by the WordPiece tokenizer (Sennrich et al., 2015) that splits a word into several sub-words included in the predefined vocabulary. Next, the [CLS] token is appended to the beginning of the token sequence.

To determine the type of relation between two given entities, TRI utilizes final hidden state vectors of the BERT model for [CLS] and two entity types. The reason for using the vector of [CLS] token is that it contains sentence-level contextual information. Since some entity types are tokenized into multiple tokens, they are represented by multiple hidden state vectors. We thus apply the average operation to the hidden state vectors for each entity type. Then, the three vectors for [CLS] and two entity types are fed into fully connected (FC) layers, separately. The outputs of the FC layers are concatenated and fed into another FC layer followed by a softmax layer. Finally, TRI identifies the type of relation between the input entities. For example, in Fig. 7, TRI assigns the `variant_of(e2,e1)` type between the entity pair (ChaCha, Maze), which means that Maze is a variant of the ChaCha ransomware.

5. Evaluation

5.1. Settings

In this work, we conduct all of the experiments on a server with 40 of Intel Xeon E5-2630 CPUs, 12 of 16 GB memories, and 4 TITAN Xp. Here, we explain the dataset and the language model used for experiments. We also describe data integration with existing sources that provide structured CTI data to increase Vulcan's data coverage.

Dataset To evaluate our system, Vulcan, we build a ground-truth dataset that consists of the sentences annotated with 6,791 entities and 4,323 relations between the entities. To generate a dataset, we invited five graduate students in computer science to perform the annotation task. The annotation task involves two steps. First, we manually label the words of a sentence with one of the entity types defined in Fig. 3. The words annotated in this step are used to train and evaluate the threat entity identifier (TEI). Second, we label the semantic relationship between the annotated entities. The relations annotated in this step are used to train and evaluate the threat relation identifier (TRI). Fig. 8 shows examples of annotated sentences. Each sentence contains one or more relationships between the named entities.

In addition to the relation types for ransomware attacks, we consider the negative samples of relations corresponding to the `other` type to make TRI distinguish the relations of our interest defined in Fig. 3 from the others. For example, consider the following sentence: "*As far as WannaCry is concerned, it would not infect Mac system.*" Here, we assign the `other` type between the entity pair (WannaCry, Mac).

Considering that some types of entities consist of multiple words, we label the words using the BIO tagging scheme that assigns `B` to the beginning of named entities, `I` to the inside, and `O` to the others. For example, as shown in Fig. 8, the two words 'Cobalt' and 'Strike,' constituting the single entity 'Cobalt Strike,' are annotated with `B-TOOL` and `I-TOOL`, respectively.

Language model There are many language models based on deep neural networks including BERT (Devlin et al., 2018), GPT-3 (Brown et al., 2020), and XLNet (Yang et al., 2019). Since these models are pretrained on a large corpus of textual data, their outputs contain rich contextual information of words. For example, the BERT model is pretrained with English Wikipedia (2500 M words) and BookCorpus (800 M words) based on masked language modeling (MLM) and next sentence prediction (NSP). As a result, various NLP tasks based on the pretrained language models, including question answering, natural language inference, show state-of-the-art results. The choice of which neural language model to use depends on the objective (e.g., performance, inference speed) and target NLP task. As an example, DistilBERT

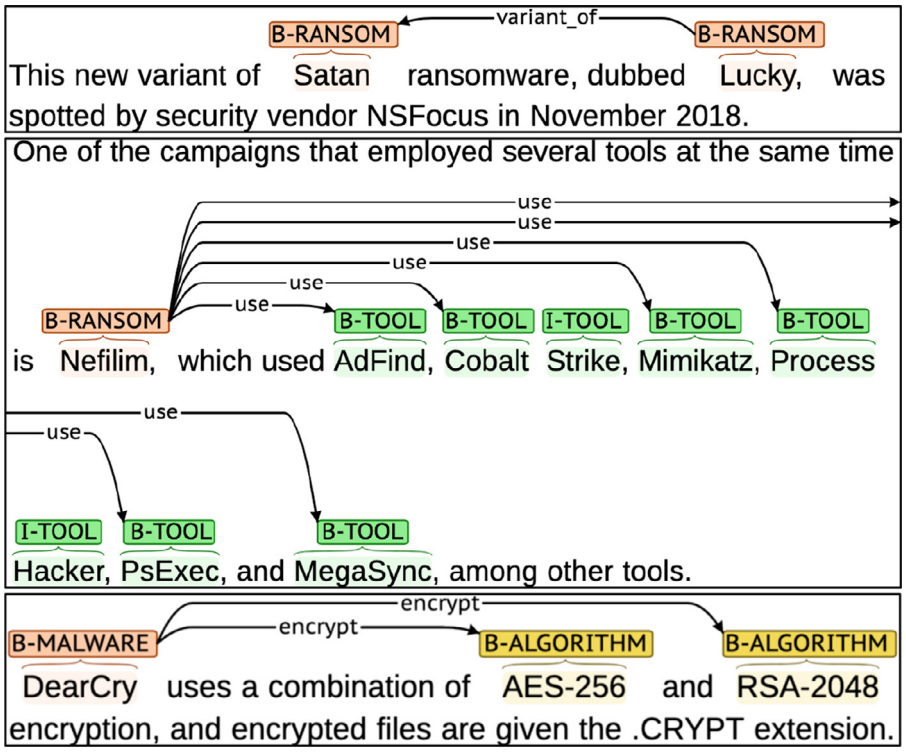

Fig. 8. Examples of annotated sentences.

