Measuring and Modeling Vulnerability Exploitation Using Threat Intelligence Feeds

ABSTRACT

Many organizations rely on Threat Intelligence (TI) feeds to assess the risk associated with security threats. Due to the volume and heterogeneity of data, it is prohibitive to manually analyze the threat information available in different loosely structured TI feeds. Thus, there is a need to develop automated methods to vet and extract actionable information from TI feeds. To this end, we present a machine learning pipeline to automatically detect vulnerability exploitation from TI feeds. We first model threat vocabulary in loosely structured TI feeds using state-of-the-art embedding techniques (Doc2Vec and BERT) and then use it to train a supervised machine learning classifier to detect exploitation of security vulnerabilities. We use our approach to identify exploitation events in 191 different TI feeds. Our longitudinal evaluation shows that it is able to accurately identify exploitation events from TI feeds only using past data for training and even on TI feeds withheld from training. Our proposed approach is useful for a variety of downstream security tasks such as data-driven vulnerability risk assessment.

KEYWORDS

threat intelligence; software vulnerability; risk assessment

1 INTRODUCTION

Threat Intelligence (TI) has become a cornerstone of collaborative data-driven security [14]. Many organizations rely on TI feeds to assess the risk associated with security vulnerabilities and deploy countermeasures in a timely manner. The interest in TI feeds has grown over the last few years, with an increasing amount and variety of TI feeds [5]. TI feeds are now available in many different flavors—from free to commercial, from many different providers ranging from third-party firms or public groups, and targeting various kinds of threats from phishing to botnets.

A key challenge in effectively leveraging TI feeds is coping with the volume of data. Large amounts of new threat information is regularly published, sometimes in excess of millions per day. As the volume of threat information has increased, it has become infeasible to manually sift through them [36]. Thus, automated methods to systematically sift through TI feeds are needed. Another key challenge is that TI feeds are loosely structured due to heterogeneous data sources, software ecosystems, internal processes, maturity of the TI program, and staff expertise. There has been progress over the last decade to standardize information sharing in TI feeds [20, 31].[1] While such standardization mitigates this challenge, a significant fraction of the information in TI feeds is made available within free-form text fields that are manually filled by security experts with differing styles. These challenges make it non-trivial to automatically process the information in TI feeds to extract actionable information.

In this paper, we consider the problem of detecting whether information about the exploitation of vulnerabilities in the wild is present in a given TI feed event. Detection of vulnerability exploitation in the wild is crucial for risk assessment and incident response purposes. Consider, for instance, the Common Vulnerability Scoring System (CVSS) [10], one of the current de facto solutions to parametrize vulnerability risk. While CVSS does provide a subscore track existence of exploits for vulnerabilities, it does not provide a subscore to track exploitation incidents *in the wild*. The recently proposed Exploit Prediction Scoring System (EPSS) [11] aims to bridge this gap by forecasting whether a vulnerability will be exploited following its public disclosure. Thus, information about vulnerability exploitation in the wild can complement CVSS and EPSS based risk assessment strategies. To the best of our knowledge, there are currently no large-scale measurements leveraging TI feeds to assess exploitation in the wild, or tools to leverage those measurements directly for risk assessment purposes.

To address the aforementioned challenges, we propose a machine learning approach to automatically ingest TI feeds for detecting vulnerability exploitation in the wild. Our machine learning pipeline leverages natural language processing (NLP) based embedding techniques to parse and encode loosely structured information in TI feeds. To this end, we consider both static and dynamic embeddings techniques. Static embeddings generate the same embedding for the same vocabulary in different TI feed events: vector matrices which encode vocabulary into vectors are shared across events. Dynamic embeddings, in contrast, aim at capturing vocabulary semantics in different contexts to address the issue of context-dependent nature of information in TI feed events. Specifically, we train a static embedding (Doc2Vec [16]) and dynamic embedding (BERT [7]) in an unsupervised manner to capture the semantics of loosely structured information in different TI feeds. Moreover, we also train purpose-built embeddings that are further trained (or fine-tuned) on data from TI feeds. TI2Vec and TIBERT represent specialized embeddings corresponding to Doc2Vec and BERT, respectively. We then leverage these embeddings along with labeled data to train a supervised classifier for detecting exploitation of security vulnerabilities from TI feeds.

Our dataset comprises events from 191 TI feeds, including data from the default feeds of MISP [21].[2] To assess the effectiveness

[1]The Malware Information Sharing Platform (MISP) [20] and the Structured Threat Information (STIX) [31] are two important initiatives in this direction. Both MISP and STIX establish formats that allow different organizations to share data in a uniform manner.

[2]MISP is an Open Source Threat Information Sharing Platform, which comes pre-configured with some public data feeds. In this work, we also use data from various Cyber Threat Intelligence communities. Information disclosure in such communities is regulated by the Traffic Light Protocol [12]. In this work, we restrict our analysis to tlp:white (=public) events and anonymize the name of the source organizations. By focusing on anonymized tlp:white events we avoid restrictive non-disclosure agreements, we avoid biases and we allow for our results to be reproduced and extended by

Conference'17, July 2017, Washington, DC, USA
2022. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

of different classifiers in detecting exploitation events in the TI feeds, we perform temporal and spatial splits of our dataset. In temporal setting, the training and test sets contain past and future data, respectively. In temporal+spatial setting, the training and test sets are setup such that the TI feed used for testing is set aside during training. While F1 scores vary over different time horizons, they average around 56–78% on average. Overall, we find that dynamic BERT embeddings outperform static Doc2Vec embedding and purpose-built specialized embeddings (i.e., TIBERT, TI2Vec) outperform their pre-trained counterparts.

Our machine learning approach to detect exploitation events from TI feeds is useful for a variety of downstream security tasks. For example, it can help organizations implement risk-aware patch management strategies. In particular, the exploitation events detected by our approach from TI feeds can be used to determine exploit code maturity level impacting CVSS/EPSS scores. Overall, our work suggests the feasibility of using TI feeds to automatically detect vulnerability exploitation in the wild, paving the way towards data-driven enrichment of vulnerability risk assessment.

Our key contributions and findings are summarized below:

(1) We conduct a longitudinal analysis of TI feeds, identifying the prevalence of different categories of events, such as network activity and payload delivery, across multiple organizations. We also analyze the flow of information across TI feeds, identifying that some feeds play the role of sources whereas others behave as aggregators (Section 2).
(2) We curate and label the events in TI feeds, e.g., indicating which of those correspond to exploitation as opposed to patch disclosures or security advisories. To that aim, we mine for association rules between event tags and exploitation. Whereas tags are instrumental for curating the dataset, they require manual intervention, motivating fully automated word embeddings (Section 3).
(3) We capture semantics from loosely structured information in TI feeds using static and dynamic embeddings (Doc2Vec and BERT, respectively and their specialized counterparts TI2Vec and TIBERT) that are then used to train classifiers to automatically detect exploitation of vulnerabilities. The training of the classifiers conforms to temporal constraints, allowing us to assess the relevance of history on the accuracy of future predictions. We also evaluate knowledge transfer across feeds, noting that such transfer is feasible from TI feeds that behave as sources to aggregators, particularly when the envisioned flow of knowledge is aligned with the flow of information across TI feeds (Section 4).

Paper organization: Section 2 describes our dataset of 191 TI feeds and their key characteristics. Section 3 presents our proposed machine learning pipeline to detect vulnerability exploitation in the wild. Section 4 reports the evaluation results, and Section 5 summarizes related work on measurement and modeling of information in TI feeds before concluding.

the scientific community, which is a fundamental step towards a better understanding of the TI feed ecosystem.

Listing 1 An event in .json format from TI Feed 2. It details how a command and control server is accessed through an app created by Pawn Storm, a cyber espionage campaign. The IoCs or Attributes include a textual description written by a security expert and related url address. Further, the tag "threat-actor=Sofacy" indicates this campaign has ties to the Russian government. This event was manually labeled by a security expert as exploitative. Only significant fields are shown for clarity.

```
{
    "id": "955",
    "date": "2015-02-04",
    "info": "OSINT - Pawn Storm Update: iOS Espionage App Found",
    "attribute_count": "10",
    "timestamp": "1539794378",
    "publish_timestamp": "1557925412",
    "Attribute": [
        {
            "id": "93978",
            "type": "text",
            "category": "External analysis",
            "event_id": "955",
            "timestamp": "1539794378",
            "comment": "",
            "value": "In our continued research on Operation
            ↪  Pawn Storm, we found one interesting poisoned
            ↪  pawn\u2014spyware specifically designed for
            ↪  espionage on iOS devices...As of this
            ↪  publishing, the C&C server contacted by the iOS
            ↪  malware is live."
        },
        {
            "id": "93981",
            "type": "url",
            "category": "Network activity",
            "event_id": "955",
            "timestamp": "1539794378",
            "comment": "",
            "value": "/adhoc/XAgent.plist",
        }
    ],
    "Tag": [
        {
            "id": "24",
            "name": "misp-galaxy:threat-actor=Sofacy"
        }
    ]
}
```

2 TI FEED CHARACTERIZATION

In this section, we analyze TI feeds in our dataset with respect to IoC categories as well as their temporal and spatial characteristics. Our findings indicate that automated methods are needed to efficiently interpret loosely-structured threat data for various downstream security tasks.

Preliminaries. A *TI feed* is a general term used to describe a curated source of information such as IP addresses, file hashes, textual threat description, and other features providing context to existing and potential threats. All of these features are collectively referred to as *Indicators of Compromise* (or IoCs). Each TI feed, from a public or private provider,[3] contains a list of *events* and each event includes a collection of IoCs. Listing 1 is an event from our dataset

[3]In general, it is not always the case that there is a one-to-one correspondence between TI feeds and providers, as some TI feeds may contain events from multiple providers. In this work, however, as we anonymize providers (also known as organizations) we

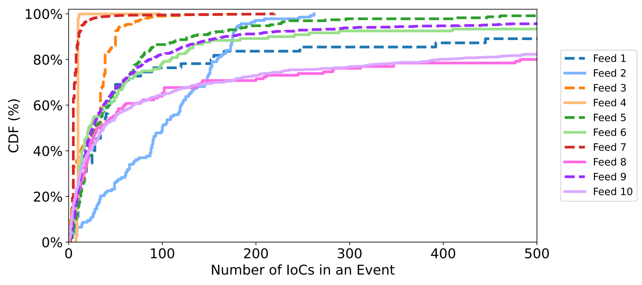


Figure 1: CDFs of the number of IoCs in an event for the top 10 feeds.

