Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence

Xiaojing Liao[1]*, Kan Yuan[2]*, XiaoFeng Wang[2], Zhou Li[3], Luyi Xing[2], Raheem Beyah[1]
[1]Georgia Institute of Technology, [2]Indiana University Bloomington, [3]ACM member
{xliao, rbeyah}@gatech.edu, {kanyuan, xw7, luyixing}@indiana.edu, lzcarl@gmail.com

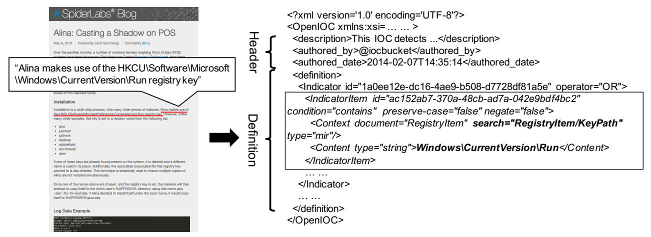
ABSTRACT

To adapt to the rapidly evolving landscape of cyber threats, sec rity professionals are actively exchanging *Indicators of Compr mise* (IOC) (e.g., malware signatures, botnet IPs) through publ sources (e.g. blogs, forums, tweets, etc.). Such information, c ten presented in articles, posts, white papers etc., can be convert into a machine-readable OpenIOC format for automatic analys and quick deployment to various security mechanisms like an i trusion detection system. With hundreds of thousands of sourc in the wild, the IOC data are produced at a high volume and velo ity today, which becomes increasingly hard to manage by humar Efforts to automatically gather such information from unstructur text, however, is impeded by the limitations of today's Natural La guage Processing (NLP) techniques, which cannot meet the hi standard (in terms of accuracy and coverage) expected from the IOCs that could serve as direct input to a defense system.

In this paper, we present *iACE*, an innovation solution for fully automated IOC extraction. Our approach is based on the observation that the IOCs in technical articles are often described in a predictable way: being connected to a set of context terms (e.g., "download") through stable grammatical relations. Leveraging this observation, iACE is designed to automatically locate a putative IOC token (e.g., a zip file) and its context (e.g., "malware", "download") within the sentences in a technical article, and further analyze their relations through a novel application of graph mining techniques. Once the grammatical connection between the tokens is found to be in line with the way that the IOC is commonly presented, these tokens are extracted to generate an OpenIOC item that describes not only the indicator (e.g., a malicious zip file) but also its context (e.g., download from an external source). Running on 71,000 articles collected from 45 leading technical blogs, this new approach demonstrates a remarkable performance: it generated 900K OpenIOC items with a precision of 95% and a coverage over 90%, which is way beyond what the state-of-the-art NLP technique and industry IOC tool can achieve, at a speed of thousands of articles per hour. Further, by correlating the IOCs mined from the articles published over a 13-year span, our study sheds new light on the links across hundreds of seemingly unrelated attack instances, particularly their shared infrastructure resources, as well as the impacts of such open-source threat intelligence on security protection and evolution of attack strategies.

[1]The two lead authors are ordered alphabetically.

CCS'16, October 24-28, 2016, Vienna, Austria
© 2016 ACM. ISBN 978-1-4503-4139-4/16/10…$15.00
DOI: http://dx.doi.org/10.1145/2976749.2978315


SpiderLabs® Blog
Alina: Casting a Shadow on POS
"Alina makes use of the HKCU\Software\Microsoft \Windows\CurrentVersion\Run registry key"
Header
Definition

```
<?xml version='1.0' encoding='UTF-8'?>
<OpenIOC xmlns:xsi= … … >
  <description>This IOC detects ...</description>
  <authored_by>@iocbucket</authored_by>
  <authored_date>2014-02-07T14:35:14</authored_date>
  <definition>
   <Indicator id="1a0ee12e-dc16-4ae9-b508-d7728df81a5e" operator="OR">
     <IndicatorItem id="ac152ab7-370a-48cb-ad7a-042e9bdf4bc2"
condition="contains" preserve-case="false" negate="false">
       <Context document="RegistryItem" search="RegistryItem/KeyPath"
type="mir"/>
       <Content type="string">Windows\CurrentVersion\Run</Content>
     </IndicatorItem>
     … …
   </Indicator>
   … …
  </definition>
</OpenIOC>
```

Figure 1: Example of OpenIOC schema.