(Sanh et al., 2019), a smaller version of BERT (Devlin et al., 2018), shows slightly lower performance than BERT but guarantees faster inference speed. Therefore, we select BERT, which has multiple versions suitable for different environments, as a base neural language model.

Integration with external sources One way to gain a deeper and broader understanding of cyber threats is to collect as much CTI data as possible. To expand the dataset of Vulcan, we gather structured CTI data and their relationships from several external sources (e.g., MITRE, 2022; NVD, 2022) and integrate them into Vulcan. For example, MITRE provides structured relationships between tools and adversarial behaviors, such as (PsExec, lateral-movement), (AdFind, discovery). From the integration of these relationships into Vulcan, we could figure out what kinds of malicious behaviors ransomware does through the relationships of (ransomware, tool) extracted from Vulcan (see details in Section 6.2).

5.2. Performance evaluation

Threat entity identifier (TEI) Given input sentences, TEI extracts the following 6 types of entities: (1) ransomware, (2) attack-vector, (3) vulnerability, (4) platform, (5) algorithm, and (6) tool. Here, we use three metrics to evaluate how accurately TEI classifies the words in the sentences into appropriate entity types: (1) *Precision* is the ratio of the number of entities TEI correctly identified over the total number of entities detected by TEI; (2) *Recall* is the ratio of the number of entities TEI correctly identified over the total number of entities contained in the sentences; (3) F_1*-score* is the harmonic mean between precision and recall. In the experiment, the batch size is 16 and the number of epochs is 30. Empirically, we find that all sentences in the ground-truth dataset consist of no more than 128 tokens. Thus, the maximum sequence length of input is set to 128.

We randomly split the ground-truth dataset with a ratio of 7:3 for the training and testing set and repeat the experiment with 5 times to obtain average performance over the trials. Table 1 shows the average precision, recall, and F_1-score for each entity type. TEI detects the entities of our interest with an average precision of 0.968 and an average recall of 0.977. Among the entity types addressed in this work, the vulnerability type yields the highest F_1-score of 1.000, and the attack-vector type shows the lowest F_1-score of 0.951. We observe that the entity types that tend to follow fixed patterns show high performance, such as vulnerability and algorithm. This can be attributed to the fact that the patterns of entity types are crucial features that determine the entity type of a given word. On the other hand, for the entity types such as ransomware and attack vectors, the features that can determine the type do not appear well in a given word. As a result, there is some performance degradation when recognizing ransomware and attack vector.

Table 1
The performance of the threat entity identifier.

Entity type	Precision	Recall	F_1-score
Ransomware	0.960	0.984	0.972
Attack-vector	0.951	0.935	0.943
Vulnerability	1.000	1.000	1.000
Platform	0.972	0.984	0.978
Algorithm	0.983	0.983	0.983
Tool	0.973	0.944	0.958
Average	**0.968**	**0.977**	**0.972**

Threat relation identifier (TRI) We then evaluate the performance of TRI to understand how accurately TRI determines the semantic relationships between the entities. Here, the entities are correctly labeled, and we focus on evaluating whether TRI correctly identifies the relationships of the entity pairs. Like the previous experiment, we split the ground-truth dataset with a ratio of 7:3 for the training and testing set for each relation type. The batch size is 32, and the number of epochs is 20.

As shown in Table 2, TRI achieves high performance with an average precision of 0.981 and an average recall of 0.990. The ex-

```
def getRansomEvolution(input):

    output = {}
    ransoms = [input]

    variants = getReachableEntities(start=input, r_type='
     variant_of')
    for variant in variants:
        ransoms.append(variant.value)

    for ransom in ransoms:
        date = getEntity(start=ransom, r_type='detected_at
     ')
        output[ransom] = {date.type:date.value}

    output = sorted(output,key=lambda x:(output[x]['date'])
     )

    for ransom in ransoms:
        features = getEntities(start=ransom, dst_type='any
     ')
        for feature in features:
            if (feature.type in output[ransom]):
                output[ransom][feature.type].append(feature
     .value)
            else:
                output[ransom] = {feature.type:[feature.
     value]}

return output
```

Fig. 9. An application built with search APIs provided by Vulcan. The application takes the name of a ransomware as an input and returns its evolution timeline.