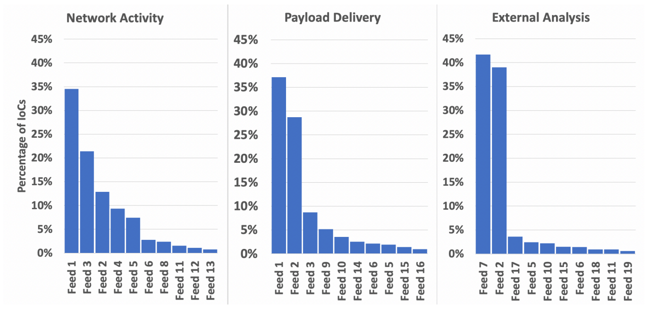


Figure 2: Contribution of TI feeds to popular IoC categories.

and the "Attributes" field includes its IoCs. An event could be a security incident such as a phishing attack, a botnet attack, or a security advisory regarding a software vulnerability. Each event also includes timestamps indicating when the event was created and last modified. The IoCs within an event are also individually timestamped based on when they are created or last modified. The *event timespan* is the difference in days between the latest and earliest IoC timestamps associated to that event. Each IoC is accompanied by a comment field which contains free-form textual description of the IoC as well as an IoC category. An event may also contain a comment field typically summarizing the security analyst's assessment of the event. The list of IoCs, comments, and timestamps together constitute the *attributes* of the event. In addition to these attributes, an event may be assigned *tags* by a security analyst. Tags follow standardized taxonomies. An event tagged as *tlp:white*, for example, contains information that can be shared with the general public according to the Traffic Light Protocol (tlp) [12], whereas an event tagged as *type:osint* contains information that is considered Open Source Intelligence (OSINT).

Summary Statistics. Our dataset is comprised of a collection of 191 TI feeds that come from a variety of public and private providers. In total, our dataset contains about 14 thousand events and 866,541 IoCs over the duration of 7 years (2013-2019). Table 1 lists key statistics of top-10 TI feeds ranked based on the number of IoCs.[4] The largest contributor is Feed 1 (a government organization) which contributes 276,481 IoC references with an average of 429 IoCs per event. We note that the number of IoCs per event vary by several orders of magnitude across different feeds. Some feeds such as Feed 3 have more than a thousand IoCs per event on average while others such as Feed 4 have less than 10. Figure 1 further plots the distribution of the number of IoCs per event across different feeds. We note that even for feeds with a high average number of IoCs per event, the majority of events contain only a handful of IoCs. For example, each of the top 10 feeds have at least 70% of their events having no more than 200 IoCs each. Further, all of the top 10 feeds, except Feed 9, have at least 60% of their events having no more than 100 IoCs each.

refer to feeds and providers interchangeably, with each provider corresponding to its own feed.

[4]Table 8 in Appendix A provides a general description of TI feed providers.

2.1 IoC Categories

Different TI feeds are geared towards different types of IoCs. Specifically, each IoC is classified by the feed vendor into one of several IoC categories.[5] Table 1 shows the most dominant IoC categories for top-10 TI feeds in our data. We note that a majority of popular TI feeds contain mostly network activity related IoCs while others contain IoCs related to payload delivery and external analysis. Network activity IoCs generally consist of IP addresses, hostnames, domains, and URLs. Payload delivery IoCs generally consist of filenames, hashes (e.g., sha1, sha256 and md5), and malware samples. External analysis IoCs generally consist of textual analysis by security experts and links to other pertinent organizations.

Next, we analyze the relative contribution of different TI feeds across popular IoC categories. Figure 2 plots the percentage of IoCs contributed by different TI feeds across popular IoC categories. We limit our analysis to top-3 IoC categories that comprise 87% of the IoCs in our data. Less popular IoC categories include artifacts dropped, social networks, support tools, etc. The largest IoC category is network activity, which accounts for 557,532 IoCs. Across different TI feeds, Feed 1 and Feed 3 together contribute more than 50% of network activity IoCs. The second largest IoC category is payload delivery, which accounts for 220,321 IoCs. Across different TI feeds, Feed 1 and Feed 2 each contribute about 30% of payload delivery IoCs. The third largest IoC category is external analysis, which accounts for 38,531 IoCs. Across different TI feeds, Feed 7 and Feed 2 each contribute almost 40% of external analysis IoCs.

Next, we analyze IoC categories with external comments. Table 2 provides insight into the prevalence and length of IoC comments in top-10 feeds.

We find that all except one feed have a majority of events with at least one IoC with a comment. Across these feeds, on average, 67.7% of IoCs have a comment and the comment length is 27.7 characters. We identified three categories of comments: additional description of an attack, format of the IoC, or logistical upload information. The first type of comments usually contain a note/advice from a security expert. For example, an IoC within one of our feeds had an additional information comment: "Kill switch domain. Monitor, do not block. (Source: Cisco Umbrella)." Such comments could be useful in determining how to proceed when responding to a security incident. The second type of comments describe the format of IoCs. For example, an IoC within one of the considered feeds contained

[5]IoC categories are self-reported by each TI feed vendor. We do not expect different TI feeds to use exactly the same categorization procedure.

Table 1: Statistics of top-10 TI feeds ranked based on IoC count

Provider	Event count	IoC count	Average number of IoCs per event	Average event timespan (days)	Dominant IoC category (%)	Dominant event tag (%)
Feed 1	644	276,481	429.3	1.77	network activity (69.65%)	misp-galaxy:tool="emotet" (11.89%)
Feed 2	1,234	165,292	133.9	45.86	network activity (43.35%)	type:osint (11.30%)
Feed 3	130	139,852	1,075.8	21.29	network activity (84.43%)	osint:source-type="block-or-filter-list" (11.96%)
Feed 4	6,955	52,803	7.6	0.72	network activity (98.51%)	circl:incident-classification="malware" (51.04%)
Feed 5	120	48,952	407.9	90.62	network activity (84.73%)	type:osint (28.01%)
Feed 6	230	27,376	119.0	14.12	network activity (56.62%)	osint:source-type="blog-post" (18.41%)
Feed 7	1,623	16,056	9.9	0.09	external analysis (100%)	njrat (30.04%)
Feed 8	613	15,953	26.0	0.00	network activity (82.94%)	malware:emotet (33.30%)
Feed 9	138	14,028	101.7	99.85	payload delivery (81.10%)	misp-galaxy:ransomware="locky" (29.91%)
Feed 10	55	11,448	208.1	49.62	payload delivery (68.52%)	malware_classification:malware-category="trojan" (13.90%)
Other feeds	2256	98,305	6.6	4.32	network activity (45.98%)	unstructured (6.22%)

Table 2: Analysis of comment fields in top-10 TI feeds.

Provider	Events with an IoC comment (%)	IoCs with comment (%)	Average comment length (characters)
Feed 1	67.9%	63.5%	56.2
Feed 2	85.7%	71.1%	49.3
Feed 3	90.8%	96.8%	12.9
Feed 4	0.0%	0%	0.00
Feed 5	63.3%	13.7%	42.3
Feed 6	97.4%	87.8%	19.8
Feed 7	100%	69.7%	9.40
Feed 8	100%	100%	10.7
Feed 9	99.3%	95.8%	46.0
Feed 10	63.6%	72.1%	30.5

the comment "RAT name," thus providing useful information that the IoC mentions the name of a specific remote access trojan. The third type of comments contain other logistical information that also provide context for the IoC. For example, an IoC in one of the considered feeds contained the comment "This attribute has been automatically imported." This information could be necessary when prioritizing which events or IoCs need to be analyzed first. Thus, we conclude that the information in these free-form textual comments provide deeper insights into security incidents.

2.2 Temporal analysis

Since an event is essentially a timeseries of IoCs, we start our temporal analysis by breaking down event timespan across different TI feeds. Recall that the *event timespan* refers to the interval between the lowest and the highest timestamp of the IoCs in the event. As shown in Table 1, average event timespan significantly varies across different TI feeds. Some TI feeds such as Feed 1 have shorter average event timespans of less than a couple of days while others such as Feed 5 have longer average event timespans of up to 3 months. Note that several TI feeds such as Feed 4 and Feed 7 have the average event timespan of less than one day. In fact, the average timespan

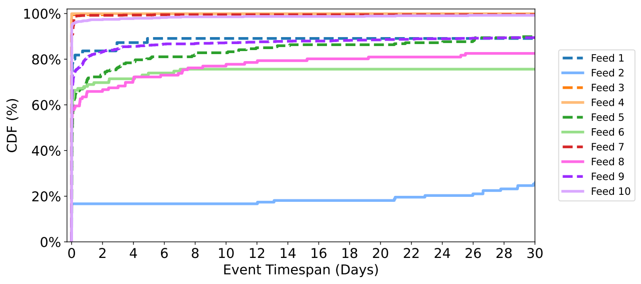


Figure 3: CDFs of the event timespan for the top 10 feeds.

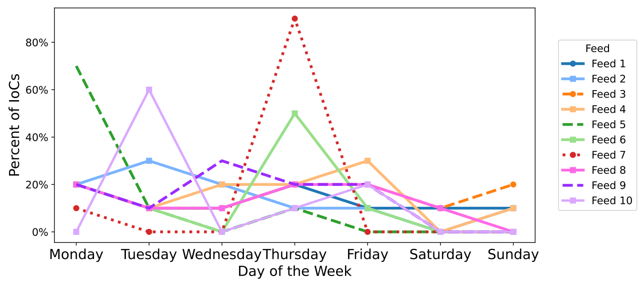


Figure 4: Temporal analysis of IoC count across the top 10 feeds (by day of week)

of Feed 8 is zero, which means that all IoCs in each of its events have the same timestamp. Figure 3 further plots the distribution of event timespan across different feeds. We note that 9 of the top 10 feeds have at least 70% of their events having timespans of no more than 4 days. Feed 9 is distinct in that its events tend to have much longer timespans, with under 30% of its events having timespans of fewer than 30 days.