1. INTRODUCTION

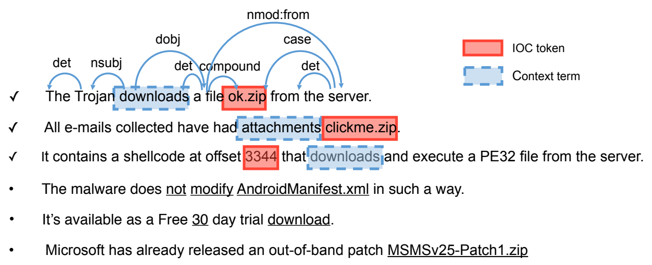
According to Gartner, *Cyber Threat Intelligence* (CTI) is defined as "evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice, about an existing or emerging menace or hazard to assets that can be used to inform decisions regarding the subject's response to that menace or hazard" [42]. Such knowledge is essential for an organization to gain visibility into the fast-evolving threat landscape, timely identify early signs of an attack and the adversary's strategies, tactics and techniques, and effectively contain the attack with proper means. Given its importance, CTI has been aggressively collected and increasingly exchanged across organizations, often in the form of *Indicators of Compromise* (IOC) [35], which are forensic artifacts of an intrusion such as virus signatures, IPs/domains of botnets, MD5 hashes of attack files, etc. Once collected, these IOCs can be automatically transformed and fed into various defense mechanisms (e.g., intrusion detection systems) when they are formatted in accordance with a threat information sharing standard, such as OpenIOC [9], that enables characterization of sophisticated attacks like drive-by downloads. The challenge, however, comes from the effective gathering of such information, which entails significant burdens for timely analyzing a large amount of data.

Finding IOCs online: challenges. While IOCs can be extracted from traditional blacklists, like CleanMX [22] and PhishTank [37], the information delivered by such IOCs is rather thin: only a small number of IOC classes are covered (URL, domain, IP and MD5), the relation between IOCs is not revealed and no context information is provided (e.g., the criminal group behind malfeasance). Analyzing the cyber-attack campaigns and triaging incident responses become quite difficult when relying on such information. Instead, IOCs from articles in technical blogs and posts in forums are more favorable to security practitioners and extensively harvested, since comprehensive descriptions of the attack are often found there. Such descriptions are typically informal, in natural languages, and need to be analyzed semantically to recover related attack indicators, before they can be converted into the standard IOC format such as OpenIOC, as illustrated in Figure 1. For years, this has been done manually by security analysts. Increasingly, however, the volume and velocity of the information generated from these sources become hard to manage by humans in a cost-effective way. As an example, in our research, we studied over 71,000 articles from 45 technical blogs extensively used by security professionals and found that the number of articles posted here has grown from merely a handful back 10 years ago to over 1,000 every month since last year (see Section 4.1). Note that these blogs are just a drop in the bucket: for example, Recorded Future is reported to utilize over 650,000 open web sources in 7 languages to harvest IOCs [41]. With the huge amount of information produced by those sources, new technologies are in dire need to *automate* the identification and extraction of valuable CTI involved.

Automatic collection of IOCs from natural-language texts is challenging. Simple approaches like finding IP, MD5 and other IOC-like strings in an article, as today's IOC providers (AlienVault, Recorded Future) do, does not work well in practice, which easily brings in false positives, mistaking non-IOCs for IOCs: e.g., as illustrated in Figure 2, although three zip files show up in attack-related articles, `MSMSv25-Patch1.zip` is clearly not an IOC while the other two are. Further, even for a confirmed IOC, we need to know its context, e.g., whether it is linked to drive-by download (`ok.zip` in the figure) or Phishing (`clickme.zip`), in order to convert it to the IOC format and help an organization determine its response. This can only be done by establishing a relation between the IOC token and other content in the article, such as the terms "downloads", "attachment" in the example.

