2018 IEEE European Symposium on Security and Privacy

ChainSmith: Automatically Learning the Semantics of Malicious Campaigns by Mining Threat Intelligence Reports

Ziyun Zhu, Tudor Dumitraș
University of Maryland, College Park, MD, USA
Email: {zhuziyun, tdumitra}@umiacs.umd.edu

***Abstract*—Modern cyber attacks consist of a series of steps and are generally part of larger campaigns. Large-scale field data provides a quantitative measurement of these campaigns. On the other hand, security practitioners extract and report qualitative campaign characteristics manually. Linking the two sources provides new insights about attacker strategies from measurements. However, this is a time-consuming task because qualitative measurements are generally reported in natural language and are not machine-readable.**

We propose an approach to bridge measurement data with manual analysis. We borrow the idea from threat intelligence: we define campaigns using a 4-stage model, and describe each stage using IOCs (indicators of compromise), e.g. URLs and IP addresses. We train a multi-class classifier to extract IOCs and further categorize them into different stages. We implement these ideas in a system called ChainSmith. Our system can achieve 91.9% precision and 97.8% recall in extracting IOCs, and can determine the campaign roles for 86.2% of IOCs with 78.2% precision and 80.7% recall. We run ChainSmith on 14,155 online security articles, from which we collect 24,653 IOCs. The semantic roles allow us to link manual attack analysis with large scale field measurements. In particular, we study the effectiveness of different persuasion techniques used on enticing user to download the payloads. We find that the campaign usually starts from social engineering and "missing codec" ruse is a common persuasion technique that generates the most suspicious downloads each day.

1. Introduction

The field of threat intelligence focuses on gathering and analyzing information about security threats to understand adversary strategies and respond to attacks. The premise of threat intelligence is that sharing the technical details of breaches and attacks makes everybody safer because it makes it harder for attackers to reuse attack methods and artifacts, thus increasing their *work factor* [57]. Threat intelligence is a billion-dollar industry [34] and has introduced standards [11], [14], [13], and information-sharing platforms [28], [6], [15], [8]. These standards define threats using *indicators of compromise* (IOCs). For example, to represent recent measurements of *malware delivery networks* [39], [61], [36], [40] using these standards, we can identify IOCs such as file hashes of malware droppers and of their payloads, URLs and IP addresses of command-and-control (C&C) servers, or names of malware families and exploit kits.

However, the *utility* of threat intelligence is not well understood. For example, malware samples are frequently repacked [22] and malicious domains do not remain active for a long time [38]. In consequence, security tools based on IOC detection may not provide an adequate defense. We propose a novel use of threat intelligence: producing *new insights* about the threat landscape by connecting the intelligence with measurement data, collected in the field. For example, we can identify the methods that underground actors employ for baiting users to run malware or to visit an exploitation site (e.g. spam, malvertising, compromised sites), by analyzing the threat intelligence, and we can measure their effectiveness in the real world, by quantifying the resulting volume of malware downloads recorded in the field data.

Producing such insights automatically requires understanding the role of each IOC in a malicious campaign. For example, malware delivery campaigns consist of a sequence of actions, e.g. baiting, exploitation, installation, command & control, that are often performed by different actors. However, individual IOCs do not allow us to distinguish between components used for these various stages. Understanding the sequential actions of malicious campaigns requires careful manual analysis, and the results of this analysis are seldom encoded in a machine-readable format. However, these results are often published, either in practitioner-oriented journals such as Virus Bulletin, or on the blogs of analysts or security companies, such as Webroot or Trend Micro. These reports connect the dots between multiple attack identifiers and state the semantics of the corresponding campaigns in natural language.

It is challenging to extract useful information from security articles with off-the-shelf natural language processing (NLP) techniques. These techniques usually rely on the context of complete sentences, however in security articles IOCs are often included in bulleted lists and tables, while the relevant relations are discussed in the text. Moreover, the security arms race gives rise to a growing number of technical terms [65], while language models cannot usually handle previously unseen words. The recent work on mining security articles [65], [43], [53] does not extract the roles of IOCs in campaigns, and does not correlate the information

DOI 10.1109/EuroSP.2018.00039

extracted from natural language with measurement data.

We present the design and implementation of ChainSmith, a system that automatically extracts IOCs, as well as their corresponding campaign stages, from technical documents. ChainSmith implements new NLP techniques that address technical challenges specific to the security domain. To identify the stages accurately, we propose a malware delivery model with four stages. Our model is inspired by a recent standard for sharing threat intelligence [13], but we modify it to account for the writing style of security articles and for the characteristics of malware delivery campaigns. We also collect the ground truth for training and evaluating ChainSmith, using a Web application for manually labeling articles according to our malware-delivery model.

We apply ChainSmith to 14,155 *unstructured* technical reports. manually extracting the details of campaigns from such a large corpus is infeasible, which illustrates the need for automated techniques. With our system, we extract 24,653 IOCs and are able to classify 20,774 of them into their corresponding campaign stage. We further classify IOCs into 8,902 campaign groups based on the campaign stage and blog post time. We apply the campaign information to study the influence and effectiveness of different persuasion techniques on the subsequent download behaviors. Unlike prior work [50], which manually labels the attacks that belong to social engineering, we automate this annotation process. We also provide new insights about individual campaigns (e.g. underground business relationships), by connecting the dots among multiple articles. We find that the "missing codec" ruse is most common in malware delivery campaigns, resulting in 4× more download events per day than any other technique. However, social network messaging and fake update notifications are the most successful techniques which result in the most downloads per payload.

In summary, we make the following contributions:

- We design ChainSmith, an IOC extraction system that collects indicators from security articles and classifies them into different campaign stages.
- We evaluate the effect of different persuasion techniques on the subsequent payload delivery in the wild, by connecting campaigns from blog posts and the real-world telemetry data.
- We report new findings about the underground business relationship and the characteristics of campaign infrastructures. This allows us to assess the utility of threat intelligence.
- We set up a Web application to release the latest data from ChainSmith to further stimulate the research on threat intelligence.

The rest of paper is organized as follows. In Section 2, we state our goals and give the definition of payload delivery model. In Section 3, we describe the design of ChainSmith. In Section 4, we evaluate the performance of ChainSmith and analyze the effect of different baiting techniques on binary delivery. Finally, we discuss the lessons learned from this study and the related work in Section 5 and 6 respectively.

2. Problem statement

Unlike measurement data, which is collected automatically, threat intelligence reflects a careful manual analysis of security threats, with conclusions drawing from the human knowledge and intuition of expert analysts. These conclusions can be encoded in *indicators of compromise* (IOCs). An IOC is an observable attack artifact—e.g., a file (identified by name or hash), a URL, an IP address—that the analyst has linked to a given security threat. To understand the strategies of an *attack group*, the analysts identify multiple IOCs that correspond to a malicious *campaign* and the *role of each IOC* in the attack—e.g. whether the URL points to an exploitation site or command-and-control server. The analysts publish this threat intelligence on technical blogs and in industry reports. Additionally, some of the indicators identified during these investigations are also made available in commercial or open-source IOC feeds. These indicators are then used for detecting security threats, for forensic investigations after data breaches, and for attack attribution. However, the utility of threat intelligence is not well understood. For example, malware samples are frequently repacked [22] and malicious domains do not remain active for a long time [38], so security tools based on IOC detection may not provide an adequate defense against these malicious campaigns.

We explore a novel use of threat intelligence by linking the threat intelligence to the measurement data. Threat intelligence describes the semantics of security threats in detail, but it is available for only a few threats that the analysts consider representative. In contrast, measurement data can be collected systematically but cannot determine the semantics of the indicators recorded in an automated fashion. By linking the two independent sources, we gain new *insights* about attacker strategies from measurement data and quantitative analysis on the impact of attacks mentioned in threat intelligence.

Application example: understanding malware delivery campaigns. A *malware delivery campaign* [40], [39], [35], [56], [49] involves multiple steps: baiting the user to perform a risky action, such as opening an email attachment or visiting an unknown site; exploiting an unpatched vulnerability on the user's computer; and installing a *dropper* [39], which then downloads additional malware. If one step fails, the malware delivery fails; for example, an exploit kit [30] is useless unless the attacker can lure users to the landing page. The technical challenges for implementing each of these steps make it difficult for an individual actor to execute a campaign from end to end. Instead, malware creators rely on a thriving underground economy [22], [30], [36], [61], where specialized services are provided for a fee and one can quickly set up a campaign by summoning the hacking expertise of third parties.

In this ecosystem, we call the actors that provide malware-delivery services *suppliers* and the actors that uti-

lize the service *customers*. Understanding the business relationships among suppliers and their customers can uncover important dependencies among these actors and may guide effective interventions [62]. For example, we distinguish between *tier 1 suppliers*, which are directly involved in user baiting and exploitation, and other suppliers that act as middlemen. Because only tier 1 suppliers bring victims to the delivery network, the effectiveness of their techniques is critical for the entire ecosystem.