Next, we leverage IoC timestamps to analyze temporal variations in IoCs across different days of the week. Figure 4 plots the distribution of IoCs across the top-10 feeds by day of the week. IoC volume spikes exist for days of the week, with many feeds containing IoCs

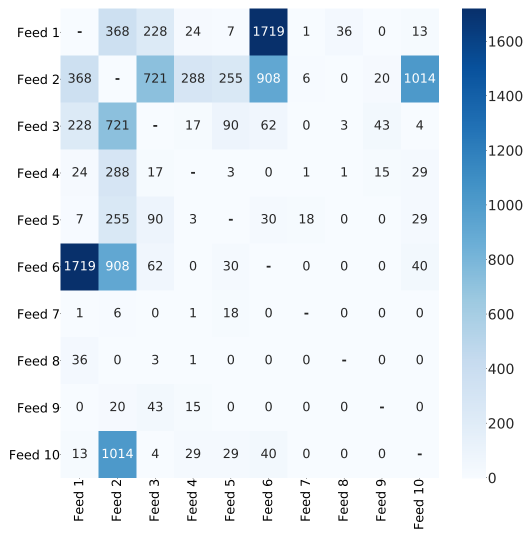


Figure 5: IoC overlap across top-10 TI feeds. Note that some TI feeds, in particular Feed 2, share common IoCs with several other TI feeds, noting that overlap only accounts for hashes and IPs.

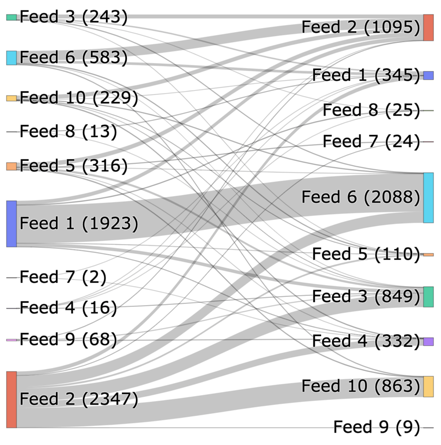


Figure 6: Flow of common IoCs across top-10 TI feeds. The left side shows counts of IoC sources and the right side shows the counts of IoC sinks.

disproportionately timestamped on certain days of the week. For example, Feed 3 had a significant portion of its IoCs timestamped for Thursday and Friday. Similarly, Feed 5 had approximately 73% of its IoCs created/updated on Monday and Feed 6 had approximately 52% of its IoCs timestamped for Thursday. These IoC frequency spikes across different days of the week show that there may be periodic disclosures/authoring/updates of IoCs and these schedules may vary across different TI feed providers.

2.3 Spatial Analysis

Next, we analyze the IoC overlap to characterize information redundancy across TI feeds. Note that we restrict ourselves to IoCs containing specific IP addresses and hashes. Figure 5 shows that some feeds share only a few IoCs with other feeds, while others share a lot of IoCs with several other feeds. Light streaks in the plot indicate that a particular feed shares very few IoCs with other feeds. For example, Feed 8 shares less than 4 IoCs with all but one of the top-10 TI feeds. Dark streaks show that many TI feeds tend to have a significant number of their IoCs in common a particular feed. For instance, Feed 2 shares over 250 IoCs with 6 of the other top feeds, with over 900 IoCs shared with Feed 6 and over 1000 IoCs shared with Feed 10. Similarly, Feed 1 shares upwards of 200 IoCs with Feed 2 and Feed 3 and over 1700 with Feed 6. Such overlap between different TI feeds is not surprising and has been reported in prior work on IP blocklists [17, 25].

We next analyze the flow (or temporal ordering) of common IoCs across TI feeds. Specifically, in contrast to the IoC overlap analysis above, we take into account which feed publishes a common IoC before the other. Figure 6 plots a Sankey diagram to visualize the flow of IoCs across the top-10 TI feeds in our dataset. First, we note that some TI feeds are mainly the source of common IoCs. For example, Feed 1 is the source of approximately 84.8% of its common IoCs. Second, some TI feeds are mainly the sink of common IoCs. For example, Feed 6 is mainly the sink of IoCs from Feed 1 and Feed 2. This shows that Feed 6 basically aggregates a subset of IoCs from multiple TI feeds. Third, other feeds are more symmetric in being the sources/sinks of common IoCs. For example, Feed 2 is the source of 68% of its common IoCs with other feeds. Thus, we conclude that TI feeds may have source/sink relationships for common IoCs with other feeds.

Takeaways. Our characterization highlighted three main insights. First, TI feeds contain various categories of information about different threats in a loosely-structured format. Second, the information in TI feeds and specific events is not static—it is dynamically updated and spread over often long time intervals. Third, TI feeds complement each other by providing differing amounts and types of IoCs but do sometimes contain redundant information. As we discuss next, there is a need to develop automated methods to analyze loosely-structured, spatio-temporal threat data in different TI feeds for downstream risk assessment, incident response, and patch management tasks.

3 TI FEED CLASSIFICATION

The increasing volume of loosely-structured information in TI feeds renders them unwieldy for manual sifting, calling for methods to extract actionable information in an automated manner. In this section, we present a machine learning pipeline to automatically analyze and classify TI feed events. Our proposed machine learning pipeline first leverages state-of-the-art natural language processing (NLP) techniques to extract semantic information from TI feeds and then trains supervised machine learning models for classification.

3.1 Problem statement

We focus on the problem of detecting exploitation of security vulnerabilities from TI feeds, classifying TI feed events for that matter.[6]

[6]Note that this problem is related but different compared to vulnerability weaponization (i.e., whether an exploit for a particular vulnerability exists) [4, 13, 27]. In particular, weaponization can be determined in a controlled environment, whereas exploitation requires measurements in the wild.

This is a useful undertaking because exploitation information is crucial for threat response and vulnerability management. The most concrete application is the recently proposed Exploit Prediction Scoring System (EPSS) [11], which specifically aims to predict the likelihood of vulnerability exploitation after disclosure. EPSS scores (i.e., likelihood that a vulnerability will be exploited within the 12 months following its public disclosure) are calculated based on a regression model that takes into account features such as the vendor's name and whether a proof-of-concept exploit has been developed [15]. Measurement and detection of exploitation events in the wild essentially provides the ground truth for EPSS scores. Indeed, after an event is classified as an exploitation, the set of vulnerabilities in the attack surface of the environment where the exploitation occurred are natural candidates to have their CVSS/EPSS scores updated. Thus, we aim to build models to automatically classify TI feed events into two classes: *exploitation* and *non-exploitation*.

3.2 Ground truth

We need labeled ground truth to train and test our machine learning classification pipeline. Thus, we discuss our approach to bootstrap ground truth using a small set of TI feed events that are painstakingly labeled by experts manually.

3.2.1 Manual labeling by experts. We begin by filtering TI feed events that mention specific vulnerabilities (i.e., CVE–ID) in their textual descriptions, comment, or IoC attributes. The filtered TI feed events are then manually labeled by two independent experts, with 10% concurrently labeled events achieving the inter-rater agreement score of 89%. The experts used the following code book to manually label exploitation events. Events that are labeled exploitation: (1) have a definite malicious components such as APTs, DDoS, and trojan attacks; (2) are part of a well-known malicious campaign; (3) mention a well-known hacker group/organization; or (4) contain keywords such as espionage or campaign (given suitable surrounding context). Listing 1 was labeled as an exploitation event since it fits criteria (2), (3), and (4). In contrast, events that are labeled non-exploitation: (1) are updates to existing software without a malicious component for that update; (2) contain keywords such as patch, update, reset (given suitable surrounding context); (3) have no clear-cut indication of malicious action; or (4) contain insufficient data or ambiguous terminology to be classified as exploitation. This manual procedure was used to label four thousand events.

3.2.2 Event tags. Each event in TI feeds may contain one or more tags.[7] Composed of a namespace, predicate, and optional value, a tag of the form namespace:predicate=value can be utilized to further classify an event. "misp-galaxy:threat-actor=Sofacy" and "tlp:white" are tags in the exploitative event Listing 1. These tags are part of the MISP taxonomies [22], which allow various organizations to adopt standard classification vocabulary. The tags are manually assigned to each event and are designed for automated consumption by downstream tasks (e.g., intrusion detection systems). There are drawbacks to this approach as bias from a security expert could sway which tags are being added and tagging each event is time intensive. On the other hand, it can be easier to control quality with manual tagging. Since every event in our dataset is annotated with at least one tag, tags could be utilized as a baseline for classification analysis.

[7]Tags are referred to as 'machine tags' in MISP.

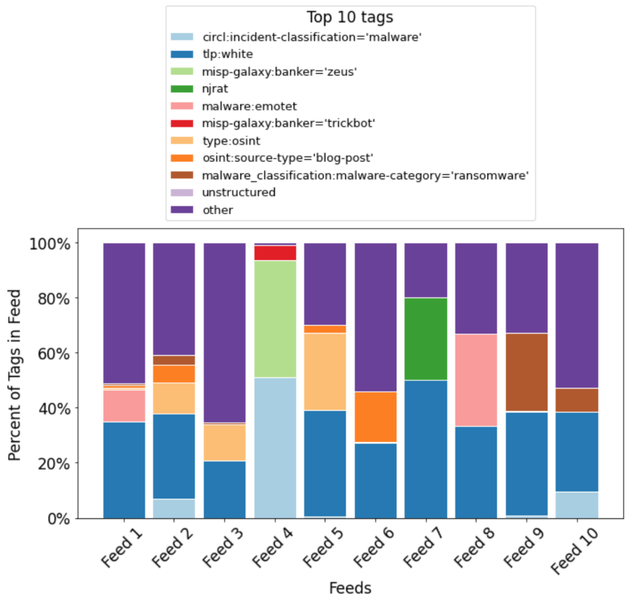


Figure 7: Breakdown of most common tags throughout top 10 feeds

We analyze the distribution of tags in the top-10 feeds in our dataset. Figure 7 plots the percentage of each tag type in terms of all tags found per feed. In most feeds, the events typically contain 2-3 tags. Common tags across feeds include circl:incident-classification="malware", malware:emotet, type:osint, and malware_classification:malware-category= "ransomware". The specific nature of each tag helps to verify the focus of an event. Since tags are manually added and hold pertinent information about an event, they may be useful in determining exploitation incidents.