Identifying semantic elements (called *Named Entity Recognition* or NER [33]) and extracting relations between them (called *Relation Extraction*, or RE [18]) have been extensively studied in the Natural Language Processing (NLP) community. However, existing NLP techniques *cannot* be directly applied for IOC and its context discovery. NER systems are known to be brittle, highly domain-specific — those designed for one domain hardly work well on the other domain [45]. So far, we are not aware of any NER techniques developed specifically for recognizing IOCs and a direct use of the state-of-the-art tools like Stanford NER [26] leads to low precision (around 70%) and recall (less than 50%) (see Section 4). Further, the current study on RE focuses on the relation between two known entities, which are typically nominals [24], whereas the relations between an IOC and the terms describing its context (which we call *context terms*) are way more complicated: e.g., "downloads" and `ok.zip` has a verb-noun relation. Also important, the accuracy and coverage offered by the existing RE tools are typically low. For example, the classic tree kernel approach [24] reports a precision between 50 and around 90% and a recall between 10 and 50%. As another example, a recent proposal for extracting events among genes and proteins from biomedical literature [45] has a precision and recall around 50%. This level of performance cannot meet the demand for high-quality IOCs, which could go straight to a security system to offer immediate protection for an organization.

iACE. A key observation from the open intelligence sources producing high-quality IOCs is that technical articles and posts tend to describe IOCs in a simple and straightforward manner, using a fixed set of context terms (e.g., "download", "attachment", "PE", "Registry", etc.), which are related to the *iocterms* used in OpenIOC to label the types of IOCs [9]. Further, the grammatical connections between such terms and their corresponding IOCs are also quite



✓ The Trojan downloads a file ok.zip from the server.

✓ All e-mails collected have had attachments clickme.zip.

✓ It contains a shellcode at offset 3344 that downloads and execute a PE32 file from the server.

• The malware does not modify AndroidManifest.xml in such a way.

• It's available as a Free 30 day trial download.

• Microsoft has already released an out-of-band patch MSMSv25-Patch1.zip

Figure 2: Examples of sentences with/without IOC.

stable: e.g., the verb "downloads" followed by the nouns "file" and `ok.zip` (the IOC) with a compound relation; "attachments" and `clickme.zip` also with the compound relation. This allows us to leverage such relations to identify an IOC and its context tokens, combining the NER and RE steps together. To this end, we developed a new approach, called *iACE* (IOC Automatic Extractor), in our research, which tailors NLP techniques to the unique features of IOC discovery. More specifically, after preprocessing articles (using *topic-term classification* to find those likely involving IOCs and converting figures to text), iACE utilizes a set of regular expressions (regex) and common context terms extracted from iocterms to locate the sentences within the articles that contain *putative IOC tokens*, such as IP, MD5-like string. Within each of such sentences, our approach attempts to establish a relation between the IOC token and the context term: it converts the sentence into a *Dependency Graph* (DG) to describe its grammatical structure and extracts the shortest paths linking each pair of a context token and the putative IOC token; over each path, a *graph mining* technique is applied to analyze the relation between the tokens, which cannot be handled by existing RE techniques, for the purpose of determining whether they indeed include an IOC and its context.

This simple approach leverages the known "anchors" (context and regex terms) to locate potential IOC-carrying sentences and the predictable relations among them to capture IOCs, avoiding the complexity of direct application of existing NLP techniques, which needs to solve the NER first to identify an IOC token before addressing the RE problem to find its context. Our evaluations on a prototype we built show that this new technique is very effective in practice: it achieved a precision of 95% at a coverage over 90%. In the meantime, our system is also highly efficient, capable of analyzing four articles per second even with a single process.

Our discoveries. Running on all 71,000 articles collected from the 45 blogs (including AlienVault, Malwarebytes, Webroot, etc.) in the past 13 years (from 2003/01 to 2016/04), iACE automatically extracted 900K IOC tokens together with their context. By inspecting these items and correlating them across different blogs over the long time frame, we were able to gain an unprecedented understanding of the relations between different attacks reported, the impact of open-source threat intelligence on attack evolutions, the defense responses it triggered, as well as the qualities of these blogs and effectiveness of the IOCs they document. More specifically, we found that some apparently unrelated attack instances were actually connected, sharing the same attack assets and infrastructures such as command and control (C&C) hosts. Particularly, by linking together 396 articles and more than 7,000 IOCs, our study reveals that a C&C campaign continued to evolve over a four-year span, changing the targets of exploits from one vulnerability (CVE-2010-1885) to another (CVE-2013-0422). Also interestingly, we observed that the attackers might adjust their strategies in response to the release of IOCs: e.g., we found that the IOCs receiving intensive reports tend to be short-lived, typically disappearing after a month.