However, neither threat intelligence nor measurement studies can provide a complete picture of malware delivery campaigns. Threat intelligence reports analyze the strategies employed in a campaign, but do not assess the effectiveness of these strategies in a systematic manner. For example, the reports may discuss a tier-1 supplier's baiting methods (e.g. spam, malvertising, compromised sites) but do not quantify the volume of malware downloads that these methods produce in the wild, as this would require comprehensive field data. Field measurements can record download events systematically, and provide various IOCs, but they may not be able to indicate the role of them. For example, the measurement data does not usually indicate whether a dropper corresponds to a tier 1 supplier, nor does it indicate how the dropper was installed on the host. Moreover, measurement studies often focus on a single phase of the campaign—e.g., baiting [42], [44], exploitation [27], [26], or installation [39], [40]—and do not shed light on the strategies of long-lasting campaigns. These insights require the ability to connect the threat intelligence with the field measurements.

Challenges. The key challenges to automating the insight generation process are the semantics of security threats in a machine-readable format, extracting them from the available threat intelligence, and developing analytics that combine the threat intelligence with measurement data. Existing standards for sharing threat intelligence like OpenIOC [11] and STIX [13] are incomplete and do not capture important concepts, such as the persuasion strategy employed by attackers to convince users into running malware or to lure them to an exploit kit. Worse, the IOC feeds currently available omit critical information, even when this information can be represented in the current standards. For example, STIX specifies the `kill_chain_phase` IOC attribute, which indicates the role of the indicator in the campaign; however, current feeds omit this attribute. Instead, this information is usually provided in the threat intelligence reports in natural language. However, prior work on mining security articles [65], [43] does not extract the roles of IOCs in campaigns, and does not correlate the information extracted from natural language with field measurement data.

To overcome these challenges, we aim to extract the threat and campaign semantics from intelligence reports written in colloquial English. In doing so, we further identify three technical challenges that are specific to security discourses. First, natural language processing (NLP) techniques usually rely on the context of complete sentences. However in security articles IOCs are often included in bulleted lists and tables, while the relevant relations are discussed in the text. Second, some indicators are presented in an obfuscated form. For example, to prevent the mis-clicking of a malicious site, malicious links are transformed to use "hxxp" instead of "http", and "[dot]" or (dot)" instead of ".", and authors use different names and delimiters for malware families. Third, the security arms race gives rise to a growing number of technical terms [65], while language models cannot usually handle previously unseen words.

Goals. Our *first goal* is to build a generic system that mines descriptions of security threats written in natural language, and extracts threat intelligence in a format that enables correlations and complements measurement data. Specifically, we aim to extract both IOCs *and* their roles in an attack, which allows us to augment the measurement data with the semantics of security threats observed. To this end, we develop new NLP techniques that address technical challenges specific to the security domain.

Our *second goal* is to apply our system to the concrete problem of understanding malware delivery campaigns and to gain new insights into this security threat. To this end, we propose a model of malware delivery campaigns (detailed in Section 2.1), and we build a Web application for manually labeling articles according to this model. We also correlate the threat intelligence with a comprehensive data set of download events [39], observed on 5M hosts. With this combined data set, we ask the following research questions:

- RQ1 What is the relative effectiveness of various persuasion techniques in generating downloads on real-world hosts?
- RQ2 What roles do various attack groups play in the malware delivery ecosystem, and what are the business relationships among them?
- RQ3 How long do malware delivery campaigns remain active?
- RQ4 How long do their support infrastructures remain active?

In addition to our findings and their actionable implications, we expect that the process of answering these research questions will provide important lessons about the utility of threat intelligence—both for the tasks it was originally meant to address and for the novel application proposed in this paper. This is our *third* and final *goal*.

Non-goals. We do not aim to validate the threat intelligence published in technical blogs and industry reports with field measurement data. We are unable to assess the quality of the information discussed in these articles because we lack a corpus of known malware delivery campaigns; the data sets currently available focus on malware samples and malicious domains, rather than end-to-end campaigns. Moreover, the threat intelligence and the field measurements are collected from different observation perspectives and likely reflect different aspects of the underground economy. For the same reason, we do not aim to systematically attach semantics to all the measurements. Instead, our goal is to gain new insights about the malware delivery ecosystem.

2.1. Malware delivery model

The emerging standards for sharing threat intelligence [11], [13], [14] are based on the observation that cyber attacks often follow certain high-level patterns. This allows analysts to define generic models for representing and sharing the information.

In particular, the STIX standard [13] has been adopted by an increasing number of security products [1]. STIX defines a comprehensive schema that uses IOCs to describe campaigns [13]. In this schema, a `campaign` consists of a set of `indicators` which specify an attack pattern. An `Indicator` is then defined to be a set of `observables`, which includes all the resources and infrastructure in the attack pattern. An `Observable`, defined in CybOX (v2.1) [14], contains 88 different types including URI, IP address, file, registry key, etc.. To add semantics for the attack pattern, STIX defines `kill_chain_phase` as an attribute for `indicator`. A *kill chain*[1] breaks down an attack in a sequence of stages. The original definition of the cyber kill chain [33] specified 7 stages: reconnaissance, weaponization, delivery, exploitation, installation, command & control and actions on objective.

Malware delivery campaigns. To represent malware delivery campaigns, we adopt three stages from the STIX data model: *exploitation*, *installation*, and *command & control*. Additionally, we redefine the first stage of these campaigns based on our observation that, while attackers will sometimes gain an initial foothold on a host by delivering malicious files (e.g. as email attachments), in other causes they rely on malvertising to lure users to an exploit kit's landing page or to persuade them to download and install a dropper. We therefore replace the delivery stage with a *baiting* stage, which captures all these strategies.

There are 4 different types of `indicators`, which cover the most common and important stages involved in malware delivery.

1) *Baiting*: This is the first stage of delivery campaign. The most common approaches to draw users to the malware delivery chain are email spam, malvertising and compromised sites. Once the user clicks on a link from a spam message, or visits a compromised site, the browser will be redirected to a malicious site. Before landing on the exploit server, users might be redirected through several relay pages, which are included this stage.
2) *Exploitation*: After user is redirected to the landing page of an exploit kit, the exploit server fingerprints the browser and the installed plugins and tries to identify open vulnerabilities. The server then tries to exploit a vulnerability and deliver a payload.
3) *Installation*: A malicious payload is downloaded on the end host, and the exploit server installs and executes the malware. Without using exploits, attackers can also lure users to install and execute the malware through social engineering.
4) *Command & Control*: The executed malware contacts its command and control server and receives remote commands. The commands may involve downloading the next stage of the malware, dropping unrelated samples (e.g. in the case of a Pay-per-Install infrastructure [61]) or performing other malicious actions (e.g. spam, DDoS).

1. "Kill chain" is a military term describing the stages of an attack that results in the destruction of a target (i.e. a "kill"). Separating the attack stages is helpful for identifying different defensive techniques that could break the chain and disrupt the attack.

TABLE 1: IOCs and indicator phase in malware delivery model.

Phase	IOCs	IOC Type
baiting	`attachment_hash`	hash
	`attachment_family`	malware family
	`server_URL`	URL
	`server_IP`	IP
exploitation	`exploit_site_URL`	URL
	`exploit_site_IP`	IP
	`exploited_vulnerability`	vulnerability
	`exploit_kit`	exploit kit
installation	`malware_hash`	hash
	`malware_family`	malware family
C&C	`C&C_server_URL`	URL
	`C&C_server_IP`	IP

These stages are not necessary for every payload delivery. For example, campaigns can entice users to download and install the payload by users themselves through social engineering. In this case, the exploit stage would be unnecessary.

Table 1 lists all the observables we use for each type of indicators. `Observable` contains 6 types, i.e. *URL*, *IP address*, *file hash*, *malware family*, *exploit kit*, and *vulnerability*, in which CybOX `observable` includes the former 3 types. Figure 1 shows one example of how the malware delivery schema can well fit the actual blog posts. Our payload delivery model simplifies the data model in STIX in that (1) we primarily focus on network activity in the campaign rather than the behavior on the victim machine, and (2) many `observables` are unlikely to be discussed in security articles and cannot be used to uniquely determine the campaign. Note that the term IOC (indicator of compromise) is equivalent to the concept of an observable in STIX. In order not to confuse the reader, we use the terms IOC and indicator phase instead of STIX's observable and indicator in rest of this paper.

A *campaign group* corresponds to the actors of a campaign, represented by a set of IOCs. A *supplier* corresponds to an actor that provides delivery services by controlling droppers in the field. A *tier-1 supplier* is a supplier that operates baiting and exploitation domains.