Table 2
The performance of the threat relation identifier.

Relation type	Precision	Recall	F_1-score
variant_of(e_1, e_2)	0.980	0.971	0.975
variant_of(e_2, e_1)	0.962	0.962	0.962
spread_via	0.974	0.997	0.985
exploit	1.000	1.000	1.000
target	0.990	0.980	0.985
encrypt	0.972	0.996	0.984
use	0.986	0.995	0.991
Average	**0.981**	**0.990**	**0.985**

Table 3
The performance comparison of Vulcan and existing NER, RE models.

Model	Precision	Recall	F_1-score
Named entity recognition (NER)			
CRF	0.896	0.832	0.861
BiLSTM + CRF	0.950	0.878	0.913
TEI	**0.968**	**0.977**	**0.972**
Relation extraction (RE)			
CNN (w/ CRF)	0.783	0.839	0.810
CNN (w/ BiLSTM + CRF)	0.845	0.881	0.863
CNN (w/ g-truth)	0.905	0.957	0.930
TRI (w/ TEI)	0.959	0.974	0.966
TRI (w/ g-truth)	**0.981**	**0.990**	**0.985**

perimental result confirms that the parameters of TRI are trained to correctly determine the relation type of a given input. Besides, TRI shows high precision and recall rates for distinguishing the `variant_of` type of relations in which there is a direction between entities. This is due to the fact that the words placed between the entities are distinctly different, e.g., $<e_1$, a successor to $e_2>$ for `variant_of` (e_1, e_2), and $<$a variant of e_1, dubbed $e_2>$ for `variant_of`(e_2, e_1). Compared to existing works for relation extraction in the cybersecurity domain (Dong et al., 2019; Jo et al., 2020; Pingle et al., 2019), TRI shows comparable performance while extracting more diverse types of relationships between CTI data.

Baseline comparison To compare the performance of Vulcan with existing NER, RE models, we design baseline NER, RE models and train them with the dataset used in Vulcan. More specifically, we implement two NER models based on CRF and BiLSTM + CRF, respectively, to recognize named entities from unstructured text. Then, we implement a RE model based on convolutional neural network (CNN) (Liu et al., 2013; Zhao et al., 2020a) to determine the semantic relationships between the entities.

Table 3 shows the performance comparison of Vulcan and existing NER, RE models. First, compared with CRF, BiLSTM + CRF models, our NER model, TEI, shows better performance in terms of both precision and recall. Second, we compare the performance of two RE models, CNN and TRI, with two different types of datasets, (1)

Table 4
Summary of search APIs.

API	Description
attachToDB()	connect to graph database provided by Vulcan
detachFromDB()	disconnect to graph database provided by Vulcan
getEntities()	return a list of entities of a given type
getRelations()	return a list of relations that have a relationship with a given entity
getEntityType()	return an entity type of a given entity
getRelationType()	return a relation type of a given entity pair
getReachableEntities()	return all entities reachable from a given entity

the ground-truth (g-truth) dataset and (2) the outputs of the NER models (i.e., CRF, BiLSTM + CRF, and TEI). The g-truth dataset consists of *correctly* labeled entities and their relationships. It is used to evaluate how well the RE models classify the relations between the entities. On the other hand, the outputs of the NER models include several *misclassified* entities and their relationships. It is used to evaluate the end-to-end performance of the RE models.

With the g-truth dataset, both the CNN model and TRI show high performance with a precision and recall of more than 0.9. And, TRI shows the highest performance among 5 different experimental cases. On the other hand, the performance of the RE models drops when using the outputs of the NER models. Especially, the CNN model with the output of the CRF model yields the worst performance with a precision of 0.783 and a recall of 0.839.

The performance degradation occurs because the errors of the NER models are inevitably cascaded to the RE models. The misclassification of the NER models makes the RE models unable to identify correct relationships of entity pairs. For example, a NER model fails to recognize the ransomware entity Magniber from the following sentence: *"PC users in South Korea are seriously threatened by the Magniber ransomware which hackers distribute via the Magnitude Exploit Kit."* Then, the entity pair (Magniber, Magnitude) that have a `use` relation cannot be configured. Therefore, both NER and RE models should provide high precision and recall to extract correct relations between entities.

6. Applications for threat analysis

Our proposed CTI system, Vulcan, extracted lots of CTI data and their relationships from unstructured data. To provide an environment where security practitioners can develop applications for threat analysis, we implement a set of search APIs to access the collected CTI data. Table 4 lists a summary of search APIs supported by Vulcan. Currently, we offer basic functionalities such as finding the entities that have a y-relationship with an entity x but plan to support complex functionalities in the near future. With the supported APIs, security practitioners can develop applications they need for threat analysis. In this section, we present two applications that help understand and detect ever-evolving threats: evolution identification, and threat profiling.