3.2.3 Automated labeling based on event tags. To expand ground truth beyond the four thousand manually labeled events, the rest of the labeling process was automated using these event tags. Specifically, we extracted association rules between event tags and exploitation on manually labeled events. We used the apriori algorithm with support ≥ 0.001, confidence ≥ 0.5, and lift ≥ 1 to shortlist approximately 735 rules. Note that we filtered some unrelated tags (e.g., `tlp`) before extracting association rules. Once the rules which mapped combinations of tags to exploitation were generated, these rules were used to determine the ground truth for the remaining ten thousand unlabeled events. More specifically, an event was labeled as an *exploitation* if and only if all of the tags in at least one of the shortlisted rules matched the tags of the event. If not, the event was marked as *non-exploitation*. For example, the apriori algorithm determined a rule `sectorfinancial` & `kill-chain:delivery` & `kill-chain:command and control` → **Exploitation**, meaning that if each of "sectorfinancial", "kill-chain:delivery", and "kill-chain:command and control" tags are present then it would be labeled as an exploitation event. Additional shortlisted rules are provided in Appendix B.

3.2.4 Label distribution. Table 3 lists the number of events in our dataset that were labeled as exploitation or non-exploitation

Table 3: Results from ground truth labeling for top-10 TI feeds.

Feed ID	1	2	3	4	5	6	7	8	9	10	Total
Exploitation	108	766	116	0	92	182	0	0	5	48	1317
Non-exploitation	536	468	14	6955	28	48	1623	613	133	7	10425

across the top-10 feeds. We note that 11% of all events in the top 10 feeds were labeled as exploitation and that the distribution of threat events varies across different feeds. For example, Feed 3 had approximately 89% of its events labeled as exploitation and Feed 10 had approximately 87% of its events labeled as exploitation. This is in stark contrast to the ground truth labeling results of feeds such as Feed 4, Feed 7, and Feed 8, all of which had no exploitation events. Across all TI feeds in our dataset, 82% of the events were labeled as non-exploitation and the remaining 18% were labeled as exploitation.

3.3 Longitudinal evaluation

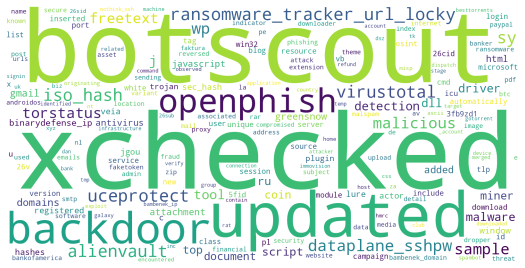
A key challenge in detecting incidents such as vulnerability exploitation is their constantly evolving arms race nature. Thus, it is important to evaluate the accuracy of detection models in a longitudinal manner because detection models need to evolve over time in response to the dynamic threats. To this end, we use two methods to split the dataset for training and testing: *temporal* and *temporal+spatial*.

3.3.1 *Temporal splits.*

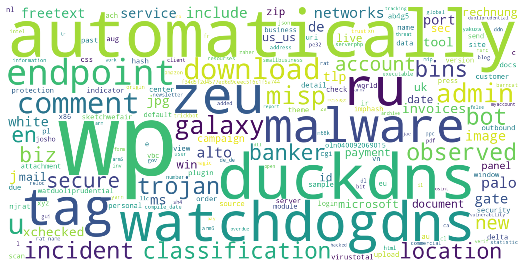
To evaluate the accuracy of our classification models in a longitudinal setup, we split our entire dataset into distinct time windows such that the earlier windows are used for training and the later windows are used for testing. Thus, we make sure that the events in the train set occur prior to those in the test set. We consider two different approaches to longitudinally split our dataset: aggregate window and equal window. Both aggregate window and equal window splits divide data into 11 time windows such that each window contains an equal number of events. In aggregate window split, the model uses the events in the next time window for testing and the events in all previous time windows for training. Thus, the amount of training data increases over time as more time windows are accumulated. In equal window split, the model uses the events in the next time window for testing and the events in the immediately preceding time window for training. Thus, the amount of training data remains the same over time in equal window splits. Such temporal splits would help us evaluate whether the accuracy of our machine learning pipeline is sensitive to the amount and recency bias of training data.

3.3.2 *Temporal+spatial splits.*

We further build upon the aforementioned temporal splits by additionally splitting events across different TI feeds. Specifically, we consider the scenario where a feed is not available during the training but its events need to be classified at the test time. Thus, the test set is restricted to contain events only from one feed and the associated train set contains no events from that feed while respecting the aforementioned temporal splits. For example, consider Feed 2. If we split spatially, the train set of each window will not contain any events from Feed 2,



(a) exploitation events



(b) non-exploitation events

Figure 8: Word clouds suggest the feasibility of natural language processing to encode events from TI feeds into embeddings, which may be used to classify them, e.g., between exploitation and non-exploitation.

while the test set will only contain events from Feed 2. Such temporal+spatial splits would help us evaluate whether the accuracy of our machine learning pipeline generalizes to other previously unseen feeds.

3.4 Features

To provide insights into the kind of features we can extract from TI feeds, we first visualize various event attributes for exploitation and non-exploitation events in Figures 8(a) and 8(b). Exploitation events are usually followed by updates, which are typically made explicit in the event description, hence the term 'updated' being often present. Listing 1, an exploitative event with "update" in its info field, describes a detected espionage app and that the command-and-control server it was in contact with is still live. Similarly, the strategy to detect such exploitation is also usually reported, e.g., indicating that the event was detected using 'botscout' or 'uceprotect'. Non-exploitation events, in contrast, refer to security advisories, or report weaponization or vulnerabilities that do not necessarily have had an impact in the wild. Vulnerabilities associated to Wordpress, for instance, are typically discussed in some of the non-exploitation events, hence the word 'wp' appearing in this wordcloud. As another example, both Microsoft Windows and Linux keep 'watchdog' timers that are frequently restarted to indicate that the system is operating normally. Comments related to when and if watchdogs timeout in face of a given malware family can be derived from sandboxes and controlled experiments, in this case characterizing non-exploitation events. It is noteworthy that the presence or absence of a word is not sufficient to determine the class of an event, as context plays a key role in the classification process.

Next, we explain our feature extraction pipeline.

3.4.1 Pre-processing. Before features are extracted, we conduct standard pre-processing, including splitting the text on all special characters, such as hyphens, and removal of stop words. Some IoCs, such as hashes, might refer to a malware payload, and receive special treatment. As threats have polymorphic capabilities to mask file signatures, most hashes are unique to one feed. This, in turn, implies that information about hashes cannot be readily generalizable across feeds. To address this issue, we replace all instances of hashes with an abstract representation noting the existence of a hash value.

3.4.2 Feature extraction. We consider two different embedding-based feature extraction techniques. First, we use a context-independent static embedding technique, called Doc2Vec [16]. Doc2Vec maps all words in an event to a feature vector such that it captures the relationship between a word given its neighboring words. Second, we use a content-dependent dynamic embedding technique, called Bidirectional Encoder Representations from Transformers, BERT [7]. BERT maps all words in an event to a feature vector using an attention-based model that learns contextual relations between words in a text. We use BERT base uncased in [41] to encode our input data into fixed-length vector representations. As compared to Doc2Vec, which generates the same embedding for the same word in different contexts, BERT is better able to capture the semantics by modeling the surrounding context of an input word. For both Doc2Vec and BERT, we not only consider pre-trained models but also train purpose-built models that are further trained on data from TI feeds. For Doc2Vec, we train TI2Vec in an unsupervised manner using the corpus of 191 TI Feeds in our dataset. Training is done over 15 epochs. This customized embedding aims to capture the context of domain specific vocabulary that is typically found in TI feeds. With BERT for sequence classification, we train TIBERT in a supervised manner from the standard bert-base-uncased model with our TI dataset. We utilize a classification layer to finetune it over 10 epoches.

3.4.3 Feature selection. We perform feature selection to avoid over-fitting. For Doc2Vec, we select n to be 1000. For BERT, input is transformed into 768-dimensional vectors.

3.4.4 Supervised classification. Given the features and labeled dataset, we train different supervised machine learning algorithms to classify TI feed events. We begin by considering Naive Bayes and Decision Tree classifiers, as both serve to classify non-linearly separable classes, with Naive Bayes relying on conditional independence among features for that purpose. Following the discussion in Section 2, there are significant inter-dependencies across features within each class, suggesting that Decision Trees are better suited than Naive Bayes. As a natural extension to Decision Trees, we also consider the AdaBoost ensemble classifier. Boosting in that context consists of using many low-fidelity Decision Trees, which together are supposed to outperform a single tree. While the Decision Tree is amenable to interpretability, ensemble classifiers tend to provide better classification performance. Note that TIBERT has a built-in head that classifies the transformer's vector representation of the input based on what was learned during training.

Table 4: Temporal classification accuracy (F1 scores) results of different features and classifiers using equal window temporal splits.

Split ID	Approach									
	Doc2Vec			TI2Vec			BERT Embeddings			TIBERT
	Naive Baynes	Decision Tree	Ada Boost	Naive Baynes	Decision Tree	AdaBoost	Naive Baynes	Decision Tree	Ada Boost	
2	0.93	0.64	0.86	0.9	0.75	0.83	0.68	0.63	0.75	0.72
3	0.87	0.81	0.72	0.82	0.87	0.83	0.8	0.75	0.78	0.92
4	0.69	0.50	0.55	0.37	0.68	0.74	0.36	0.44	0.33	0.83
5	0.06	0.55	0.75	0.5	0.81	0.83	0.79	0.75	0.83	0.89
6	0.57	0.47	0.64	0.6	0.63	0.65	0.42	0.49	0.57	0.69
7	0.49	0.46	0.67	0.59	0.6	0.67	0.67	0.58	0.63	0.66
8	0.60	0.60	0.74	0.49	0.7	0.75	0.58	0.67	0.75	0.78
9	0.32	0.40	0.64	0.53	0.61	0.65	0.57	0.63	0.61	0.65
10	0.51	0.76	0.74	0.46	0.49	0.68	0.54	0.54	0.74	0.84
Average	0.56	0.58	0.70	0.58	0.68	0.74	0.60	0.61	0.67	0.78

4 EXPERIMENTAL EVALUATION

4.1 Evaluation metrics

To evaluate the accuracy of our classifiers, we use three well-known metrics: precision, recall, and F1 score. Precision is the fraction of presumed exploitation events that are correctly detected. Recall is the fraction of exploitation events that are detected. The F1 score combines precision and recall through their harmonic mean. Our dataset has imbalanced class distribution (82% non-exploitation; 18% exploitation); thus, we focus on F1 score here. Due to space constraints, we do not include precision and recall results that show similar overall trends.