On the other hand, organizations do not seem to react quickly to the release of IOCs: it could take around 2 days for AV scanners to include the hash of new malware and over 12 days for web scanners to update their domain and IP blacklists, after related information was made public by the blogs. Also, we found that a buffer overflow vulnerability CVE-2012-0158 was first reported by AlienVault in April, 2012 for APT attacks on the military and aerospace industry and then showed up again in an article on TrendMicro, September 2012, for an attack on political organizations; later the same vulnerability was found in malware distribution (2013) and Spear Phishing campaigns on the film industry (2014) and banks (2015), indicating that such a long-standing IOC was not adopted timely.

In terms of the qualities of open-source intelligence, we found that *Hexacorn* and *Naked Security* often provide timely and comprehensive information about new attacks. For example, articles from Naked Security first reported three malicious name servers m.sea.sy, mod.sea.sy and sea.sy under the control of the Syrian Electronic Army for DNS hijacking. Also, some IOCs apparently are more predictive than others. An example is the name server "132.248.49.112", which was reported by 19 blogs for the multiple-theme Spear Phishing attack and remained unchanged for 140 days.

Contributions. The contributions of the paper are as follows:

• *Novel IOC discovery technique*. We present iACE, the first fully-automated technique for generating OpenIOC compatible, semantic-rich intelligence, which addresses the emerging challenge in the effective analysis of massive open-source data for timely CTI gathering. Our approach leverages the unique features of IOCs and the way they are described in mainstream technical articles to come up with a specialized, scalable information extraction technique that achieves high accuracy and coverage. Our evaluation shows that iACE can effectively recover valuable attack indicators from popular technical blogs and convert it into industry-standard, machine-readable threat intelligence, which cannot be done by any existing techniques, to the best of our knowledge.

• *New findings*. Running iACE on over 71,000 articles from 45 most popular technical blogs across 13 years, our study sheds new light on the effectiveness of such open-source intelligence exchange, and the impact that it may have on the security industry and the adversary's strategies. The new understandings are invaluable for improving the IOC release, discovery and utilization process, contributing to the better protection of organizations' information assets.

2. BACKGROUND

2.1 Cyber Threat Intelligence

CTI gathering. As mentioned earlier, CTI is a collection of information that details the current and emerging security threats, which enables an organization to determine its responses at the strategic, operational and tactical levels [42]. At the center of CTI are IOCs, which elaborate the forensic artifacts of an attack and can therefore be used to analyze the attack once it happens or counter it during its execution. An IOC includes not only individual data fingerprints involved in a specific attack, such as an attack domain, the MD5 of the malware delivered, but also the context of the attack and an analysis of the adversary's behavior, like the type of the attack or the specific technique deployed (e.g., change to the Windows Registry). To find out such information, CTI gathering includes identification of the adversary's tools, techniques and attack procedures, which, together with the fingerprints, helps an organization's security team to understand their security posture, detect early signs of threats and continuously improve their security controls.

The sources of CTI can be closed, e.g., a corporation's internal network traces, or public, such as technical blogs or online forums. Examples of public sources include the blogs of AlienVault, FireEye, Malwarebytes and Webroot. With the surge of cyber attacks in recent years, a large number of attack artifacts have emerged, which has been extensively reported by the public online sources and aggressively collected by different organizations. To bootstrap this research, we reached out to a security company and obtained a list of 45 blogs which were operated by renowned organizations and practitioners. These blogs altogether covered major security incidents in the world and were consistently publishing verified IOCs.

Table 1: Examples of iocterms and their corresponding general type and context terms.