6.1. Evolution identification

One of the useful functionality for which Vulcan can be used is understanding how ransomware attacks have technically evolved over time. For security practitioners, this is an imperative step in establishing defense strategies to combat ransomware attacks.

Fig. 9 shows the code of an application that takes a ransomware entity as an input parameter and returns its evolution timeline. To do this, we first find all reachable ransomware entities (variants) from an input entity while the type of relation between the entities corresponds to a '`variant_of`.' Next, we sort the entities chronologically by the time they were detected. Finally, we extract any types of entities connected to each ransomware entity to extract the features of ransomware entities.

Figs. 10 and 11 show the evolution timelines of the two ransomware families, Satan and Magniber, that show what features have been added to each variant. Note that the directions of the edges between ransomware entities are reversed to indicate the direction of evolution. And, the features that previously appeared are not marked for the visibility of the timeline.

Satan ransomware family After the initial version of the Satan ransomware family was first discovered in January 2017, its three variants (i.e., DBGer, Lucky, and 5ss5c) emerged over three years. As with most ransomware families, Satan employs a hybrid encryption scheme with a combination of AES and RSA algorithms to guarantee the security and performance of encryption. All its variants also utilize the same encryption scheme.

A notable change to the Satan ransomware family is its ability to propagate to other devices in a network. The second version of Satan, DBGer, starts utilizing two tools, Mimikatz and EternalBlue, and two vulnerabilities, CVE-2017-12149 and CVE-2017-10271. The DBGer variant dumps all login credentials from an infected device via Mimikatz. It also scans the device's local network to find other devices exposed to the EternalBlue vulnerability (i.e., CVE-2017-0144) or the other two vulnerabilities (i.e., CVE-2017-12149, CVE-2017-10271). Then, it attempts to compromise devices using stolen credentials or one of the three vulnerabilities.

The third version of Satan, Lucky, adds multiple vulnerabilities discovered in several softwares (e.g., JBoss, Tomcat, Samba) to exploit Windows or Linux-based systems. In the case of the 5ss5c variant, there are no newly identified features from Vulcan.

Magniber ransomware family Fig. 11 shows the evolution timeline of the Magniber ransomware family. Note that the dotted edges between the vulnerability and date nodes are imported from NVD (2022). One of the key features of the Magniber family is that they continue to evolve over a period of time. Since the first version appeared in 2017, new variants have appeared until recently. The second feature is that they exploit newly disclosed vulnerabilities shortly after the vulnerabilities are discovered. For example, the PrintNightmare vulnerability (CVE-2021-34527) was added to NVD in July 2021. Then, it took a month for the Magniber family to start exploiting this vulnerability to compromise benign devices. This fact points out that users should install the latest security updates of all software to prevent and minimize the damage caused by ransomware attacks.

6.2. Threat profiling

Threat profiling is a crucial process for security practitioners to detect sophisticated and fast-evolving cyber threats. In particular, a detailed analysis of behavioral elements, which are difficult for attackers to change, is required. To do this, we implement an application that takes two input parameters, a ransomware entity and a feature to analyze, and returns all entities of the input feature reachable from the input entity, including the entities located between them.

Fig. 12 shows the profiling result for malicious behaviors of Ryuk ransomware. From the application's output, we could identify what behaviors ransomware does other than encryption and what tools it utilizes to make these behaviors.

Ryuk ransomware performs four additional behaviors through several tools from open source to Windows built-in. The goal of the first three behaviors (i.e., discovery, credential access, and lateral movement) is to compromise as many devices as possible, thereby maximizing profits. This fact indicates that we need to strengthen security between devices in a network to diminish the propagation of ransomware attacks. The last behavior (i.e., command and control) is used to download additional malware from remote servers

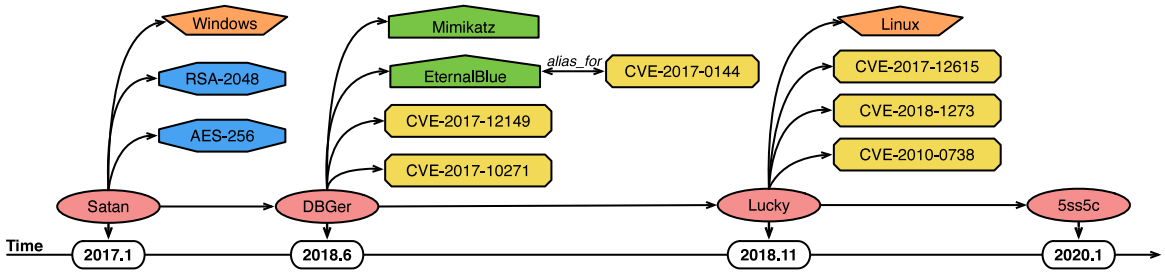

Fig. 10. The evolution timeline of the Satan ransomware family.