High-risk binaries. We utilize the VirusTotal service [17] to assess the binaries recorded in the measurement data. VirusTotal provides file scan reports for up to 54 anti-virus (AV) products. In line with prior work [40], if a binary receives more than 30% anti-virus detection from VirusTotal, then we considered it a *high-risk binary*. Because the blog posts discuss both malware and potentially unwanted programs

...
Upon clicking on the links, users are exposed to the client-side exploits served by the latest version of the Black Hole Exploit Kit.
...
Sample compromised URLs used in the campaign:
hxxp://www.alacinc.org.nz/impdiscm.html;
...
Client-side exploits serving URLs:
hxxp://netgear-india.net/detects/discover-important_message.php;
Upon loading, these URLs attempt to exploit CVE-2010-0188 by

dropping a malicious PDF file.
Sample detection rate for the dropped malware:
MD5: 80601551f1c83ee326b3094e468c6b42
detected by 4 out of 44 antivirus scanners as
UDS:DangerousObject.Multi.Generic.
Upon execution, the sample phones back to
200.169.13.84:8080/AJtw/UCyqrDAA/Ud+asDAA

(a) Blog post from Webroot [18]

```
{
  baiting: {
    server_URL: ["www.alacinc.org.nz"]
  },
  exploitation: {
     server_URL: ["netgear-india.net"],
     exploited_vulnerability: ["CVE-2010-0188"],
     exploit_kit: ["Black Hole"]
  },
  installation: {
     malware_hash: ["80601551f1c83ee326b3094e468c6b42"],
     malware_family: ["UDS:DangerousObject.Multi.Generic"]
  },
  C&C: {
     C&C_server_IP: ["200.169.13.84:8080"]
  }
}
```

(b) Structured malware delivery campaign

Figure 1: An example of mapping unstructured blog post to structured schema for malware delivery.

(PUPs), we do not aim to distinguish if a specific binary is malware or PUP. Instead, we focus on binaries that present a *high risk* to the end hosts because (1) they are highlighted as security threats in the articles we analyze, and (2) they have a high detection rate among AV products. A *high-risk download* is a download event with a high-risk binary payload.

Generality. Our system for mining security articles, described in Section 3, requires a model of the malicious activity to extract the relevant semantics. The model defined in this section is specific to malware delivery campaigns; however, our NLP techniques are generic and may be applied to other models. The results in this paper could be extended to other cyber threats by defining a pertinent model, for example by starting from an existing standard such as STIX.

2.2. Threats to validity

First, because the existing threat intelligence does not cover all ongoing campaigns, and anti-virus programs may not detect malware as soon as it is released in the wild, we cannot determine which binaries present a low risk. A binary with no IOCs or VirusTotal detections may be either a benign program or a false negative. However, if we find a suspicious binary in both threat intelligence reports and in the measurement data, this represents a confirmation from at least two independent sources. This increases our confidence in the fact that the binary presents a high risk to end hosts.

Second, threat intelligence reports may be incorrect [7], which introduces noise in the extracted IOCs. This threat also affects other empirical results from prior work, as data collected in the wild is likely to blend normal activities with attacks. This makes it difficult to obtain a clean ground truth for security research [29], [59]. To mitigate this, we do not detect threats based on specific IOCs. Instead, we focus on aggregating data from multiple articles and from download events that occur on multiple hosts. This approach minimizes the impact of individual false positives.

Finally, since our measurement data was collected using Symantec's WINE platform [25], it does not cover the whole Internet. We select this data set because (1) the security telemetry in WINE is similar to the data available to other security vendors and (2) the data is collected on a large number of hosts (5M) and is representative for the events that Symantec—a major security vendor—observes in the wild [54]. Additionally, the data is collected on hosts where there is an anti-virus program, which blocks the malicious activities. It prevents the measurement from observing long running campaigns. We utilize this incomplete data to demonstrate how we can fill the gaps in measurement data with information from threat intelligence.

3. Experimental methods

In this section we describe the design of ChainSmith, a system that extracts and categorizes IOCs from security technical articles in colloquial English, according to the schema defined in Section 2.1. The key intuition behind ChainSmith is that the context words in adjacent sentences indicate the stage of a campaign, and the context words that directly relate to the IOC determine its level of maliciousness. In addition, we are able to group IOCs from different actors by considering the campaign stages and article post time.

Figure 2 shows the architecture of ChainSmith. The system consists of 6 components: *article crawler*, *expression detector*, *syntactic parser*, *semantic parser*, *named entity recognition*, and *IOC classifier*.

Article crawler. Just like the actors involved in malware delivery tend to specialize on narrow tasks, security analysts also focus on specific phases of the delivery campaigns. The full picture of an end-to-end campaign often emerges only after reading articles from multiple sources. We therefore implemented a generic crawler to collect security articles published online.

We use our crawler to mine 10 sources, listed in Table 2. We select these sources in that (1) the articles are

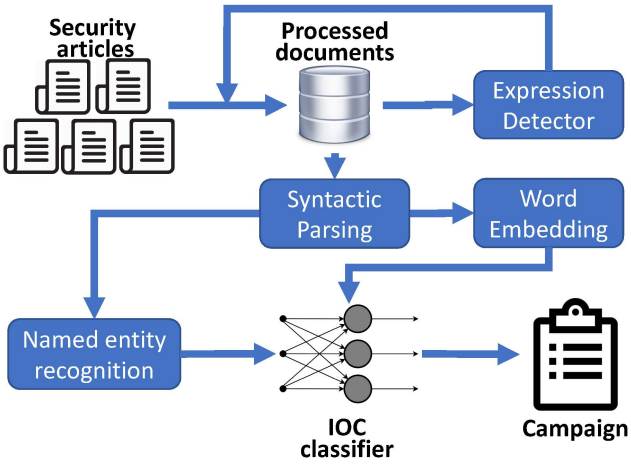


Figure 2: Architecture of ChainSmith. The article crawler module is not shown in this figure.

likely to include detailed information about the campaign, and (2) the sources are diverse—including news websites covering cyber threats, blogs from anti-virus companies, or the personal blogs of security experts.

TABLE 2: Summary of security articles. For some articles, we fail to identify the posting time.

Article source	Time span	Count
Forcepoint	2010-02-11 – 2017-05-13	227
Hexacorn	2011-10-01 – 2017-05-15	331
Malwarebytes	2012-04-20 – 2017-05-15	1590
Sophos	2000-11-24 – 2017-04-17	1171
Sucuri	2009-09-13 – 2017-05-12	927
TaoSecurity	2003-12-01 – 2017-05-08	2653
Trend Micro	2009-09-13 – 2017-05-15	1382
Virus Bulletin	2005-09-01 – 2016-01-29	601
WeLiveSecurity	2009-05-08 – 2017-05-16	4295
Webroot	2009-03-23 – 2017-05-14	978
Total	2000-11-24 – 2017-05-16	14155

Expression detector. Because security narratives often include multi-word expressions, we cannot simply analyze individual words. For example, *Black Hole* is the name of an exploit kit; in this context, the words *Black* and *Hole* are meaningless when considered separately. We identify multi-word expressions using the method proposed by Mikolov *et al.* [48], and we consider such expressions as single terms in the rest of our analysis pipeline. The intuition behind this method is that the joint probability of words is much higher than the product of the probability of individual words for multi-word expression. In the example, since the word "Black" is likely to appear together with "Hole", we identify "Black Hole" as a multi-word expression.

Syntactic parser. This component performs tokenization and dependency parsing, which processes the raw string to a tree structure that indicates the word dependency. This stage is equivalent to the lexical analysis and syntactic analysis in a compiler. Dependency parsing represents sentences as a directed graph to express the relationship between words. The

Figure 3: Example of dependency parsing, where the arc label is the dependency type [24].

parser identifies the grammatical relationship between words and labels each relationship with a type, which is essential in determining which words are directly related to the IOC candidates. For example, Figure 3 shows one example of dependency tree. From the typed dependencies, we know that the MD5 "`4462c5b3556c5cab5d90955b3faa19a8`" is the object of "drop", while the subject of "drop" is "campaign". Therefore, we learn that the given hash depends on "drop" instead of "on", even though they are equally close to the hash. Syntactic parsing is widely used in NLP applications and the state-of-the-art system has a reasonable experimental performance. In this paper, we use both Python NLTK [19] and Stanford CoreNLP [46] for syntactic parsing.