General type	iocterms	context terms
IP	Email/ReceivedFromIP, PortItem/remoteIP	email, IP, received, remote, port
Hash	ServiceItem/serviceDLLsha1sum, FileItem/Md5sum	service, dll, sha1, md5
Datetime	Email/Date, EventLogItem/genTime	email, date, event, log, generation

We monitored these 45 blogs and were able to download as many as 71,000 articles. Also noteworthy is that the number of IOC-related articles there continued to grow in the past 13 years (2003/01 to 2016/04), from 20 to 1,000 per month, with an increased rate of 500% (see Figure 5a). While the volume of articles we have collected is substantial, it only constitutes a small piece of the IOC landscape. Rapidly collecting and sharing such information, and deploying it to various security systems is the key to quickly detecting, responding and containing different kinds of security threats organizations face, which requires the descriptions of IOCs to be standardized and machine digestible. To this end, multiple IOC frameworks have been proposed, including OpenIOC [9], STIX [8] and yara [12], with each format easily convertible to the others. In our research, we utilized OpenIOC for automatic extraction of CTIs from technical articles.

OpenIOC framework. OpenIOC is an extensible XML schema created by Mandiant to record technical characteristics that identify a known threat, an attacker's methodology, or other evidence of a compromise. As an example, Figure 1 shows how to use the OpenIOC format to describe a family of POS malware, *Alina* [30]. Such a description includes two components, a header and a definition. The header part has a summary of the attack (under the `description` tag) and the source of the information (under `authored_by` and `authored_date`). The definition contains a set of indicator items (under `Indicator Item`), each representing an IOC (`Content`) and its context (`Context`). Two such items are illustrated in Figure 1. In each of them, the `document` attribute under `Context` gives the main category of an IOC (e.g., process, event, file, registry, etc.) and `search` elaborates on its subcategory using an *iocterm*, essentially a concatenation of common terminologies (e.g., process, name, etc.) security professionals use to describe the IOC. The OpenIOC standard provides 600 such iocterms, covering various types of artifacts left by different attacks, such as cookie history, DNS entry, Email information, Hook items and others. The examples of iocterms related to IP, hash and datetime are shown in Table 1. For example, In the case of IP, its iocterms describe 15 different contexts, such as spammer's IP (Email/ReceivedFromIP), IP in malicious infrastructure (RouteEntryItem/Destination), or C&C attack's remote IP (ProcessItem/PortList/PortItem/remoteIP).

In Figure 1, the two indicator items record the registry change made by the Alina malware: the first shows the location where the change happens, i.e., the path Windows\CurrentVersion\Run with the context RegistryItem/KeyPath, and the second identifies the name of the code dropped there, i.e., `ALINA`.

2.2 Information Extraction

To collect IOCs automatically, we leveraged a set of NLP tools. Also, due to the unique characteristics of this open problem, the generic information extraction techniques cannot achieve good performance for our tasks, and therefore we also applied *graph mining* techniques to analyze relations between context terms and IOC anchors. These techniques are briefly introduced here.

Dependency parsing. Dependency parsing is an NLP technique for describing grammatical relations between words in a sentence. Such relations include direct object, determinant, noun compound modifier and others. Also, the content of a relative clause is further analyzed to identify the dependencies between the words it includes. For example, the sentence in Figure 2 is parsed into dependency relations, such as the determinant relation between "trojan" and "the", `det(Trojan-2, the-1)` (where the number shows the position of the word in the sentence), and the nominal-subject relation between "trojan" and "download", `nsubj(downloads-3, Trojan-2)`. Each of the formulae represents a binary relation between a *governor* (the first term) and a *dependent* (the second one).

Such dependency relations within a sentence form a directed and weighted graph (V, E, W), where each token is a vertex in V, and the relation between them is represented by a set of edges in E. Each arc connecting two tokens can also be assigned a *weight* W to differentiate the relation between them from others. Figure 2 further illustrates the dependency graph for the example sentence. The state-of-the-art dependency parser (e.g., Stanford's typed dependencies system [20]) can achieve a 92.2% unlabeled attachment score (UAS) in discovering the grammatical relations in a sentence. In our research, we utilized the relations discovered by the parser to capture the semantic links between context terms and an IOC token. For example, the dependency of `clickme.zip` on "attachments" in the sentence "all e-mails collected have had attachments clickme.zip" reveals a compound relation between the terms (the content of the "attachment" is `clickme.zip`), which falls in line with the descriptions typically explaining the security issues related to email attachments.

Content term extraction. Another set of techniques extensively utilized in information extraction is content term extraction, which automatically determines important terms within a given piece of text. It includes parts-of-speech (POS) tagging that labels a word corresponding to a particular part of speech (such as nouns and verbs), and phrase parsing that divides sentences into phrases logically belonging together. Specifically, after parsing phrases from the given content, a POS tagger labels its terminological candidates, such as syntactically plausible terminological noun phrases. Then, these candidates are analyzed using statistical approaches (e.g., point-wise mutual information) to find out important terms.