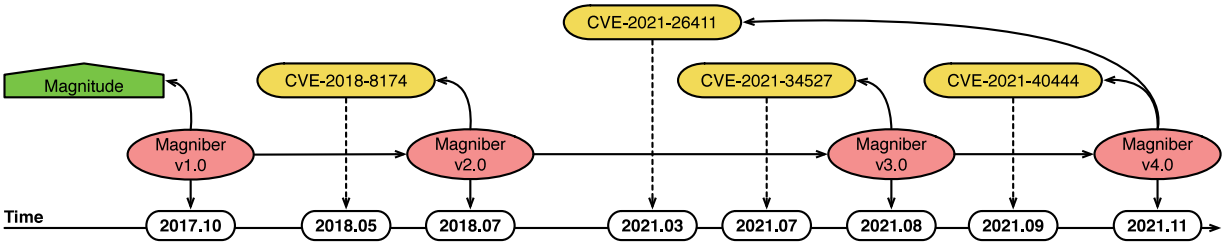

Fig. 11. The evolution timeline of the Magniber ransomware family.

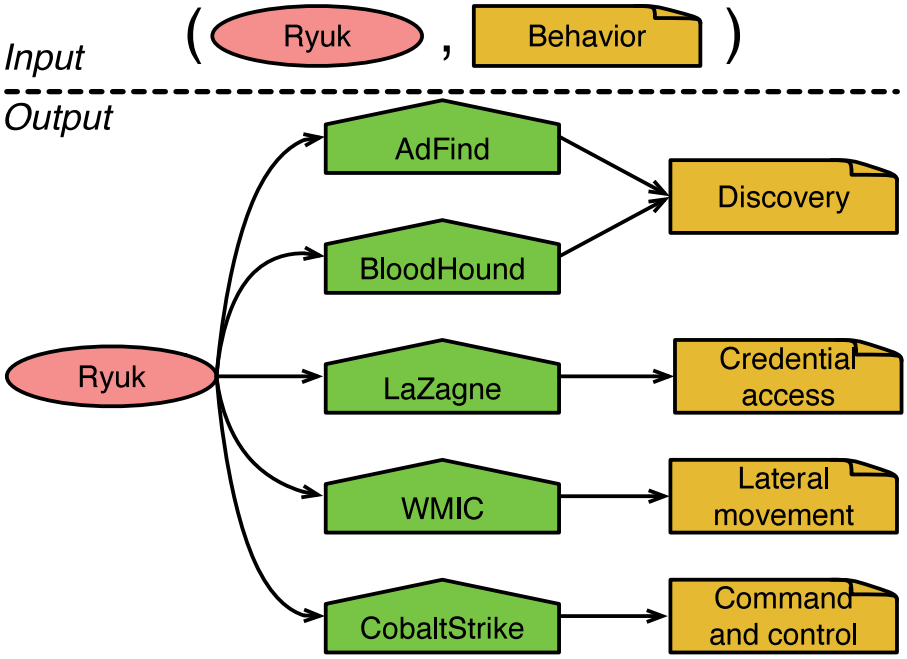

Fig. 12. The profiling result for adversarial behaviors of Ryuk ransomware.

or upload stolen data to attackers. The presence of the tools used to perform these behaviors could be utilized as an indicator to identify potentially vulnerable devices.

7. Discussion

In this work, we present a novel CTI system, Vulcan, that extracts structured CTI data from unstructured textual data with high accuracy. However, there are some areas for further improvement.

Textual data collection The two most crucial factors in textual data collection are (1) coverage and (2) reliability. First, a CTI system should gather as much textual data as possible from publicly available sources since it cannot extract CTI data without textual data.

In this work, the scope of the collection was limited to manually selected security-related sources and Twitter. In the future, we plan to consider more diverse sources, including scientific publications that unveil previously unknown facts.

Second, a CTI system should ensure the reliability of the collected textual data. If the textual data is unreliable, the trustworthiness of the CTI data contained in the textual data may also be denied. Several previous studies have indeed revealed the existence of conflicting textual data in the cybersecurity domain (Dong et al., 2019; Jo et al., 2020). In this case, a CTI system may obtain different CTI data depending on where it collects textual data. Furthermore, adversaries may upload fake textual data to the web to disrupt a CTI system (Ranade et al., 2021). To address this issue, we need to consider designing curation APIs that enable the implementation of existing reliability verification techniques (Yin et al., 2008; Zhang et al., 2018).

CTI data extraction Our proposed system, Vulcan, is trained to identify 6 types of CTI data and 7 types of relationships between

them. It means that Vulcan cannot recognize a new type of CTI data beyond the training scope. To broaden our view of threats, we need to cover more types of CTI data, such as threat actors and victims.