Semantic parser. Since syntactic parsing cannot tell the semantic similarity among words, to make the system more generic, we have to understand the meaning of words. Therefore in this step, we parse the words semantically by using the state-of-the-art algorithm `word2vec` [47], [48]. The words with close semantic meaning will be represented in a close position in the vector space, and the word vector is trained by maximizing the probability of a word given the words around it. ChainSmith uses dependency-based word embedding [41], which utilizes context *words+dependencies* instead of just context words. For example, for the MD5 in Figure 3, the context is `drop:dobj` rather than {"drop", "on"}. The benefit of dependency-based `word2vec` is that it learns functional similarity rather than topical similarity. We also record the embedding for word dependency pairs, which indicates the common context pattern of a word. For example, the dependency `drop:dobj` represents a direct object of *drop* and is usually the context for *malware*. In the embedding space, `drop:obj` is close to `drop:nsubjpass` (a passive nominal subject of *drop*), and `download:nsubjpass` (a passive nominal subject of *download*). In rest of the paper, we refer to this process as *dependency embedding* . We use both the word embedding and dependency embedding as features for topic and IOC classification.

Named entity recognition. Since IOCs strings tend to follow fixed patterns, classifying every word in an article is unnecessary. To remove the irrelevant words, we use regular expression to identify candidate IOCs. Table 3 lists the rules and additional constraints of recognizing each type of entity. We take into account certain writing practices common to security articles that do not follow general patterns. For example, to prevent the mis-clicking of malicious site, some bloggers use "hxxp" instead of "http", and "[dot]" or (dot)" instead of ".". Additionally, authors might use different delimiters for malware family names, e.g. replacing under-

TABLE 3: Rules of named entity recognition.

Type	Rules
URL	Identified top level domain must be found [16].
IPv4	Contains 4 digits (<256) and the address is not reserved [12].
hash	A hexadecimal string of length 32, 40 or 64.
family	Starts with malware types, and contains common delimiter [10].
EK	Either in defined dictionary [2] or ends with EK or exploit kit.
vuln.	CVE-[0-9]{4}-[0-9]{4,5}

scores with semicolons, so we accept the delimiter to be any one of underscore, slash, period, colon, and semicolon.

IOC classifier. The primary goal of this step is to identify whether the given word is an IOC as well as the stage of the campaign it belongs to. The intuition of this step is that the context words in close sentences determine the campaign stage and the dependencies in the same sentence decide whether the candidate is an IOC.

As discussed in Section 2.1, we model malware delivery campaigns as a chain of 4 phases: baiting, exploitation, installation, and command & control. We first train a classifier to check if the topic of a sentence falls in any of these 4 phases. Since the various words used to identify the campaign stage are not usually located in a single sentence, we construct the IOC classifier to consider features at both the dependency tree and sentence levels. This is in contrast to prior work by Liao et al. [43] which focused only on features at the sentence level.

First, we identify informative words in the sentence. We use Equation (1) to estimate the importance of a word in identifying topics,

$$S(w) = \max_{t \in T} \frac{p(w|t)}{p(w)} \tag{1}$$

where T is the set of malware campaign phases, w is word we evaluated, $p(w)$ is the probability of word w, and $p(w|t)$ is the probability that word w will be used for describing topic t. A higher word score means that w is more likely to be used in a certain campaign phase. For example, *iframe*, *src*, *malvertising* have high scores for the baiting phase, and *exploit-serving* has a high score for the exploitation phase. We consider *informative words* to be words with high score and high occurrence. For each sentence, we determine the *context* from 3 sources: (1) informative words from the current sentence; (2) informative words from previous sentences, if no informative words are found for the current sentence; and (3) informative words from the previous sentence that mention the same IOC. The motivation behind source 2 is that the topic of an article is usually consistent. For example, if the topic of the current sentence is command and control, then it is very likely that the next sentence is also discussing command and control. The last context source comes from the observation that the same IOC can be discussed repeatedly in different sentences.

We use 3 types of features to identify the sentence topic:

- Average word embedding of the context words.
- Average word embedding of the article title.
- Number of IOCs of each type.

Because the topic may not be mutually exclusive, we train 4 binary classifiers to identify topic probabilities. We implement the classifier using neural network with 1 hidden layer and 50 hidden nodes. To reduce the false positives, we skip the IOC candidates if no informative words are found.

Next, we use the result of topic classification as the feature for another neural net for IOC classification. The candidate IOCs extracted in the previous steps may not be related to malware delivery campaigns. For example, a file hash mentioned in the blog post may refer to a benign executable, such as a new patch or anti-virus product.

To verify that the named entities identified represent IOCs of a campaign, we train a multi-class classifier for each entity type (e.g. URL, IP address). The feature set includes:

- The probability of each topic.
- Average dependency embedding for all dependencies connected to the named entity.
- Type specific features (URL and IP only).

The first feature is useful in determining which campaign stage the named entity belongs to, and the rest of the features are effective in determining whether the named entity is an IOC. The type specific features include (1) the top level domain of the URL and (2) mis-click prevention strategies, e.g. “hxxp” and “[dot]”.

Instead of applying a softmax layer to the output layer, we use a logistic function to scale the output probability, such that the sum of the probabilities is not necessarily to be 1. To reduce the false positive rate, we add a special label “malicious” as another output, for the case when the classifier returns more than one label from a single sentence. For these IOCs, the classifier cannot reliably tell which stages they belong to. Therefore, rather than drop the results, we label them as “malicious”.

Campaign group identification. Technical articles are likely to mention both suppliers and customers in the underground economy. From our manual investigation, we find that different from scientific literature, security blogs usually discuss one observation from single campaign rather than investigate the problem in a broader view. For example, blog posts are likely to record the current status of single campaign as shown in Figure 1. However, unlike scientific literature, the comparison between different campaigns is usually absent from blog posts. Therefore, we assume that each technical article discusses one supplier and multiple customers. For each article, we group all the IOCs in baiting and exploitation phase as one group and consider the other IOCs as individual groups (one IOC for each group). In addition, as campaigns may change infrastructure and strategies over time to evade detection, we must connect campaigns from different articles in order to study the long-term behavior and campaign evolution. We use a more conservative idea to connect different campaigns with the same IOCs than the method in [43] . If two campaigns contain identical IOCs and appear within a short time frame of another, then we consider that they are in fact the same campaign. For example, if one domain is mentioned by two articles in May 2012 and August 2012, then we group

Figure 4: Screenshot of annotation website.

the campaigns from two articles. In this study, we choose the time window to be 6 months, which means that the campaign groups are merged only if they share common IOCs and are discussed within 6 months of one another.

4. Insights into malware delivery campaigns

In this section, we first discuss the performance of ChainSmith and how to link the results to the real-world telemetry data. Then we study campaigns from 3 directions: baiting techniques (Section 4.2), business relationship between campaign groups (Section 4.3) and implication to threat intelligence (Section 4.4)

4.1. ChainSmith performance

A key challenge for semantics extraction is the lack of a ground truth. While IOCs are available on several threat intelligence platforms, the malware delivery phase that these IOCs correspond to are usually unavailable. To train and evaluate ChainSmith, we built a web application for manually annotating articles, as shown in Figure 4. The annotators can first specify which campaign stage the IOC belongs to and then enter the IOC name in the box. This tool allows us to collect the ground truth for our system. 4 graduate students from our group participated in the annotation exercise. To increase their chances of extracting meaningful information about malware delivery campaigns, we presented only articles that contain at least 20 named entities for labeling. In this way, we annotate 153 articles and 6264 IOCs.

Even with the ground truth collected from manual annotation, off-the-shelf NLP techniques are inadequate for characterizing malware delivery campaigns. We design a baseline system that models the campaign stage classification using traditional topic modeling. We consider each sentence as an individual document, and train a topic model using Latent Dirichlet allocation (LDA) [21]. LDA is an unsupervised algorithm widely used for topic modeling. For example, the Toronto Paper Matching System (TPMS) [45] employs LDA to model the reviewers' research areas and the areas of papers under submission. Additionally, a growing

TABLE 4: Performance comparison of ChainSmith and a baseline NLP system, which utilizes Latent Dirichlet Allocation.

IOC detection		
	Precision	Recall
ChainSmith	91.9%	97.8%
Baseline	78.2%	66.7%
Stage classification		
	Precision_avg	Recall_avg
ChainSmith	78.2%	69.6%– 80.7%
Baseline	37.0%	28.7%

TABLE 5: Summary of extracted IOCs.

Stage	**Attribute**	**Count**
baiting	`server_URL`	4,874
	`server_IP`	1,001
exploitation	`exploit_site_URL`	3,735
	`exploit_site_IP`	732
	`exploited_vulnerability`	554
	`exploit_kit`	572
installation	`malware_hash`	4,263
	`malware_family`	1,604
C&C	`C&C_server_URL`	2,161
	`C&C_server_IP`	1,271
unknown		3,879
Total		24,653

number of conferences utilize TPMS for making automated reviewer assignments. After training in this manner, we identify the topic for each stage using the training set. If a sentence belongs to one malware campaign stage, then we extract all the named entities and label them as the attribute of that stage.