4.2 Results

4.2.1 Impact of feature extraction. Table 4 summarizes the classification results for different feature extraction approaches using the equal window split. We break down the results for 4 different feature extraction approaches: Doc2Vec, TI2Vec, BERT, and TIBERT. The best average classification accuracy of 78% is achieved by TIBERT. TIBERT turns out to be a better approach than the pre-trained BERT embeddings as well as Doc2Vec and TI2Vec embeddings. TIBERT achieves 4% better F1 score than TI2Vec.As compared to TI2Vec, TIBERT further produces word representations that are context-dependent. Thus, the same word would have multiple embeddings differing by the context of the words around them whereas TI2Vec is a context-independent embedding. Thus, we expect the context-dependent TIBERT to be advantageous to the TI2Vec embeddings. Overall, TI2Vec and TIBERT significantly outperform their vanilla pre-trained Doc2Vec and BERT counterparts. The improvement in accuracy can be attributed to the specialization of these embeddings on a large corpus of IoC data from many different TI feeds. By doing so, TI2Vec and TIBERT capture the context of security events in TI feeds, being able to better characterize its similarity to other security events when compared to vanilla pre-trained embeddings.

Table 5: **TiBERT temporal results considering both types of splits. Each value is the F1 score.**

Split Type	Split ID									Average
	2	3	4	5	6	7	8	9	10	
Equal Window	0.72	0.92	0.83	0.89	0.69	0.66	0.78	0.65	0.84	0.78
Aggregate Window	NaN	0.86	0.37	0.17	0.56	0.77	0.77	0.67	0.84	0.63

Table 6: TiBERT Spatial Results. Each value is the F1 score when using TiBERT and temporal+spatial aggregate window Split 7. Note that, we do not consider the omission of Feed 9 results in the analysis as it only contains 5 exploitation events.

Split Type	Feed Omitted					
	Feed 1	Feed 2	Feed 3	Feed 5	Feed 6	Feed 10
Aggregate Window	0.45	0.79	0.94	0.92	0.89	0.60

4.2.2 *Impact of classifier.* Comparing different machine learning classifiers[8], we note that AdaBoost outperforms Naive Bayes and Decision Tree. Different classifiers appear to handle the precision-recall trade-off differently. For instance, not shown in Table 4, the Decision Tree classifier is inclined to optimize for precision while the Naive Bayes classifier is inclined to optimize for recall. AdaBoost achieves the best balance between precision and recall. AdaBoost outperforms Decision Tree by 12% for Doc2Vec, 6% for TI2Vec, and 6% for BERT in terms of overall F1 score.

4.2.3 *Impact of temporal splits.* Table 5 reports F1 score for TIBERT for two types of temporal splits: equal and aggregate window. As compared to equal window splits, aggregate window splits accumulates training data over time. However, we note that the classification accuracy for aggregate window splits does not monotonically increase across splits – it increases initially and then decreases sharply for split 4 and then improves again eventually as the size of the train set increases. The equal window split type explores the impact of equal-sized train and test sets. Note that the test sets are comparable between both split types, but aggregate window uses more training data. It is interesting to note that TIBERT achieves higher F1 scores overall for equal window splits. For split 4, the sharp drop in F1 scores is explained by the large difference in the vocabulary between train and test sets.[9]

Comparing aggregate window and equal window, we note that equal window achieves better classification performance. For each split, limiting our training data to only recent events instead of aggregating it generally resulted in a higher F1 score. For instance, the equal window split average (78%) outperformed aggregate window split average (63%) for TIBERT. Thus, we conclude that the limiting the training data to the most recent ultimately helps the classifier to more accurately detect exploitation events than using more training data.

4.2.4 *Impact of TI feed.* Next, we report the accuracy results for joint temporal+spatial splits. These joint splits allow us evaluate the transferability of the classification model to an unseen TI feed. Table 6 shows the F1 scores for the various temporal+spatial splits, grouped by the feed omitted from the train set (which is the only feed present in the test set). Since valid F1 scores depend on the presence of exploit events in the test set, we further restrict analysis to temporal+spatial splits for feeds that contain at least 1 exploit event in the test set. We focus our analysis on the TIBERT aggregate window approach as it yielded the highest F1 scores. We observe a strong correlation between the findings of Figure 6 and Table 6. We note that the feeds with IoCs that are mostly "sinks" also have higher average F1 scores when that feed is omitted in the temporal+spatial splits. For example, Table 6 shows that the omission of Feed 6 yields a high F1 score (89%). Feed 6 also notably has the most number of Hash and IP IoCs (2088) that are sinks of IoCs from other feeds in the dataset. Likewise, the omission of Feed 2, which has the second highest number of sink Hash and IP IoCs (1095), yields F1 scores that are in the top 4 highest scores for aggregate window split.

In the same vein, the omission of Feed 1 yields a relatively low F1 score (45%). While Figure 5 shows that Feed 1 has high IoC overlap with other feeds, Figure 6 shows that the majority ($\approx 85\%$) of Feed 1's IoC overlap occurs when it is a source and not a sink. Exceptions to these general findings do exist, however. For example, as Figure 6 shows, Feed 5 is more often a source than a sink, yet its omission yields a high F1 score (92%). This may indicate that Feed 5 shares many similar IoCs with other feeds, despite not having an abundance of exact IoC overlap.

Overall, these findings indicate that our model's ability to detect exploitation events in TI feeds it has not been introduced to yet is dependent on the relationship between the IoCs of the training and testing data. High accuracy exploitation detection is possible under spatial restrictions if the training data has same/similar IoCs as those present in the testing feed.

4.2.5 *Feature ablation analysis.* We conduct feature ablation analysis to assess the relative contribution of different IoC categories. Table 7 plots the classification accuracy for top three IoC categories: network activity, payload delivery, and external analysis. We note that classification accuracy for network activity IoCs is better than both payload delivery and external analysis. The results are inline with the earlier finding that our TI feed dataset contains the most network activity IoCs (557,532), followed by payload delivery (220,321), and then external analysis (38,531). Overall, all IoCs of different categories instead of just one, such as network activity, yields the best classification accuracy (78% vs 66%). Thus, we conclude that while some IoC categories are more helpful than others, they complement each other and using them together outperforms standalone classification performance.

4.2.6 *Applications of TIBERT.* The exploitation events predicted by TIBERT can be utilized to update the temporal exploitability

Table 7: TIBERT feature ablation results. Each value is the F1 score.

Feature Ablation Type			All Categories
Network Activity	Payload Delivery	External Analysis	
0.40	0.39	0.66	0.78

[8]except for TIBERT which has a built-in classifier

[9]For split 4, the difference in vocabulary between the test and train sets was about 44,000 words. The difference was determined by computing the number of words in the test set that did not appear in the train set.

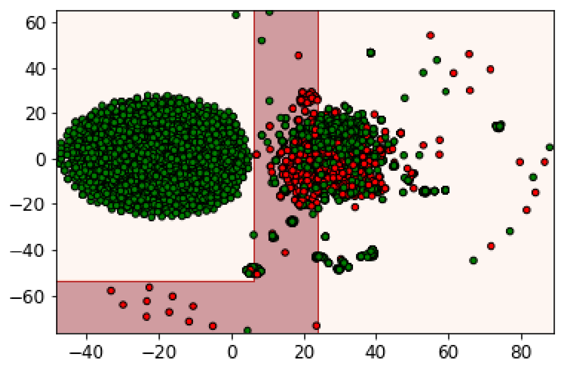
score of CVSS/EPSS related to the availability of PoC or fully functional exploits. We focus specifically on temporal metrics since TIBERT generates predictions based on the past history of a event. The *Exploit Code Maturity* metric describes the maturity level of an exploit. The output of TIBERT can be used to automatically revise this metric, such as updating it from "Proof of concept code" to "Functional exploit exists". By temporally analyzing large volumes of loosely structured TI feed data, TIBERT provides a way to automatically improve CVSS/EPSS score quality, saving time and effort of security experts.

As the accuracy score of TIBERT is 0.78 on average, we acknowledge that false positives and false negatives can impact security operations. A false positive exploitation can untimely escalate patching, causing transient unplanned system unavailability and could increase costs. This leads to the misprioritization of vulnerabilities. On the other hand, a false negative exploitation could cause security experts to not identify the exploit altogether, leaving the software susceptible to attacks. However, since the goal of TIBERT is to update CVSS scores not replace them, these risks are present, but to a lesser degree. Note that TIBERT's false positive rate (1%) is configured to be half that of its false negative rate (2%). It is also noteworthy that since there are 2 other categories and 19 additional metrics, inaccurate Exploit Code Maturity and Report Confidence metrics within the temporal category does not yield a significant impact on the overall EPSS/CVSS score of the vulnerability. Thus, we argue that TIBERT is a useful approach to automatically updating EPSS/CVSS scores with minimal risks on security operations despite its less than perfect accuracy.