Graph mining. Our approach tailors graph mining techniques to analyze the dependency graphs constructed from sentences of interest. Graph mining is a structural data mining problem with the purpose of identifying distinguishing characteristics (such as common subgraph) of graphs. The problem can be stated as follows. Given a dataset of graphs $G_i \in D$, with $i = 1...N$, each graph $G_i = (V_i, E_i)$ is a collection of vertices $V_i = \{v_{i1}, \cdots, v_{in}\}$ and edges $E_i = \{(v_a, v_b)|v_a, v_b \in V_i\}$. G_i may also have labels on its nodes and edges, which are drawn from a common label set of the whole dataset D. Also, each graph G_i is in a *class* with a label $c_i \in C$. The goal of the *graph classification* problem is to learn a model $f : D \to C$ that predicts the class label for any graph, that is, classifying the graph to a class based on its similarity to other graphs as measured by various *graph kernel* methods, such as direct product kernel [27] and subtree kernel [39]. In our research, we utilize graph mining to determine whether a sentence involving context terms and an IOC token indeed contains IOC.

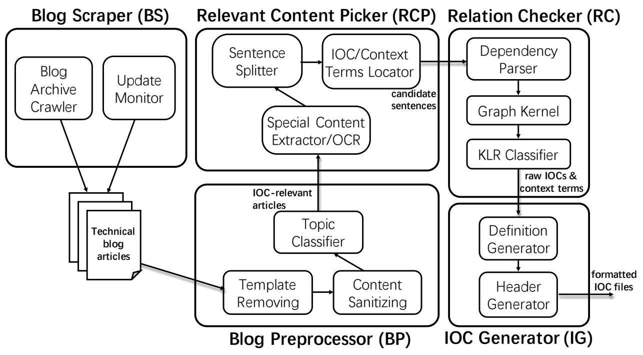


Figure 3: The architecture of iACE.

3. IACE: DESIGN AND IMPLEMENTATION

Although the generic problem of information extraction from natural language text is hard, particularly when high accuracy and coverage are required (see Section 7), articles documenting IOCs tend to utilize common technical terminologies (e.g., IP, process, etc.) and present the artifacts and their context in a predictable way (e.g., utilizing compound dependency, see the example in Figure 2), thereby making identification of IOCs more manageable. These unique features were fully leveraged in the design of iACE, which utilizes high-confidence context terms from iocterms and a set of regexes to locate the sentences likely containing IOCs. Then iACE analyzed the relations just between those anchors (the context terms and the putative IOC token) against a model learned offline to accurately and efficiently capture IOCs. Below we present the high-level design of this technique and explicate how it works through an example.

Architecture. Figure 3 illustrates the architecture of iACE, including a blog scraper (BS), a blog preprocessor (BP), a relevant-content picker (RCP), a relation checker (RC) and an IOC generator (IG). The BS first automatically collects ("scrapes") technical articles from different technical blogs, removing irrelevant information (e.g., advertisements) from individual blog pages. These articles are then inspected by the BP, using NLP techniques to filter out those unlikely to contain IOCs. For each article considered to be IOC-relevant, the RCP converts all its content, including pictures, to text when possible, breaks the content into sentences and other special content elements (tables and lists) and then searches among the sentences for those likely involving IOCs with a set of context terms and regexes, which describe the formats of IOCs such as IP addresses (Section 3.1). The context terms here are automatically extracted from iocterms and can also come from other manually labeled sources, and the regex is built specifically for each IOC type and its consistency with the content terms is first checked before the sentence is selected. For each sentence, the RC analyzes its grammatical structure connecting the context anchors (context terms) and the IOC anchor (the string matched by the regex), using a learned model to determine whether the latter is indeed an IOC, and if so, mark the IOC and the context terms (Section 3.2). Using such labeled content, the IG automatically creates the header and the definition components, including the indicator items for all the IOCs identified, according to the OpenIOC standard.