We use 5-fold cross-validation to evaluate the performance of ChainSmith and of our baseline system. Table 4 shows the results. ChainSmith achieves 91.9% precision and 97.8% recall, which is 13.7% and 31.1% higher than the baseline model, respectively. Since the campaign stage classification is a multi-class problem, we take the average precision and recall for all classes. Out of the detected IOCs, we are able to classify 86.2% of them into a campaign stage with 78.2% precision and 80.7% recall, which is more than twice as high as the baseline model. [2] ChainSmith outperforms the baseline because it is able to better determine the sentence context and the informative words.

We run ChainSmith on all 14,155 articles we collected, and discover 24,653 IOCs. Table 5 shows the summary for each type of IOCs. In addition, from the extracted IOCs, we further identify 8,902 campaign groups mentioning either suppliers or customers. We use one Sun Fire X2200 M2 server with 8 logical processors and 8GB RAM for this experiment. Extracting IOCs from one security article requires 0.214 seconds on average for each article.

To assess the need for ChainSmith, we collect 568,348 fully qualified domains and 297,218 IPv4 addresses from

2. Because ChainSmith is able to detect but fails to categorize part of IOCs, we cannot precisely calculate the recall . For a fair comparison to the baseline model, we calculate the lower bound by multiplying the classification rate (86.2%) to the recall (80.7%).

Hailataxii [8], a repository providing threat intelligence feeds in the STIX format [13]. [3] Compared to the IOCs collected from ChainSmith, only 60 overlapping domains and 26 overlapping IP addresses are found in both data sets, and Jaccard index is $1.0 * 10^{-4}$ and $8.7 * 10^{-5}$ for domain and IP address respectively. This illustrates the fact that most of the indicators discussed in the threat intelligence reports are not made available in a machine-readable format. Moreover, none of the IOCs from Hailataxii specifies the `kill_chain_phase` attribute, which would preclude the analysis we conduct in the rest of this section from being performed on the dataset.

Next, we link the campaign groups extracted from blog posts to Kwon *et al.*'s data set of download events [39]. This data set is based on Symantec's WINE platform, which provides security telemetry from real-world hosts. Each download event in the data set specifies the file hash of the downloader and its payload (the downloaded file), and optionally the URL or IP from which the payload was downloaded. In total, the telemetry data records 50.5M download events from 5M real-world users. Using the download event data, we check if IOCs for each campaign group are used as the downloader or downloading portal. We are able to find the subsequent download events for 59 groups, 37 of which produce high-risk downloads in the measurement data. We label all the suspicious download events based on the campaign IOCs. Consequently, we further discover 224 suspicious downloaders and 3,395 suspicious payloads.

Data release. We plan to release the ChainSmith system in the form of a Web application at https://ioc-chainsmith.org. We set up a web crawler for collecting articles each week. Then ChainSmith parses the articles and updates the database with new results. The website provides the entire data for download as well as a search interface to access the data.

4.2. Persuasion techniques

To entice users to download the payloads, attackers are creative in designing persuasion methods. Email spam, compromised websites, and malvertising can be used as the basic approaches to targeting large group of individuals. In addition, the attacker can post deceptive advertisements to persuade users to click on malicious link. By integrating human discovery from blog posts and the real-world download data from WINE, we aim to study different persuasion strategies as well as their impact on subsequent download behaviors.

In Section 4, we identify 59 campaign groups that are both reported by security analysts and WINE. Then we manually label the campaign groups based on the platform where the information is displayed and the trigger that initiated the download, e.g. the message that lures users to click or exploit kit that exploits a machine directly. As a result, we are able to label baiting techniques for 44 campaign groups.[4]

Table 6 shows the top campaign groups that drop the most high-risk binaries in WINE. To study the timeline of campaign groups, we further define *active time* to be the time span when high-risk downloads are observed in measurement data, and *discovery time* as the time period that the corresponding blogs are posted.

26 campaign groups start from social engineering, which is the most commonly used technique in our data. Unlike drive-by downloads, social engineering does not involve exploitation and intrusion, and malware delivery is initiated by user actions. Table 6 shows that all the top campaign groups entice users simply through social engineering, which suggests that social engineering is prevalent and effective.

To dig deeper into the social engineering methods, we group the high-risk binary payloads according to the persuasion technique utilized, and we calculate the daily rate of high-risk downloads produced by each technique. Figure 5 shows the daily high-risk downloads for the top 10 baiting techniques. For a fair comparison among the social engineering methods, we remove drive-by downloads that deliver payloads directly using exploits. Figure 5a shows the total daily downloads of high-risk binaries. Media player related advertisement involves providing deceptive information that entices users to download something in order to watch an online video. For example, group 1 in Table 6 provides a fake video, and asks users to download the missing plug-in in order to play it. Most high-risk download events occur in response to media player advertising. In fact, this ruse can be substantially more persuasive than other techniques: groups 1 and 3 from Table 6 generate an order of magnitude more downloads than other groups.

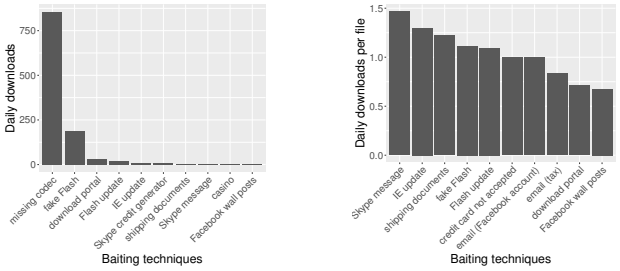
Figure 5b shows the daily downloads per file, for each persuasion technique, which reflects the delivery ability from the perspective of customers. The most prolific campaign group sends messages containing the download URL to the contacts of compromised Skype accounts. The effectiveness of this technique in distributing high-risk files suggests that users are less alert to the message received from their friends and colleagues. However, we do not observe a large volume of downloads for these campaigns, which are limited by the number of compromised accounts. Another effective baiting trick is to present a fake software update notification, and the common target applications are Flash Player and IE. To enhance credibility, the page hosting the supposed plug-in can even mimic the Flash Player installation process; instead, a dropper is usually installed on the user's machine. The least attractive baiting content is anti-virus scanners. These anti-virus scanners are usually labeled as PUP (potentially unwanted program) since they do not behave as they claimed and ask users to buy the license. The low rate of downloads suggests that users are less susceptible to the rogue anti-virus advertisement,

3. The data is collected by 2016-12-15.

4. Some blog posts focus more on the payload and only report the fact where the payload is dropped, without describing the download infrastructure. In this case, we are not able to collect the baiting techniques.

TABLE 6: Top 10 campaign groups that drop the most suspicious binaries. The group name is identified from the information provided by the references. Active time corresponds to the time span of suspicious downloads, and discovery time corresponds to the date when the references are posted. hrd: high-risk download, hrb: high-risk binaries.

Rank	Name	# of hrb.	# of hrd.	Active time	Discovery time	Persuasion technique
1	Pinball Corp	874	25,578	05/19/2010 - 10/06/2013	03/29/2011	Advertise missing codec.
2	InstallCore	281	502	03/19/2012 - 02/07/2014	06/24/2013 - 10/18/2013	Advertise missing codec.
3	YieldManager	167	29,756	10/06/2012 - 06/28/2014	12/20/2012 - 07/03/2013	Advertise Flash Player.
4	Somoto installer	137	4,703	04/16/2012 - 06/27/2014	05/15/2013 - 07/26/2013	Advertise missing codec.
5	EzDownloaderpro	29	119	07/06/2012 - 09/30/2013	10/22/2013	Download portal.
6	OpenCandy	18	497	07/29/2013 - 06/15/2014	05/02/2014	Download portal.
7	Clikug	16	450	11/20/2013 - 06/30/2014	12/04/2013- 02/13/2014	Advertise Skype credit generator.
8	Awimba LLC	14	144	05/28/2012 - 05/23/2013	06/19/2013	Flash update notification.
9	BubbleDock	10	58	08/08/2013 - 01/05/2014	11/11/2013	Advertise missing codec.
10	–	6	88	02/04/2013 - 10/27/2013	03/02/2013	Flash update notification.



(a) Total downloads (b) Average downloads

Figure 5: Daily downloads of high-risk binaries. Total daily download (left) reflects the frequency that a specific baiting technique is used in high-risk download; while the average of individual file (right) reflects the effectiveness of binary delivery of a single file.

which is also supported by prior work [50]. Interestingly, an application that warns about an impending zombie invasion produces a higher download rate than the rogue anti-virus scanners.

We do not include drive-by downloads in our analysis, as the total number of downloads is too small to accurately estimate the influence and effectiveness of the technique. This suggests that attackers likely change the domains used for exploitation and intrusion frequently so that the IOCs collected from blog posts cannot be used to capture the download behaviors. The binary download behaviors from three exploit kits are recorded in WINE. However, the total number of high-risk downloads is less than 10 for each of groups, which suggests that the domains can change rapidly for the exploitation and intrusion campaigns.