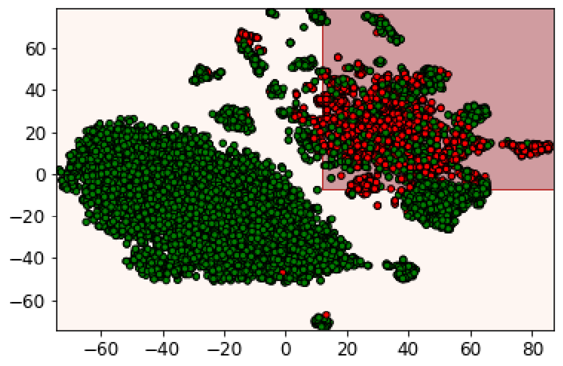
4.3 Visualization and Interpretability of Doc2Vec and BERT

4.3.1 Visualizing Doc2Vec and TI2Vec embeddings. Figure 9 illustrates Doc2Vec and TI2Vec embeddings. Using TSNE, we are able to visualize how the embeddings produced by Doc2Vec and TI2Vec spread over the plane. Each point corresponds to an event. Green and red points correspond to non-exploitation and exploitation events, respectively. Whereas both Doc2Vec and TI2Vec produced two clearly distinguishable clusters, the clusters produced by TI2Vec are easier to separate using a simple decision tree. A simple decision tree, with two levels, is already able to classify the events with a 55% and 66% F1-score using Doc2Vec and TI2Vec, respectively. Those numbers are in agreement with Table 4, noting that the average F1-scores of 58% and 68% reported in Table 4 for Doc2Vec and TI2Vec were obtained with deeper decision trees and the original embeddings. As expected, using shallow decision trees and the simplified embeddings perturbed by TSNE caused a slight decrease in performance. It is also worth noting that the accuracy of all considered variations remained roughly 84%.

4.3.2 Visualizing BERT and TIBERT attention. To shed light into the inner workings of BERT, we next provide additional insight on how BERT embeddings capture semantics using attention heads. To that end, we first briefly introduce minimal background on the BERT architecture. BERT is a transformer, which transforms sequences of tokens (sentences), e.g., corresponding to event descriptions, into vectors. A key element in the BERT encoding is captured through the so-called attention heads, which rely on attention weights [37].



(a) Doc2Vec



(b) TI2Vec

Figure 9: Doc2Vec and TI2Vec embeddings being input into a decision tree classifier: both embeddings allow for accurate classification, with TI2Vec being roughly 10% superior in terms of F1 score.

BERT architecture is divided into 12 layers, and each layer counts with 12 attention heads. Each attention head is a matrix, whose element at the i-th row and j-th column captures the strength of the relationship between token i and token j. Let $\alpha^{(k,l)}$ denote the k-th head of layer l. The stronger the attention weight between two tokens, the larger the impact of their presences towards the embedding produced by BERT.

Figure 10 illustrates attention weights using BertViz [38]. The darkness of a line is proportional to $\sum_{k=1}^{12} \alpha_{i,j}^{(k,l)}$ and indicates the strength of the attention weight between i and j at the l-th layer. Some lines are invisible because their corresponding attention weights are quite low. Correspondingly, the 12 heads are characterized by color bars next to each word, whose intensity is proportional to $\alpha_{i,j}^{(k,l)}$, for $k = 1, \ldots, 12$. Figures 10(a) and 10(b) show the attention head $\alpha_{i,j}^{(k,3)}$ for a sentence selected from an exploitation event in the *external analysis* category, transformed by BERT and TIBERT, respectively. Similarly, Figures 10(c) and 10(d) show $\alpha_{i,j}^{(k,1)}$ for a sentence selected from an exploitation event in the *network activity* category.

In Figures 10(a)-(b) and 10(c)-(d), the words OSINT and phishing were divided into two tokens each, following the WordPiece tokenization procedure referred to in Section 3. Figure 10(a) shows that the output of the embedding of an event containing the phrase "look into recent exploit kit activities" will rely on the fact that the tokens 'exploit', 'kit' and 'activities' appear together. Correspondingly, Figure 10(b) shows that after fine tuning the embedding, training it with TI information, the presence of the tokens comprising the term OSINT should be taken into account while producing the embedding. As a second example, Figures 10(c) and 10(d) show that with the pre-trained model the joint presence of terms 'activity' and 'network' significantly impacts the output of the embedding, while after fine tuning with TI information the presence of additional context related terms, such as 'phishing', also plays an important role on the output. Note that the term 'phishing' also appeared as relevant in the word clouds of Figure 8. In this last example, the relevance of the attention weight between 'activity' and the sentence separator (denoted by [SEP]), whose rationale is out of the scope of this work and is further discussed in [6], is also noteworthy.

Our illustrative examples indicate that after fine tuning BERT the joint presence of certain domain specific terms, such as OSINT and phishing, became relevant. Those two terms, in particular, were not considered as relevant by any of the attention heads, in any of the layers in the pre-trained BERT transformer. After fine tuning, in contrast, we observed that these and other terms typical in the TI feed domain received the deserved attention.

In summary, a pre-trained BERT model already captures the joint presence of relevant terms in a sentence to produce corresponding embeddings. By fine tuning the training of the transformer, TIBERT leverages contextual information by capturing the joint presence of additional terms, specific to the TI feed domain, to produce sentence embedding which further simplify downstream tasks such as the classification of events between exploitation and non-exploitation events.

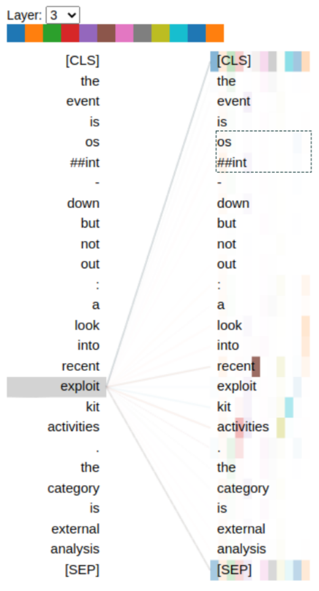
5 RELATED WORK

In this section, we report prior research pertaining TI feeds measurements and models, vulnerability lifecycles, and attack embeddings.

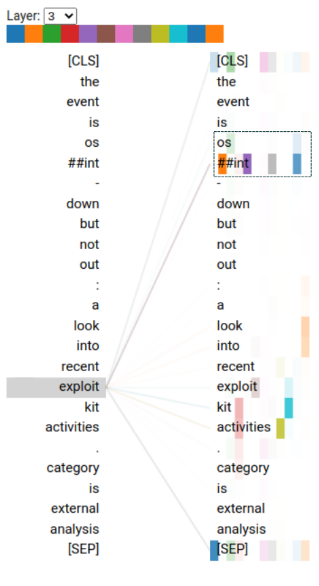
5.1 Analyzing information in TI feeds

There is growing interest in the research community to better understand the value of information in TI feeds. Early work focused on TI feeds that targeted specific threat niches [24, 25, 29].

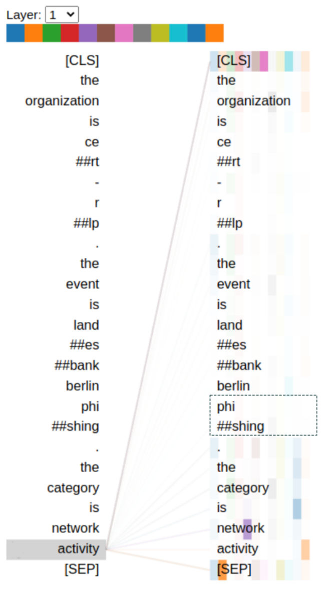
Phishing and spamming are classical themes in the realm of TI feeds [24, 29]. Sheng et al. [29] analyzed the effectiveness of 8 different phishing URL blacklists. They found that the quality of different blacklists varies with respect to coverage and information freshness. Pitsillidis et al. [24] analyzed the effectiveness of 10 different spam feeds. They discovered that some low volume feeds capture the vast majority of spam domains. They also reported biases in some feeds towards certain types of spam, suggesting their complementarity with respect to coverage. More recently, Li et al. [17] analyzed 47 different TI feeds including Facebook ThreatExchange, and aligned with the above studies also pointed that the quantity and quality of information across different TI feeds varies substantially. In conclusion, the main takeaway from


Layer: 3
[CLS] the event is os ##int - down but not out : a look into recent exploit kit activities . the category is external analysis [SEP]

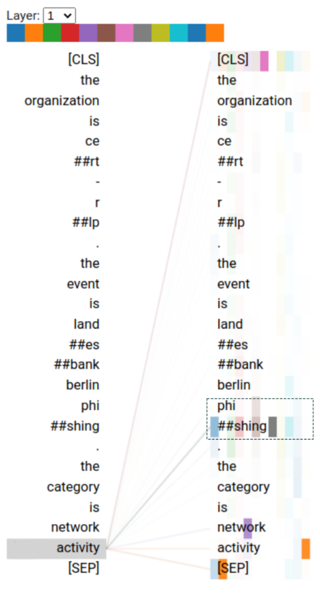
(a) BERT


Layer: 3
[CLS] the event is os ##int - down but not out : a look into recent exploit kit activities . category is external analysis [SEP]

(b) TIBERT


Layer: 1
[CLS] the organization is ce ##rt - r ##lp . the event is land ##es ##bank berlin phi ##shing . the category is network activity [SEP]

(c) BERT


Layer: 1
[CLS] the organization is ce ##rt - r ##lp . the event is land ##es ##bank berlin phi ##shing . the category is network activity [SEP]

(d) TIBERT

Figure 10: Attention heads of layers that captured important contextual relationships between tokens [38].

prior work on empirical analysis of TI feeds is that they vary widely in terms of the quantity, quality and coverage of data [35].

The combination of information from different TI feeds can produce high quality data with broad coverage [3, 9, 25, 34]. Ramanathan et al. [25] analyzed 157 different IP blacklists. They showed that aggregating different IP blacklists can improve the overall quality of information. Similar findings in the realm of the COVID-19 Cyber Threat Coalition were reported in [3]. Thomas et al. [34] designed and analyzed Babel threat exchange by combining abuse incident information from 6 different Google services. They also showed that combining information from seemingly unrelated services can improve the overall coverage.

Our work builds on prior research in this space by analyzing 191 different general purpose TI feeds. However, instead of a similar comparative analysis of different TI feeds, our main goal is to develop an automated approach to identify vulnerability exploitation events from TI feeds. As we discuss next, vulnerability exploitation is particularly useful in risk-aware patch management.

5.2 Predicting vulnerability lifecycles

Another promising line of research has focused on leveraging different sources of information for predicting various states in a vulnerability's lifecycle, such as discovery, weaponization, exploitation, reporting and patching [1, 2, 26, 39, 40, 43].