An example. Here, we use the excerpts of an article posted on Trustwave (Figure 1) to go through the whole IOC extraction procedure. From the title and other keywords identified from the article, using content term extraction (an NLP technique), iACE detects that the article is likely to contain IOCs for a malware infection, and then marks the link content and summarizes the article using *TextRank* [32] to construct the header component. After that, it converts the figures in the article into text using *Tesseract* [6], and searches for the context terms and IOC token on each sentence.

Among all the sentences captured, the following one is found to include a string fitting the description of a key path of registry, along with the context terms "registry" and "path": "It includes the addition of the registry value on the path HKCU\Software\Microsoft\Windows\CurrentVersion\Run that virus use to maintain persistence." This sentence is then analyzed against the IOC recognition model built through relation analysis, which confirms that the connection across the terms and the IOC anchor here ("HKCU\Software\Microsoft\Windows\CurrentVersion\Run") is indeed commonly used to describe the key path of an injected registry item. As a result, these terms and the IOC are labeled. From such content, the technical details of the attack reported by the article are automatically extracted and presented in the OpenIOC format (Figure 1).

Assumptions. The current design of our system is for analyzing technical blogs, under the assumptions that the blog writers present their findings in a professional fashion. Such assumption is reasonable intuitively, as the blogs are written by security professionals and their purpose is to quickly share technical details to their peers[1]. It was also confirmed by our analysis of the labeled dataset `DS-Labeled` (see Section 4.1 for description). All their descriptions of IOCs were found to be in line with our assumptions (Section 3.1), involving professional terms and predictable grammatical structures. On the other hand, the effectiveness of the technique on other types of IOC open sources like online forums needs to be further studied (Section 6).

3.1 Relevant Content Identification

As mentioned earlier, to automatically extract IOCs from technical blogs, we first need to scrape related web pages from blog sites, pre-process their content and remove noise, before filtering out those irrelevant and identifying from the remaining articles the sentences that may carry IOCs for the follow-up relationship check and IOC extraction. Here we elaborate on these individual steps.

Blog scraping and pre-processing. Our blog scraper is essentially a crawler designed to continuously monitor a list of technical blogs to collect their articles. For each blog site, the BS first scraps all its existing articles before it is set to monitoring mode to look for new ones. Specifically, the scraper performs breadth-first crawling on the blog, starting from its homepage to explore each link it discovers, until no new link can be found. A problem here is that the web pages gathered in this way may not all be articles, and may also contain login pages, contact pages, and others. To automatically remove these unrelated pages, we leverage two unique observations: the articles posted on the blog are all framed within the same HTML template, which is very different from those used by other pages such as login, contact, and others; also, on any blog site, more article pages are hosted than other types of pages. Based on the observations, the BS compares each page's DOM tree with those of others to find out a small set of pages framed over the templates unlike the ones used by the majority of the pages. These pages are considered to be outliers and dropped from the dataset scraped from the blog site. In our implementation, the BS uses the python library `beautifulsoup` [2] to extract from each page its HTML template (with tags and attributes but no content) and groups the pages using their templates' hash values to identify those unrelated to articles (the ones outside the group).

Even on those indeed relevant web pages, there still is content unrelated to the technical articles, such as blog contributors' pictures, advertisements, featured content, etc. Such content needs to be removed before the pages can be used for IOC identification. This purpose is served by another pre-processing step the BS takes to get rid of such non-UGC (*user-generated-content*) data from each page. Specifically, our approach compares all pages' DOM trees to find out the nodes with the non-UGC, characterized by their largely unchanged content across these pages (e.g., blog contributors' photos will be the same across different articles). This is different from the article content generated by the user, which varies significantly between pages. Such a difference is captured by an information-theoretic metric called *composite importance* (CI) [47] that is used by the BS to measure the uniqueness of each node on each page's DOM tree with regards to other pages: the nodes found to be less unique (rather common across different pages) are discovered and dropped from the DOM tree, using an algorithm proposed in the prior research [47].