Table 6 shows the dates when campaigns from the top tier-1 groups were active. This shows that these campaigns are long lasting, usually exceeding 1 year in duration. Because the download event data-set was collected until June 2014, we cannot track the full length of some campaigns persisting past that date. The campaigns for half of the top tier-1 groups remain active at that date, suggesting that the real duration may be significantly longer. These long durations suggest that we currently lack the technical means to stop the delivery campaigns involved in social engineering. Interestingly, for 7 out of the 10 campaigns, the articles mentioning them are published well within the activity period of the campaigns, suggesting that the campaigns go on after they are discovered by security analysts. This suggests that the baiting techniques remain effective in luring users, and the benefits of switching to another baiting technique do not outweigh the costs.

Implications. Comparing the effectiveness of the persuasion techniques employed by malicious actors suggests areas where public awareness and education are most likely to have an impact on malware delivery. In addition, the campaigns from tier-1 suppliers usually go on for at least 1 year—even after they have been discovered. This reveals the limitations of existing security tools in preventing campaigns that rely upon social engineering. Because most tier-1 suppliers employing social engineering do not exhibiting any exploitation or intrusion intentions, and because the downloaded payload may not expose obvious malicious behavior, it is difficult to detect and determine whether to block such campaigns in the gray area.

4.3. Underground business relationships

To study the business relationship between tier-1 suppliers and their customers, we identify the ownership of payloads from the certificate and the anti-virus signatures. If the payload is signed, which is often the case for PUPs, then we consider the publisher in the certificate as the owner of the payload. If the payload is not signed, then we identify the malware family using the AVclass tool [58]. As a result, we identify 289 payload owners. The payload owners maintain a direct relationship with tier-1 suppliers if the payload is dropped from the domains of tier-1 suppliers. Otherwise, we consider the relationship as indirect if the binary is bundled by the reseller without establishing direct relationship with tier-1 suppliers. 87.5% of payload owners maintain a direct relationship with the tier-1 suppliers, which suggests that the majority of downloads are directly associated with the baiting techniques.[5] In addition, one actor might have different ways to deliver its payloads. For example, Somoto

5. The actual percentage of direct should be higher, because the payload owner and tier-1 supplier can be the same. In this case, the customer of the payload dropper also has a direct relationship with tier-1 suppliers.

better installer has its own advertising network [3], but it is also delivered through OpenCandy in our data set.

Some business relationships might be hidden from the measurement data because the real creator of a binary may not be the parent process. For example, if the application creates a service, then `svchost.exe` might be recorded as the parent of any subsequent behaviors from the service. Therefore, the parent-child relationship might be incomplete from the measurement data. However, blog posts provide another complementary data source to bridge the missing relations because security analysts are able to differentiate and report the real causal relationships. By integrating the results from security articles, we are able to formulate hypotheses about the business relationship between campaign groups. For example, tpstneuknash[dot]com is reported as the C&C server for ZeroAccess, and it delivers TrojanSpy:Win32/Bafi.E in WINE. Therefore, it is likely that the victim of ZeroAccess receives the command to download Win32/Bafi from tpstneuknash[dot]com. Moreover, such business relationship might be previously unknown, since Win32/Bafi is absent from malware families that are dropped by ZeroAccess [4]. Moreover, Symantec security response lists two malware families dropped by ZeroAccess, and the payload malware we discovered is not on the list, which implies that the business relationship might be previously unknown.

Implications. Although not all campaign groups have infrastructure for baiting, most of the groups are the direct customer of tier-1 suppliers. In addition, the tier-1 suppliers can also be the customer of another group. This suggests that the business relationship is prevalent and the actors in the underground market can be highly connected. Moreover, there might be more business relationships hidden from measurement data due to the fact that it is hard to find the real parent of a download event.

4.4. Lifecycles of campaigns and infrastructures

As noted in Section 4.2, most of campaign groups use social engineering for binary delivery. This suggests that most of the domains used in exploitation and intrusion may change frequently, and consequently it is difficult to use IOCs for detecting long-running campaigns. Figure 6 shows the distribution of IOC occurrence in technical blogs for file hash, URL and IP address.[6] Owing to malware polymorphism and domain generating algorithms, most domains and malware will not be identical during the same campaign and therefore security analysts are likely to report different IOCs from the same campaign. In addition, IP addresses are more likely to be discussed in different articles, because they are more difficult to be changed than URLs and file hashes. This suggests that attackers are likely to change the domains they used in order to evade detection, and therefore IOCs are usually short-lived and may never be used again.

6. We use fully qualified domain for counting URL, which ignores the associated path name.

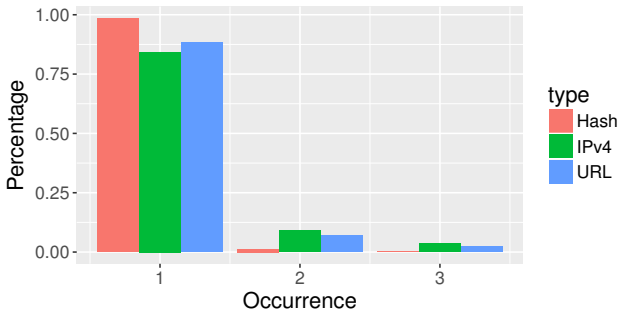


Figure 6: Percentage of the IOC occurrence in technical blogs for file hash, URL and IP address. The distribution of IOC occurrence is highly skewed, and therefore we only show the percentage of the occurrence less than 3.

While their utility for real-time detection is limited, IOCs can be useful in security forensics to identify long-term campaign groups. Domains can be changed frequently, but the attackers are still likely to reuse part of the infrastructure, which makes it possible to connect the dots and observe the long-term evolution of a campaign. Figure 7 shows examples of spam campaigns related to the Cridex family collected from 6 different articles. All the spam emails redirect users to the BlackHole exploit kit, and drop one version of Cridex malware. The malware communicates with different command and control servers, and the IP of the C&C server keeps changing over time, which is likely because the server's IP gets blacklisted. But in some cases, the attacker reuses old C&C IPs. For example, as shown in Figure 7, attacker reused 210.56.23.100 on November 19 (4 months later), and reused 95.142.167.193 on November 26 (3 months later). One possible explanation for this behavior is that IPs are removed from blacklists after several months so that the attackers may add them back to the campaign infrastructure. Kührer *et al.* studied 15 blacklists from which they estimated the average blacklisting time [38]. Although the average listing time varies for different blacklists, the malicious domain will be delisted after 100 days on average. This suggests that attackers might actively check if the domain and IP are removed from blacklist, and utilize the old delisted domain and IP.

Implications. Our findings provide important lessons for the most effective uses of threat intelligence. The premise of threat intelligence is that sharing the technical details of breaches and attacks makes everybody safer because it makes it harder for attackers to reuse attack methods and artifacts, thus increasing their *work factor* [57]. Our results suggest that the detection of IOCs based on URLs and IPs is not an effective mitigation, as the network identities of servers involved in malware delivery campaigns already change frequently. While several domain generation algorithms have been reverse engineered in prior work [55], future IP and domain name changes within one campaign cannot, in general, be predicted based only on the threat intelligence. Even though the URLs and IP addresses of

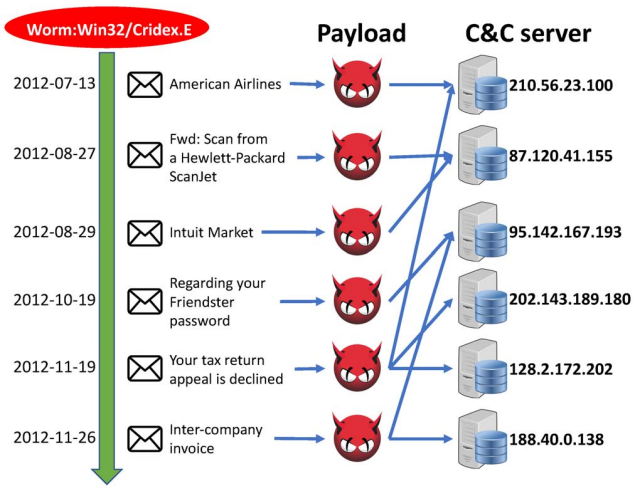


Figure 7: Spam campaigns that drop Worm:Win32/Cridex.E.

malicious servers are short-lived, campaigns using this infrastructure can be long-lasting and they may continue after some components are taken down. This suggests that the connections established among IOCs and the qualitative insights about the campaigns are the most useful outcomes of threat intelligence.