8. Related work

Cyber threat intelligence There have been many studies on cyber threat intelligence (CTI) collection (Catakoglu et al., 2016; Kurogome et al., 2019; Liao et al., 2016; Zhao et al., 2020a; Zhu and Dumitras, 2018). Liao et al. (2016), Zhu and Dumitras (2018), and Zhao et al. (2020a) proposed CTI systems that collect CTI data from unstructured text and performed threat analysis based on the collected CTI data. Catakoglu et al. (2016) presented an automated technique to extract and validate IOCs for web applications. Kurogome et al. (2019) introduced a method to generate interpretable and accurate IOCs based on the results generated by running malware samples in a sandbox environment. There are some studies to design a system that extracts threat actions from unstructured CTI reports (Husari et al., 2017; Zhang et al., 2021). To do this, Husari et al. (2017) analyzed the grammatical structure of a sentence using dependency parsing, while Zhang et al. (2021) analyzed the part of speech (POS) of three elements, (subject, verb, object), that make up threat actions. Our work differs from previous works in that we focus on extracting descriptive or static CTI data other than IOCs from unstructured text. Our work also considers the behavioral relationships between CTI data.

CTI data is also used for malware detection (Gao et al., 2021; Milajerdi et al., 2019; Sabottke et al., 2015; Zhu and Dumitraş, 2016). Sabottke et al. (2015) proposed a threat early warning system with tweets about cyber threats, and, Zhu and Dumitraş (2016) presented an android malware detection system that extracts feature sets from textual data of multiple types of sources (i.e., research paper, article, and document). Milajerdi et al. (2019) proposed an attack detection system using graph alignment between provenance graph and query graph manually built with threat descriptions. Gao et al. (2021) introduced a system that facilitates threat hunting based on threat behavior graphs built with unstructured CTI reports.

Finally, several previous studies have been conducted to quickly detect current and emerging cyber threats using social data, especially Twitter (Khandpur et al., 2017; Shin et al., 2020). Khandpur et al. (2017) proposed a system for identifying a broad range of cyber threats via a new query expansion technique. Shin et al. (2020) presented a novel event detection system based on the monitoring of new and re-emerging words. While these works could detect the emergence of cyber threats, they cannot describe the detailed characteristics of cyber threats.

Information extraction In the cybersecurity domain, there have been some previous works to extract structured relations from unstructured text (Jones et al., 2015; Joshi et al., 2013; Pingle et al., 2019; Satyapanich et al., 2020; Wang and Chow, 2019). Joshi et al. (2013) proposed an automatic framework that extracts vulnerability-related information and links it to an existing knowledge base. Jones et al. (2015) performed named entity recognition through hand-crafted lists (i.e., gazetteers) or regular expressions and relation extraction using a semi-supervised method (i.e., bootstrapping). Wang and Chow (2019) extracted useful threat intelligence using several NLP techniques including POS tagger, and gazetteer. Pingle et al. (2019), and Satyapanich et al. (2020) presented a system to extract semantic triples over cybersecurity texts using deep learning techniques. Compared with previous studies, our work designs NER and RE models based on a pretrained language model, which shows high performance. Besides, our work also provides an environment where security practitioners can develop applications for threat analysis.

9. Conclusion

To date, many previous studies proposed CTI systems that collect CTI data from public sources to gain deep security insights. However, their common problem is that they zoom in on a particular type of CTI data, IOCs. While IOCs are effective in detecting and blocking known threats, they have a very short lifespan and do not describe the technical details of threats. To address this issue, we introduce a novel CTI system, Vulcan, that extracts any forms of CTI data and their semantic relationships from unstructured textual data. For the collected CTI data, Vulcan provides security practitioners with search APIs to develop applications for threat analysis. To prove the applicability of Vulcan, we present two applications built with search APIs and reveal the trends of ransomware attacks over a long time span. In the future, we plan to continuously collect articles about emerging threats and extract structured CTI data from them to track the changing trends of threats. Besides, we will support complex search functionalities to provide a better development environment for threat analysis.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Hyeonseong Jo: Conceptualization, Methodology, Writing – original draft. **Yongjae Lee:** Conceptualization, Writing – original draft. **Seungwon Shin:** Conceptualization, Project administration.

Acknowledgment

This research was supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921).

References

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al., 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

Catakoglu, O., Balduzzi, M., Balzarotti, D., 2016. Automatic extraction of indicators of compromise for web applications. In: Proceedings of the 25th International Conference on World Wide Web, pp. 333–343.

Cook, H.V., Jensen, L.J., 2019. A guide to dictionary-based text mining. In: Bioinformatics and Drug Discovery. Springer, pp. 73–89.

Cybereason. The end game: exploiting attacker weak spots with TTP-based detection. http://cyber-360.net/wp-content/uploads/2017/10/The-End-Game-Exploiting-Attacker-Weak-Spots.pdf.

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.