Models for vulnerability lifecycles are instrumental to parametrize EPSS/CVSS. Whereas EPSS/CVSS are vulnerability-oriented, requiring a CVE ID to assess its exploitability/risk, our work is TI event-oriented. Our work (1) aims at determining if a given event corresponds to an exploitation event, (2) is event-driven, leveraging the description of security incidents for that matter, and (3) relies on a dataset in which very few incidents explicitly refer to CVE IDs. TIBERT's detection of exploitation events in the wild essentially provides the ground truth for EPSS scores.

Sabottke et al. [26] and Suciu et al. [32] showed that social media data can be leveraged to detect vulnerability exploits in the wild. They trained a machine learning classifier using keyword and other metadata features to detect proof-of-concept and real-world exploits. Complementary to those works, Almukaynizi et al. [1, 2] showed that data from darknet web forums and other traditional sources can be combined to detect vulnerability exploitation. They trained various machine learning classifiers to predict the likelihood of vulnerability exploitation. Wang et al. [39] used the TI feed from Shodan to measure patch deployment in the wild. Xiao et al. [40] later combined information from spam TI feeds with patch deployment measurements to detect vulnerability exploitation in the wild. They showed that combining threat information and patching measurements can enable early detection of vulnerability exploits. Zhu et al. [43] trained a machine learning classifier to detect malware campaign stages (e.g., baiting, installation) using TI feeds.

Our work is similar in spirit to this line of research on training machine learning classifiers to predict different stages of a vulnerability's lifecycle. Differently from prior work, we focus on training machine learning models to specifically detect instances of vulnerability exploitation from loosely structured TI feeds.

5.3 Attack embeddings

Another related line of research focuses on building novel attack representations [8, 19, 28, 30, 33]. The challenges of this endeavour stem from threat feeds containing information that is inconsistently structured and organized in fields that may differ significantly across different organizations. Indeed, those aspects together with the fact that a significant portion of the data is made available within free-form text fields make it extremely difficult to process TI feeds automatically.

Mezzour et al. [19] analyzed two Symantec threat feeds using keyword analysis to identify attack and family names. Recent work has focused on learning purpose-built representations (or embeddings) for various downstream tasks. Tavabi et al. [33] trained a paragraph embedding to predict whether a vulnerability discussed on darknet web forums will be exploited. Shen et al. [28] trained temporal word embeddings to detect different steps in complex attacks. Fang et al. [8] used a pre-trained fastText word embedding for exploit prediction, whereas Perry et al. [23] proposed a novel embedding for attack attribution, i.e., identifying the entity responsible for an attack.

Our work builds on this line of research by evaluating keywords and embeddings to detect vulnerability exploitation from TI feeds. In addition, we conform to temporal constraints while evaluating the considered classifiers, and assess how knowledge transfer relates to the natural flow of information across TI feeds.

6 CONCLUSION & FUTURE WORK

In this paper, we proposed a machine learning pipeline for automated analysis of loosely structured information in TI feeds. We also demonstrated the feasibility of using Doc2Vec and BERT based embeddings to detect vulnerability exploitation events from TI feeds. TIBERT can automatically update CVSS/EPSS scores by leveraging rich temporal data from TI feeds. This paves the way toward improving the quality of CVSS/EPSS scores with minimal effort from security experts.

There are limitations in our design and evaluation that present opportunities for future work. We discuss them below:

- **Quality of ground truth:** Our supervised machine learning classifiers heavily rely on the quality of manual ground truth labeling exercise that is somewhat subjective. While the high inter-rate agreement score indicated absence of significant bias, there is room for a more rigorous evaluation of the quality of ground truth labels. It might be possible to sidestep the problem of needing manually labeled ground truth using unsupervised or self-supervised learning techniques.
- **Better accuracy using enhanced embeddings:** There is a significant room to improve the accuracy of our machine learning pipeline for exploitation detection from TI feeds. We are interested in exploring more advanced embedding techniques to this end. For example, there is opportunity to fine-tune a pre-trained embedding for specific tasks in a supervised or semi-supervised manner. As another example, there is opportunity to train embeddings on a larger corpus of threat information not only from TI feeds but also online and darknet forums [1, 2], relevant blogs [18] and social media posts [26], and even research literature [42] to improve their coverage and diversity of vocabulary.
- **Model updates:** Our longitudinal evaluation showed that the classifiers need to be updated as new types of threats emerge and new vocabulary is introduced over time. There is opportunity to use active and online learning techniques to effectively adapt our machine learning pipeline in a streaming fashion without needing to train it from scratch.
- **Other applications:** Our machine learning pipeline can be readily adapted to detect other relevant vulnerability lifecycle events such as weaponization and patching.

REFERENCES

[1] Mohammed Almukaynizi, Eric Nunes, Krishna Dharaiya, Manoj Senguttuvan, Jana Shakarian, and Paulo Shakarian. 2017. Proactive Identification of Exploits in the Wild Through Vulnerability Mentions Online. In *International Conference on Cyber Conflict (CyCon)*. NATO, Tallinn, 82–88.

[2] Mohammed Almukaynizi, Eric Nunes, Krishna Dharaiya, Manoj Senguttuvan, Jana Shakarian, and Paulo Shakarian. 2019. Patch before exploited: An approach to identify targeted software vulnerabilities. In *AI in Cybersecurity, Springer Intelligent Systems Reference Library*. Springer, Switzerland.

[3] X. Bouwman, V. Le Pochat, P. Foremski, T. Van Goethem, C. Hernandez-Ganan, et al. 2022. Helping hands: Measuring the impact of a large threat intelligence sharing community. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA. https://www.usenix.org/conference/usenixsecurity22/presentation/bouwman

[4] Mehran Bozorgi, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. 2010. Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits. In *KDD*. ACM, Washington, DC, 105–114.

[5] David Chismon and Martyn Ruks. 2015. Threat Intelligence: Collecting, Analysing, Evaluating. Whitepaper: MWR InfoSecurity.

[6] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computer Linguistics, 276–286.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

[8] Yong Fang, Yongcheng Liu, Cheng Huang, and Liang Liu. 2020. FastEmbed: Predicting vulnerability exploitation possibility based on ensemble machine learning algorithm. *PLoS One* 15, 2 (2020), e0228439.

[9] Álvaro Feal, Pelayo Vallina, Julien Gamba, Sergio Pastrana, Antonio Nappa, Oliver Hohlfeld, Narseo Vallina-Rodriguez, and Juan Tapiador. 2021. Blocklist babel: On the transparency and dynamics of open source blocklisting. *IEEE Transactions on Network and Service Management* (2021).

[10] FIRST. [n.d.]. Common Vulnerability Scoring System (CVSS). https://www.first.org/cvss/.

[11] FIRST. [n.d.]. Exploit Prediction Scoring System. https://www.first.org/epss/.

[12] FIRST.org. [n.d.]. Traffic Light Protocol. https://www.first.org/tlp/.

[13] Stefan Frei, Martin May, Ulrich Fiedler, and Bernhard Plattner. 2006. Large-Scale Vulnerability Analysis. In *SIGCOMM workshop on Large-scale attack defense*. ACM, Pisa, Italy, 131–138.

[14] Grandview Research. 2017. Threat Intelligence Market Size & Share: Industry Report, 2014-2025. https://www.grandviewresearch.com/industry-analysis/threat-intelligence-market

[15] Jay Jacobs, Sasha Romanosky, Benjamin Edwards, Michael Roytman, and Idris Adjerid. 2021. Exploit prediction scoring system (EPSS). TPRC48: The 48th Research Conference on Communication, Information and Internet Policy.

[16] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. PMLR, 1188–1196.

[17] Vector Guo Li, Matthew Dunn, Paul Pearce, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2019. Reading the Tea leaves: A Comparative Analysis of Threat Intelligence. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX, Santa Clara, 851–867.

[18] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem Beyah. 2016. Acing the IoC game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Vienna, 755–766.

[19] Ghita Mezzour, L. Richard Carley, and Kathleen M. Carley. 2014. Longitudinal analysis of a large corpus of cyber threat descriptions. *Journal of Computer Virology and Hacking Techniques* 12 (2014), 11–22.

[20] MISP. [n.d.]. MISP - Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing. https://www.misp-project.org.

[21] MISP. [n.d.]. MISP Default Feeds. https://www.misp-project.org/feeds/.

[22] MISP. [n.d.]. MISP taxonomies and classification as machine tags. https://www.circl.lu/doc/misp-taxonomies/.

[23] Lior Perry, Bracha Shapira, and Rami Puzis. 2019. No-doubt: Attack attribution based on threat intelligence reports. In *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, IEEE, virtual event, 80–85.

[24] Andreas Pitsillidis, Chris Kanich, Geoffrey M. Voelker, Kirill Levchenko, and Stefan Savage. 2012. Taster's choice: a comparative analysis of spam feeds. In *Internet Measurement Conference (IMC)*. ACM, Boston, MA, 427–440.

[25] Sivaramakrishnan Ramanathan, Jelena Mirkovic, and Minlan Yu. 2020. BLAG: Improving the Accuracy of Blacklists. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, San Diego, CA, 1–15.

[26] Carl Sabottke, Octavian Suciu, and Tudor Dumitras. 2015. Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits. In *24th USENIX Security Symposium (USENIX Security 15)*.

[27] Muhammad Shahzad, M Zubair Shafiq, and Alex X Liu. 2019. Large scale characterization of software vulnerability life cycles. *IEEE Transactions on Dependable and Secure Computing* 17, 4 (2019), 730–744.

[28] Yun Shen and Gianluca Stringhini. 2019. Attack2vec: Leveraging temporal word embeddings to understand the evolution of cyberattacks. In *28th USENIX Security Symposium*.

[29] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Cranor, and Jason Hong Chengshan Zhang. 2009. An Empirical Analysis of Phishing Blacklists. In *Conference on Email and Anti-Spam (CEAS)*.

[30] Jennifer Sleeman, Tim Finin, Milton Halem, et al. 2020. Temporal Understanding of Cybersecurity Threats. In *IEEE International Conference on Big Data Security on Cloud*. IEEE, Baltimore, 115–121.

[31] STIX. [n.d.]. Introduction to STIX. https://oasis-open.github.io/cti-documentation/stix/intro.html.