5. Discussion

Persuasion techniques. In Section 4, we collect the IOCs for tier-1 suppliers from technical blogs and further link them to the download telemetry in WINE. Most of tier-1 groups entice users to download payload through social engineering, and the subsequent high-risk downloads can continue for more than one year. This suggests that current security tools have important limitations in stopping the campaigns using social engineering tricks. The fact that both the downloading portal and the dropped payloads do not exhibit obvious malicious behaviors makes it difficult for anti-virus vendors to detect, or even determine, if the download should be blocked. Consequently, the campaign is usually long-lasting, and the high-risk download behavior can continue for more than one year. In addition, regardless of whether the high-risk file is PUP or malware, most of the campaigns discussed by technical blogs have already exhibited suspicious intentions from the way they present the advertisement to the victims. The message from the advertisement is usually misleading and deceptive, for example the fake software download and fake update notification.

Campaign detection. As social engineering is the prevalent strategy to start the delivery campaign, human factors play an important role in the security arms race. The state-of-the-art detection systems focus more on blocking network intrusion and removing malicious programs, but they usually ignore whether the behaviors are consistent with what is as expected. In many cases, especially for PUP delivery, the message provided in the campaign is inconsistent with the behavior behind the scenes. This provides an opportunity for stopping these campaigns by blocking the misleading advertisement, regardless of whether the downloaded payload is malicious or not. For example, a benign download should be blocked when it is bundled with a fake Flash update notification. In this case, understanding the semantics of advertisement is the key to preventing the deception.

Threat intelligence. In this paper, we compare the IOCs collected from blog posts to the real-world measurement data. We show that threat intelligence is useful in security forensics, because it provides semantics to the measurement data and make it possible for security researchers to explain the measurement data. However, measuring the effectiveness of IOCs in detecting malicious campaigns is equally important and has not been studied thoroughly. In fact, that few IOCs are shared in both measurement data and security articles suggests that the infrastructure in the campaign is likely to change frequently. Therefore, how to make the most use of threat intelligence in real-time campaign detection should be an important next step.

Automatically extracting the semantics of security threats from natural language documents is a promising direction for the analysis of long-lasting campaigns. In particular, determining semantics of the *relationships among indicators of compromise* is key to reconstructing chains of actions and distinguish different actors in the campaign. Prior techniques for extracting IOCs from technical documents [43], [9], [5] are unable to reconstruct campaigns automatically (in [43], the authors established some links between a C&C infrastructure and the exploits utilized through manual analysis). In contrast, ChainSmith automatically reconstructs the semantics of entire delivery campaigns. One benefit of semantic relationships is that they allow us to identify different actors in campaigns. To stimulate further research on cyber threat intelligence, we released a Web application (https://ioc-chainsmith.org) for providing data extracted automatically from natural language reports in a timely manner.

6. Related work

Malware campaign. Most prior measurement studies focused on only one stage of malware delivery campaigns. Prior work has identified social media advertising [64], [23], [50], spam email [20], [31], compromised site [44], or SEO poisoning [42] as techniques used for delivering payloads. However, most of the work did not quantify the volume of malware downloads resulting from these techniques. Nelms *et al.* [50] measure the influence of different persuasion techniques on malware downloads quantitatively by parsing HTTP response and manually annotating the content. However, this method is not scalable from both manual annotation and network monitoring. In addition, this method fails if the package is encrypted using HTTPS. The advantage of our technique is that there are less restrictions on measurement data, because the campaign information is collected from independent sources.

Eshete *et al.* propose a system to detect if the URL is the landing page of exploit kit based on the common techniques and behaviors of exploit kit. [27]. Moreover, Eshete *et al.* design a exploit kit infiltration toolkit to detect the exploited vulnerabilities [26].

The malicious downloading behavior was analyzed by Kwon *et al.* in [39], [40]. In [39], a classifier is designed to detect malware using downloader-payload relationships. In [40], the authors propose a system to detect coordinated behaviors among downloaders on multiple hosts. PUP delivery services were studied in [37], [61]. The former identifies the PUP publisher and the delivery service from the certificate and structure of download relationships, while the latter closely investigates a few prevalent PPI services and monitors the subsequent delivery behavior.

Zhang *et al.* propose a system to detect malicious servers and group them into campaigns using HTTP traces from ISPs [63]. Plohmann *et al.* comprehensively analyze domain generating algorithms and pre-compute the DGA domains in [55].

Threat intelligence. IOCs (indicators of compromise) are commonly used to model malware campaigns. The IOCs specified in OpenIOC, STIX and CybOX standards [11], [13], [14] define the role of an indicator in a campaign, for instance that an IP address corresponds to a C&C server. Researchers at Lockheed-Martin corporation proposed a cyber kill chain model to describe the action sequence from attacker to launch malware campaign [33]. STIX includes the cyber kill chain definition, however this is seldom used in existing IOC feeds. For example, Hailataxii [8] is a repository for threat intelligence data formatted according to the STIX standard. Although more than 700,000 indicators are provided by Hailataxii, none of them include the kill chain phase, which can only be inferred manually from natural language description of indicator.

In [43], Liao and Yuan *et al.* propose a system to automatically collect IOC from blog posts in natural language. They model the problem as graph similarity problem, and identify the IOC item if it has a similar graph structure as the training set. However, the identified IOCs do not preserve their roles in a malicious campaign, which makes it difficult to analyze the characteristics of campaign in different stages and to correlate with field measurements.

NLP in security. Few references in security utilize natural language processing in system design. Neuhaus *et al.* analyze the trend of vulnerability by applying LDA to vulnerability description [51]. Pandita *et al.* identify Android permissions that are implicitly stated in the app description by using a dependency parser and first order logic [52]. Zhu *et al.* propose a framework to represent the knowledge available in security literature and generate features for detecting malware [65]. Liao and Yuan *et al.* propose a system to automatically extract OpenIOC items from blog posts [43]. Sun *et al.* design a system to generate human-friendly report for the results from Cuckoo sandbox [60]. Panwar designs a framework to generate IOCs in STIX format from Cuckoo sandbox results [53]. A concurrent work by Husari *et al.* convert the Symantec malware report to STIX format by utilizing a pre-defined ontology [32]. In terms of the NLP techniques, we use word embedding instead of manually defined rules to learn the semantics of sentences, which is more general and applicable to a broader area. In addition, most of the work only focus on how to apply NLP to security but do not show security implications behind it.

7. Conclusions

We describe ChainSmith, a system that automatically extracts IOCs from technical articles and industry reports, and classifies them into different campaign stages, i.e. baiting, exploitation, installation and command and control. This provides a semantic layer on top of IOCs that captures the role of indicators in a malicious campaign. ChainSmith achieves 91.9% precision and 97.8% recall in extracting IOCs from natural language documents and is able to determine the campaign stage for 86.2% of IOCs with 78.2% precision and 80.7% recall. In addition, we identify 8,902 campaign groups from IOCs based on the stages and article post time. This makes it possible to combine threat intelligence with field-gathered data. The data is released at https://ioc-chainsmith.org.

We use a data set of download events to measure the effectiveness of different persuasion strategies employed in the baiting stage. We find that most campaigns deliver payloads through social engineering, without exhibiting any intrusion intentions. Media player advertising is most persuasive and generates the largest number of high-risk downloads, but "friends recommendations" and fake update notifications are most effective, as they generate the most daily downloads per file. In addition, we find that most of the customers in the malware delivery ecosystem have direct relationships to the suppliers that operate baiting services. Finally, we give use cases for leveraging IOCs in security forensics, which sheds new light on the best uses of threat intelligence.

References

[1] "STIX support survey," OASIS Cyber Threat Intelligence (CTI) TC Wiki. https://wiki.oasis-open.org/cti/Products.

[2] "Overview of exploit kit," http://contagiodump.blogspot.com/2010/06/overview-of-exploit-packs-update.html, 2010.

[3] "Somoto baiting technique," https://www.webroot.com/blog/2013/07/03/deceptive-ads-targeting-german-users-lead-to-the-w32somotobetterinstaller-potentially-unwanted-application-pua/, 2013.

[4] "Symantec security response: Zeroaccess," https://www.symantec.com/security_response/writeup.jsp?docid=2011-071314-0410-99&tabid=2, 2013.

[5] "Alienvault otx," https://otx.alienvault.com, 2017.

[6] "Cyber security / information assurance (CS/IA) program," http://dibnet.dod.mil/, 2017.

[7] "cyberscoop: Dhs slammed for report on russian hackers," https://www.cyberscoop.com/dhs-election-hacking-grizzly-steppe-iocs/, 2017.

[8] "Hail a TAXII: a repository of open source cyber threat intellegence feeds in STIX format," https://http://hailataxii.com, 2017.

[9] "Ioc parser," https://github.com/armbues/ioc_parser, 2017.

[10] "Malware naming convention," https://www.microsoft.com/en-us/security/portal/mmpc/shared/malwarenaming.aspx, 2017.

[11] "The openioc framework," http://www.openioc.org/, 2017.