Dong, Y., Guo, W., Chen, Y., Xing, X., Zhang, Y., Wang, G., 2019. Towards the detection of inconsistencies in public security vulnerability reports. In: 28th {USENIX} Security Symposium ({USENIX} Security 19), pp. 869–885.

EMBROKER. 2021 must-know cyber attack statistics and trends. https://www.embroker.com/blog/cyber-attack-statistics/.

Gao, P., Shao, F., Liu, X., Xiao, X., Qin, Z., Xu, F., Mittal, P., Kulkarni, S.R., Song, D., 2021. Enabling efficient cyber threat hunting with cyber threat intelligence. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE). IEEE, pp. 193–204.

Habibi, M., Weber, L., Neves, M., Wiegandt, D.L., Leser, U., 2017. Deep learning with word embeddings improves biomedical named entity recognition. Bioinformatics 33 (14), i37–i48.

Hu, S., Tan, Z., Zeng, W., Ge, B., Xiao, W., 2019. Entity linking via symmetrical attention-based neural network and entity structural features. Symmetry 11 (4), 453.

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.

Honnibal, Matthew, and Ines Montani. “spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing. GitHub.” (2017).
Iklody, A., Wagener, G., Dulaunoy, A., Mokaddem, S., Wagner, C., 2018. Decaying indicators of compromise. arXiv preprint arXiv:1803.11052.
Jo, H., Kim, J., Porras, P., Yegneswaran, V., Shin, S., 2020. Gapfinder: finding inconsistency of security information from unstructured text. IEEE Trans. Inf. Forensics Secur. 16, 86–99.
Jones, C.L., Bridges, R.A., Huffer, K.M., Goodall, J.R., 2015. Towards a relation extraction framework for cyber-security concepts. In: Proceedings of the 10th Annual Cyber and Information Security Research Conference, pp. 1–4.
Joshi, A., Lal, R., Finin, T., Joshi, A., 2013. Extracting cybersecurity related linked data from text. In: 2013 IEEE Seventh International Conference on Semantic Computing. IEEE, pp. 252–259.
Ju, M., Miwa, M., Ananiadou, S., 2018. A neural layered model for nested named entity recognition. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 1446–1459.
Khandpur, R.P., Ji, T., Jan, S., Wang, G., Lu, C.-T., Ramakrishnan, N., 2017. Crowdsourcing cybersecurity: cyber attack detection using social media. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1049–1057.
Kurogome, Y., Otsuki, Y., Kawakoya, Y., Iwamura, M., Hayashi, S., Mori, T., Sen, K., 2019. Eiger: automated IOC generation for accurate and interpretable endpoint malware detection. In: Proceedings of the 35th Annual Computer Security Applications Conference, pp. 687–701.
Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C., 2016. Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360.
Liao, X., Yuan, K., Wang, X., Li, Z., Xing, L., Beyah, R., 2016. Acing the IOC game: toward automatic discovery and analysis of open-source cyber threat intelligence. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 755–766.
Liu, C., Sun, W., Chao, W., Che, W., 2013. Convolution neural network for relation extraction. In: International Conference on Advanced Data Mining and Applications. Springer, pp. 231–242.
Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D., 2014. The stanford corenlp natural language processing toolkit. In: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, pp. 55–60.
Milajerdi, S.M., Eshete, B., Gjomemo, R., Venkatakrishnan, V., 2019. Poirot: aligning attack behavior with kernel audit records for cyber threat hunting. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 1795–1812.
MITRE. Att&ck. https://attack.mitre.org/.
Mizunuma, Y., Yamamoto, S., Yamaguchi, Y., Ikeuchi, A., Satoh, T., Shimada, S., 2014. Twitter bursts: analysis of their occurrences and classifications. In: Proc. 8th International Conference on Digital Society, pp. 182–187.
NVD,. National vulnerability database. https://nvd.nist.gov/.
OpenCTI,. Open cyber threat intelligence platform. https://www.opencti.io/en/.
Pingle, A., Piplai, A., Mittal, S., Joshi, A., Holt, J., Zak, R., 2019. Relext: relation extraction using deep learning approaches for cybersecurity knowledge graph improvement. In: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 879–886.
Radar,. Indicators of compromise & threat feeds. https://www.btbsecurity.com/hubfs/2021-BTB-RADAR-IOC.pdf.
Ranade, P., Piplai, A., Mittal, S., Joshi, A., Finin, T., 2021. Generating fake cyber threat intelligence using transformer-based models. arXiv preprint arXiv:2102.04351.
Roy, G., Dey, L., Shakir, M., Dasgupta, T., 2020. Learning domain terms-empirical methods to enhance enterprise text analytics performance. In: Proceedings of the 28th International Conference on Computational Linguistics: Industry Track, pp. 190–201.
Sabottke, C., Suciu, O., Dumitraş, T., 2015. Vulnerability disclosure in the age of social media: exploiting twitter for predicting real-world exploits. In: 24th {USENIX} Security Symposium ({USENIX} Security 15), pp. 1041–1056.
Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.