[32] Octavian Suciu, Connor Nelson, Zhuoer Lyu, Tiffany Bao, and Tudor Dumitras. 2021. Expected exploitability: Predicting the development of functional vulnerability exploits. arXiv preprint arXiv:2102.07869.

[33] Nazgol Tavabi, Palash Goyal, Mohammed Almukaynizi, Paulo Shakarian, and Kristina Lerman. 2018. DarkEmbed: Exploit Prediction With Neural Language Models. In *AAAI Conference on Innovative Applications of Artificial Intelligence*.

[34] Kurt Thomas, Rony Amira, Adi Ben-Yoash, Ori Folger, Amir Hardon, Ari Berger, Elie Bursztein, and Michael Bailey. 2016. The Abuse Sharing Economy: Understanding the Limits of Threat Exchanges. In *International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*.

[35] Wiem Tounsi and Helmi Rais. 2018. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & security* 72 (2018), 212–233.

[36] Ryan Trost. 2014. Threat Intelligence Library - A New Revolutionary Technology to Enhance the SOC Battle Rhythm. Black Hat USA.

[37] Jesse Vig. 2019. A Multiscale Visualization of Attention in the Transformer Model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computer Linguistics, 37–42.

[38] Jesse Vig. 2019. A Multiscale Visualization of Attention in the Transformer Model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. ACL, 37–42. https://www.aclweb.org/anthology/P19-3007 https://github.com/jessevig/bertviz.

[39] Brandon Wang, Xiaoye Li, Leandro P. de Aguiar, Daniel S. Menasche, and Zubair Shafiq. 2017. Characterizing and Modeling Patching Practices of Industrial Control Systems. In *ACM SIGMETRICS*.

[40] Chaowei Xiao, Armin Sarabi, Yang Liu, Bo Li, Mingyan Liu, and Tudor Dumitras. 2018. From Patching Delays to Infection Symptoms: Using Risk Profiles for an Early Discovery of Vulnerabilities Exploited in the Wild. In *27th USENIX Security Symposium*.

[41] Han Xiao. 2018. bert-as-service. https://github.com/hanxiao/bert-as-service.

[42] Ziyun Zhu and Tudor Dumitraş. 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*.

[43] Ziyun Zhu and Tudor Dumitras. 2018. ChainSmith: Automatically Learning the Semantics of Malicious Campaigns by Mining Threat Intelligence Reports. In *IEEE European Symposium on Security and Privacy (EuroS&P)*.

A TI FEED ORGANIZATIONS

Table 8 summarizes some of the key characteristics of the providers of the top-10 feeds considered in this work. Some providers are national teams, whereas others are public or private corporations.

Table 8: General description of anonymized top-10 TI feed providers.

Provider	Description
Feed 1	National Computer Emergency Response Team (CERT) of an European country
Feed 2	Computer Incident Response Center (CIRC) of an European country
Feed 3	Lists of IPs associated to malicious activities, to be blocked or filtered as reported by a threat intelligence firm
Feed 4	List of events related to malware activity as observed by a large international corporation
Feed 5	Open source intelligence gathered by an European national authority
Feed 6	National private sharing group with large private corporations as participants
Feed 7	List of indicators about the dissemination of a specific malware family by a private threat intelligence corporation
Feed 8	List of indicators about the dissemination of a specific malware family by a private threat intelligence corporation
Feed 9	International private sharing group with cybersecurity experts as participants
Feed 10	List of indicators about the dissemination of a specific malware family by a private threat intelligence corporation

B RULE MINING FOR GROUND TRUTH

Table 9 reports the list of top most frequent rules used for labeling exploitation events. Such rules were mined using the *apriori* algorithm, based on a set of manually labeled events. In the table, for each rule we indicate the number of tags involved in the rule, its confidence and lift.

Table 9: Top most frequent rules labeling exploitation.

Rules	Tags	Confidence	Lift
(threat:ransomware)+(malware_classification:malware-category="ransomware")	2	1.00	3.58
(circl:incident-classification="spam")+(malware_classification:malware-category="trojan")	2	1.00	3.58
(veris:confidence="none")+(cssa:origin="other")+(malware_classification:malware-category="ransomware")+(cssa:sharing-class="tier-0")	4	1.00	3.58
(cssa:origin="other")+(malware_classification:malware-category="ransomware")+(cssa:sharing-class="tier-0")	3	1.00	3.58
(veris:confidence="none")+(cssa:origin="other")+(malware_classification:malware-category="ransomware")	3	1.00	3.58
(veris:confidence="none")+(malware_classification:malware-category="ransomware")+(cssa:sharing-class="tier-0")	3	1.00	3.58
(veris:confidence="none")+(malware_classification:malware-category="ransomware")	2	1.00	3.58
(cssa:origin="other")+(malware_classification:malware-category="ransomware")	2	1.00	3.58
(malware_classification:malware-category="ransomware")+(cssa:sharing-class="tier-0")	2	1.00	3.58

Continued on next page

Table 9 – continued from previous page

Rules	Tags	Confidence Score	Lift Score
(ms-caro-malware:malware-platform="macos_x")	1	1.00	3.43
(type:osint)+(cssa:origin="other")+(malware_classification:malware-category="ransomware")+(cssa:sharing-class="tier-0")	4	1.00	3.58
(veris:confidence="none")+(type:osint)+(cssa:origin="other")+(malware_classification:malware-category="ransomware")	4	1.00	3.58
(type:osint)+(veris:confidence="none")+(malware_classification:malware-category="ransomware")+(cssa:sharing-class="tier-0")	4	1.00	3.58
(type:osint)+(cssa:origin="other")+(malware_classification:malware-category="ransomware")	3	1.00	3.58
(type:osint)+(malware_classification:malware-category="ransomware")+(cssa:sharing-class="tier-0")	3	1.00	3.58
(apt)+(kill-chain:installation)+(kill-chain:command and control)	3	1.00	3.58
(type:osint)+(veris:confidence="none")+(malware_classification:malware-category="ransomware")	3	1.00	3.58
(veris:asset:variety="s - scada")+(circl:topic="industry")	2	1.00	3.58
(circl:incident-classification="system-compromise")+(circl:incident-classification="vulnerability")	2	1.00	3.58
(apt)+(kill-chain:installation)	2	1.00	3.58
(malspam)+(banker)	2	1.00	3.58
(apt)+(kill-chain:command and control)	2	1.00	3.58
(veris:asset:variety="s - scada")	1	1.00	3.43
(veris:asset:variety="u - pos terminal")	1	1.00	3.43
(enisa:nefarious-activity-abuse="spear-phishing-attacks")	1	1.00	3.43
(circl:incident-classification="spam")+(threat:ransomware)+(malware_classification:malware-category="ransomware")	3	1.00	3.58
(circl:incident-classification="phishing")+(malware_classification:malware-category="ransomware")+(phishing)	3	1.00	3.58
(circl:incident-classification="phishing")+(enisa:nefarious-activity-abuse="phishing-attacks")+(phishing)	3	1.00	3.58
(cybercrime)+(sectorfinancial)+(kill-chain:installation)	3	1.00	3.58
(apt)+(ncsc-nl-nds:search="historic")+(osint)	3	1.00	3.58
(malware_classification:malware-category="ransomware")+(circl:incident-classification="malware")+(malware_classification:malware-category="worm")	3	1.00	3.58
(circl:incident-classification="phishing")+(circl:topic="finance")+(circl:incident-classification="malware")	3	1.00	3.58
(threat:ransomware)+(ecsirt:malicious-code="ransomware")+(malware_classification:malware-category="ransomware")	3	1.00	3.58
(sectorfinancial)+(kill-chain:installation)	2	1.00	3.58
(malware_classification:malware-category="ransomware")+(malware_classification:malware-category="worm")	2	1.00	3.58

Continued on next page

Table 9 – continued from previous page

Rules	Tags	Confidence Score	Lift Score
(osint)+(ncsc-nl-nds:search="historic")	2	1.00	3.58
(type:osint)+(veris:action:malware:variety="ransomware")	2	1.00	3.58
(veris:actor:motive="financial")+(circl:topic="finance")	2	1.00	3.58
(threat:ransomware)	1	0.86	2.94
(circl:incident-classification="phishing")+(phishing)	2	0.83	2.98
(circl:incident-classification="phishing")+(enisa:nefarious-activity-abuse="phishing-attacks")	2	0.83	2.98
(malware_classification:malware-category="ransomware")+(phishing)	2	0.83	2.98
(enisa:nefarious-activity-abuse="phishing-attacks")	1	0.83	2.86
(veris:actor:motive="financial")	1	0.83	2.86
(threat:snake)	1	0.83	2.86
(circl:incident-classification="phishing")+(malware_classification:malware-category="trojan")	2	0.80	2.86
(type:osint)+(ecsirt:malicious-code="ransomware")	2	0.80	2.86
(malspam)+(nymaim)	2	0.80	2.86
(type:osint)+(threat:snake)	2	0.80	2.86
(ncsc-nl-nds:search="historic")+(ncsc-nl-ndn:feed="selected")	2	0.80	2.86
(apt)+(osint)	2	0.80	2.86
(relcircl)	1	0.80	2.74
(circl:topic="finance")+(circl:incident-classification="malware")	2	0.75	2.68
(malware_classification:malware-category="spyware")	1	0.75	2.57
(veris:action:social:target="finance")+(circl:topic="finance")	2	0.75	2.68
(certsi:critical-sector="energy")	1	0.75	2.57
(danabot)	1	0.75	2.57
(threat type:rat)	1	0.75	2.57
(sectorfinancial)	1	0.75	2.57
(type:osint)+(malware_classification:malware-category="ransomware")	2	0.72	2.58
(circl:incident-classification="spam")	1	0.71	2.45
(type:osint)+(circl:topic="finance")	2	0.71	2.55
(circl:incident-classification="phishing")+(circl:incident-classification="malware")	2	0.71	2.55
(malware_classification:malware-category="botnet")	1	0.71	2.45
(veris:action:social:variety="phishing")	1	0.71	2.45
(circl:topic="finance")	1	0.70	2.41
(ncsc-nl-nds:search="historic")	1	0.70	2.40
			End of table