[12] "Reserved ip addresses," https://en.wikipedia.org/wiki/Reserved_IP_addresses, 2017.

[13] "STIX: Structured threat information expression," http://stixproject.github.io, 2017.

[14] "A structured language for cyber observables," https://cyboxproject.github.io, 2017.

[15] "ThreatConnect," https://www.threatconnect.com/, 2017.

[16] "Top level domains," http://data.iana.org/TLD/tlds-alpha-by-domain.txt, 2017.

[17] "Virus total," www.virustotal.com, 2017.

[18] "Webroot: Your discover card services blockaded themed emails serve client-side exploits and malware," https://www.webroot.com/blog/2012/11/08/your-discover-card-services-blockaded-themed-emails-serve-client-side-exploits-and-malware/, 2017.

[19] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[20] E. Blanzieri and A. Bryl, "A survey of learning-based techniques of email spam filtering," *Artificial Intelligence Review*, vol. 29, no. 1, pp. 63–92, 2008.

[21] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[22] J. Caballero, C. Grier, C. Kreibich, and V. Paxson, "Measuring pay-per-install: The commoditization of malware distribution." in *USENIX Security Symposium*, 2011.

[23] C. Cao and J. Caverlee, "Detecting spam urls in social media via behavioral analysis," in *European Conference on Information Retrieval*. Springer, 2015, pp. 703–714.

[24] M.-C. De Marneffe and C. D. Manning, "Stanford typed dependencies manual," Technical report, Stanford University, Tech. Rep., 2008.

[25] T. Dumitraș and D. Shou, "Toward a standard benchmark for computer security research: The Worldwide Intelligence Network Environment (WINE)," in *EuroSys BADGERS Workshop*, Salzburg, Austria, Apr 2011.

[26] B. Eshete, A. Alhuzali, M. Monshizadeh, P. A. Porras, V. N. Venkatakrishnan, and V. Yegneswaran, "Ekhunter: A counter-offensive toolkit for exploit kit infiltration." in *NDSS*, 2015.

[27] B. Eshete and V. Venkatakrishnan, "Webwinnow: Leveraging exploit kit workflows to detect malicious urls," in *Proceedings of the 4th ACM conference on Data and application security and privacy*. ACM, 2014, pp. 305–312.

[28] Facebook, "ThreatExchange," https://threatexchange.fb.com/, 2017.

[29] C. Gates and C. Taylor, "Challenging the anomaly detection paradigm: a provocative discussion," in *Proceedings of the 2006 workshop on New security paradigms*. ACM, 2006, pp. 21–29.

[30] C. Grier, L. Ballard, J. Caballero, N. Chachra, C. J. Dietrich, K. Levchenko, P. Mavrommatis, D. McCoy, A. Nappa, A. Pitsillidis, N. Provos, M. Z. Rafique, M. A. Rajab, C. Rossow, K. Thomas, V. Paxson, S. Savage, and G. M. Voelker, "Manufacturing compromise: the emergence of exploit-as-a-service," in *ACM Conference on Computer and Communications Security*, Raleigh, NC, Oct 2012.

[31] G. Ho, A. Sharma, M. Javed, V. Paxson, and D. Wagner, "Detecting credential spearphishing in enterprise settings," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 469–485. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/ho

[32] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources," in *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, 2017, pp. 103–115.

[33] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, p. 80, 2011.

[34] International Data Corporation, "Worldwide threat intelligence security services 2014–2018 forecast: "iterative intelligence" — threat intelligence comes of age," IDC Market Forecast, Mar 2014.

[35] L. Invernizzi, S.-J. Lee, S. Miskovic, M. Mellia, R. Torres, C. Kruegel, S. Saha, and G. Vigna, "Nazca: Detecting malware distribution in large-scale networks," in *NDSS*, 2014.

[36] P. Kotzias, L. Bilge, and J. Caballero, "Measuring PUP prevalence and PUP distribution through Pay-Per-Install services," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 739–756. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kotzias

[37] P. Kotzias, S. Matic, R. Rivera, and J. Caballero, "Certified PUP: Abuse in Authenticode code signing," in *CCS*, 2015. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813665

[38] M. Kührer, C. Rossow, and T. Holz, "Paint it black: Evaluating the effectiveness of malware blacklists," in *RAID*, 2014. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11379-1_1

[39] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitraș, "The dropper effect: Insights into malware distribution with downloader graph analytics," in *CCS*, 2015.

[40] B. Kwon, V. Srinivas, A. Deshpande, and T. Dumitraș, "Catching worms, trojan horses and PUPs: Unsupervised detection of silent delivery campaigns," in *Network and Distributed System Security (NDSS) Symposium*, San Diego, CA, Feb 2017.

[41] O. Levy and Y. Goldberg, "Dependency-based word embeddings." in *ACL (2)*. Citeseer, 2014, pp. 302–308.

[42] X. Liao, C. Liu, D. McCoy, E. Shi, S. Hao, and R. Beyah, "Characterizing long-tail seo spam on cloud web hosting services," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 321–332.

[43] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah, "Acing the IOC game: Toward automatic discovery and analysis of open-source cyber threat intelligence," in *ACM Conference on Computer and Communications Security*, Vienna, Austria, 2016.

[44] X. Liao, K. Yuan, X. Wang, Z. Pei, H. Yang, J. Chen, H. Duan, K. Du, E. Alowaisheq, S. Alrwais *et al.*, "Seeking nonsense, looking for trouble: Efficient promotional-infection detection through semantic inconsistency search," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 707–723.

[45] X. Liu, T. Suel, and N. Memon, "A robust model for paper reviewer assignment," in *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 2014, pp. 25–32.

[46] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010

[47] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[48] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[49] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad, "Webwitness: Investigating, categorizing, and mitigating malware download paths," in *USENIX Security Symposium*, 2015. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/nelms

[50] ——, "Towards measuring and mitigating social engineering software download attacks," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 773–789. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/nelms

[51] S. Neuhaus and T. Zimmermann, "Security trend analysis with CVE topic models," in *IEEE 21st International Symposium on Software Reliability Engineering, ISSRE 2010, San Jose, CA, USA, 1-4 November 2010*, 2010, pp. 111–120. [Online]. Available: http://dx.doi.org/10.1109/ISSRE.2010.53

[52] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "WHYPER: towards automating risk assessment of mobile applications," in *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, S. T. King, Ed. USENIX Association, 2013, pp. 527–542. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/pandita

[53] A. Panwar, "igen: Toward automatic generation and analysis of indicators of compromise (iocs) using convolutional neural network," Ph.D. dissertation, Arizona State University, 2017.

[54] E. E. Papalexakis, T. Dumitras, D. H. P. Chau, B. A. Prakash, and C. Faloutsos, "Spatio-temporal mining of software adoption & penetration," in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Niagara Falls, CA, Aug 2103.

[55] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 263–278. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/plohmann

[56] B. Rahbarinia, R. Perdisci, and M. Antonakakis, "Segugio: Efficient behavior-based tracking of malware-control domains in large ISP networks," in *DSN*, 2015. [Online]. Available: http://dx.doi.org/10.1109/DSN.2015.35

[57] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975. [Online]. Available: http://dx.doi.org/10.1109/PROC.1975.9939

[58] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "Avclass: A tool for massive malware labeling," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2016, pp. 230–253.

[59] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *IEEE Symposium on Security and Privacy*, 2010, pp. 305–316.

[60] B. Sun, A. Fujino, and T. Mori, "Poster: Toward automating the generation of malware analysis reports using the sandbox logs," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1814–1816.

[61] K. Thomas, J. A. E. Crespo, R. Rasti, J.-M. Picod, C. Phillips, M.-A. Decoste, C. Sharp, F. Tirelo, A. Tofigh, M.-A. Courteau, L. Ballard, R. Shield, N. Jagpal, M. A. Rajab, P. Mavrommatis, N. Provos, E. Bursztein, and D. McCoy, "Investigating commercial pay-per-install and the distribution of unwanted software," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 721–739. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/thomas

[62] K. Thomas, D. Huang, D. Wang, E. Bursztein, C. Grier, T. J. Holt, C. Kruegel, D. McCoy, S. Savage, and G. Vigna, "Framing dependencies introduced by underground commoditization," in *WEIS*, 2015.

[63] J. Zhang, S. Saha, G. Gu, S.-J. Lee, and M. Mellia, "Systematic mining of associated server herds for malware campaign discovery," in *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*. IEEE, 2015, pp. 630–641.

[64] X. Zhang, Z. Li, S. Zhu, and W. Liang, "Detecting spam and promoting campaigns in twitter," *ACM Transactions on the Web (TWEB)*, vol. 10, no. 1, p. 4, 2016.

[65] Z. Zhu and T. Dumitras, "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*. ACM, 2016, pp. 767–778.