Satyapanich, T., Ferraro, F., Finin, T., 2020. CASIE: extracting cybersecurity event information from text. UMBC Faculty Collection.
Sauerwein, C., Sillaber, C., Mussmann, A., Breu, R., 2017. Threat intelligence sharing platforms: an exploratory study of software vendors and research perspectives.
Sennrich, R., Haddow, B., Birch, A., 2015. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909
Settles, B., 2005. Abner: an open source tool for automatically tagging genes, proteins and other entity names in text. Bioinformatics 21 (14), 3191–3192.
Shin, H., Shim, W., Moon, J., Seo, J.W., Lee, S., Hwang, Y.H., 2020. Cybersecurity event detection with new and re-emerging words. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, pp. 665–678.
Soares, L. B., FitzGerald, N., Ling, J., Kwiatkowski, T., 2019. Matching the blanks: distributional similarity for relation learning. arXiv preprint arXiv:1906.03158.
Strobl, M., Trabelsi, A., Zaiane, O.R., 2020. Wexea: wikipedia exhaustive entity annotation. In: Proceedings of the 12th Language Resources and Evaluation Conference, pp. 1951–1958.
S. 2.0. Structured threat information expression. https://oasis-open.github.io/cti-documentation/stix/intro.html.
TAXII,. Trusted automated exchange of intelligence information. https://oasis-open.github.io/cti-documentation/taxii/intro.html.
ThaiCERT. Threat group cards: a threat actor encyclopedia. https://apt.thaicert.or.th.
Wagner, C., Dulaunoy, A., Wagener, G., Iklody, A., 2016. Misp: the design and implementation of a collaborative threat intelligence sharing platform. In: Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, pp. 49–56.
Wang, Q., Hassan, W.U., Li, D., Jee, K., Yu, X., Zou, K., Rhee, J., Chen, Z., Cheng, W., Gunter, C.A., et al., 2020. You are what you do: hunting stealthy malware via data provenance analysis. NDSS.
Wang, T., Chow, K.P., 2019. Automatic tagging of cyber threat intelligence unstructured data using semantics extraction. In: 2019 IEEE International Conference on Intelligence and Security Informatics (ISI). IEEE, pp. 197–199.
Wu, S., He, Y., 2019. Enriching pre-trained language model with entity information for relation classification. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 2361–2364.
Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q. V., 2019. Xlnet: generalized autoregressive pretraining for language understanding. arXiv preprint arXiv:1906.08237.
Yin, X., Han, J., Philip, S.Y., 2008. Truth discovery with multiple conflicting information providers on the web. IEEE Trans. Knowl. Data Eng. 20 (6), 796–808.
Zhang, H., Li, Y., Ma, F., Gao, J., Su, L., 2018. Texttruth: an unsupervised approach to discover trustworthy information from multi-sourced text data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2729–2737.
Zhang, H., Shen, G., Guo, C., Cui, Y., Jiang, C., 2021. Ex-action: automatically extracting threat actions from cyber threat intelligence report based on multimodal learning. Secur. Commun. Netw. 2021.
Zhao, J., Yan, Q., Li, J., Shao, M., He, Z., Li, B., 2020. Timiner: automatically extracting and analyzing categorized cyber threat intelligence from social data. Comput. Secur. 95, 101867.
Zhao, J., Yan, Q., Liu, X., Li, B., Zuo, G., 2020. Cyber threat intelligence modeling based on heterogeneous graph convolutional network. In: 23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020), pp. 241–256.
Zhu, Z., Dumitraş, T., 2016. Featuresmith: automatically engineering features for malware detection by mining the security literature. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 767–778.
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.

Hyeonseong Jo is a Ph.D. student in the Department of Information Security at KAIST working with Dr. Seungwon Shin in NSS Lab. He received his B.S degree in Computer Engineering from Ajou University in Korea. He received his M.S. degree in Information Security from KAIST. His research interests include SDN security and cyber threat intelligence.

Yongjae Lee received his Ph.D. in Computer Science at KAIST, M.S in Computer Science at KAIST, B.S in Computer Science and Engineering at Chung-ang University in South Korea. He is currently a principal researcher at S2W Inc., where he is pursuing to solve cyber security problems with NLP. His research interests include system and network security, natural language processing, and cyber threat intelligence.

Seungwon Shin is an associate professor in the School of Electrical Engineering at KAIST. He received his Ph.D. degree in Computer Engineering from the Electrical and Computer Engineering Department, Texas A&M University, and his M.S degree and B.S degree from KAIST, both in Electrical and Computer Engineering. Before joining KAIST, he has spent nine years at industry, where he devised several mission critical networking systems. His research interests span the areas of SDN security, IoT security, and Botnet analysis/detection.