Under such a "sanitized" DOM tree, still some content cannot be directly analyzed by our text-based approach, including images and embedded PDF files. So, the last pre-processing step is to convert such content into text, if it indeed involves text information. To this end, we incorporated into our implementation of an optical character recognition engine *Tesseract*. Tesseract [6] is capable of discovering texts within an image with an accuracy of 99% in general. However, for the images collected from blogs, we found that the accuracy went down to merely 70%, due to the low quality of the images and the non-dictionary words they often have. To address this issue, we used *Gimp* [4] to resize the blog image a for better image quality, and add to Tesseract the new words (e.g., IP, MD5, HTTP) discovered from the text of an article. We tested this approach on 100 randomly sampled images from 10 blogs, which was found to push the accuracy of Tesseract to above 95%.

Topic filtering. Once all the article pages have been selected and pre-processed, we start looking into their content, first removing those not including any IOCs. Examples of such articles are those for product promotion, news or software update, which we call *non-IOC articles*. To separate the non-IOC articles from those with IOCs (called *IOC articles*), iACE runs the TC, a classifier using a set of features as described below:

• *Topic words*: Intuitively, an IOC article focuses on security risks and vulnerabilities, whose theme is reflected by its *topic terms* (e.g., malware, exploit). These terms are less likely to appear in a non-IOC article. *Topic term extraction* is an extensively studied NLP technique. In our implementation, we utilized an open-source tool `topia.termextract` [10] to collect the terms within each article. The top 20 terms discovered, together with their frequencies, are part of the features for the classification.

• *Article length*: Since the blog sites are meant to be the channels for IOC exchanges, the IOC article it contains tends to be longer, including detailed descriptions of IOCs and their context, while non-IOC articles in technical blogs are often news and digests, and hence tend to be shorter.

• *Dictionary-word density*: Compared with non-IOC articles, IOC articles tend to have a lower dictionary-word density, because most IOCs are non-dictionary words (e.g., IP, hash values, file path). Our implementation employs the `enchant` library [15] to find dictio-

[1]Note that asking those professionals to directly post formalized IOCs may not be realistic in the near future, due to the complexity of manually creating the content and their intent to get humans involved in the discussion. As a supporting evidence, we examined all 45 blog sites and found only one of them export IOCs in some articles (AlienVault).

Table 2: Examples of regexes.

General type	Regex
IPv4	(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.)\{3\}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)(/([0-2][0-9]|3[0-2]|[0-9]))?
hash	\b[a-fA-F\d]{32}\b|\b[a-fA-F\d]{40}\b|\b[a-fA-F\d]{64}\b
int number	((?<=[\s([\s\(\[\{:\+\-=\\])|^)(\+|-)?\d+(?!([a-zA-Z0-9]|\.\S))
float number	((?<=[\s\(\[\{: \+\-=\\])|^)(\+|-)?(\d+)?\.(\d+)(?!([a-zA-Z0-9]|\.\S))

nary words in an article. Then, its density within the article is calculated as the ratio of the dictionary words with regards to all the words the article contains.

Using these features, we ran a support vector machine (SVM) to train a classification model over `DS-Labeled`, including 150 IOC articles and 300 non-IOC articles. The model was evaluated through a 10-fold cross validation, and found to achieve a precision of 98% and a recall of approximate 100%. We further used this TC to analyze all 71,000 articles from the unknown set `DS-Unknown` (described in Section 4.1) and manually validated 500 instances randomly selected from the classified results (Section 4.2). The validation shows that the classifier had a high accuracy (96%) and coverage (99%).

IOC sentence identification. From each IOC article, the relevant content picker identifies the sentences, tables, and lists likely to include IOCs before they are further evaluated by the relation checker (Section 3.2). Such content is selected from the article based on the *anchors* they contain, i.e., context terms and putative IOC tokens. Specifically, the RCP parses the HTML content of each article, breaking the text into sentences and detecting tables and lists. From each sentence, we look for the presence of both the string matched by the regex and its compatible context terms: e.g., "hash" goes with an MD5 checksum. From tables and lists, we increase the search scope for context terms also to table headers, captions and the nearest sentences (see supplementary material). Figure 2 shows an example.

The OpenIOC standard specifies 600 categories of IOCs using a list of iocterms [9]. We further summarized these IOCs into 19 different data types, including `IP`, `hash`, `int`, `float`, and others. The regex for each of these categories was manually constructed and carefully examined. As we can see here, such expressions could introduce false positives (e.g., the string type matching many IOC strings, even though oftentimes, we only seek non-dictionary words), if we do not also look at the context terms and the relations between identified tokens. These terms are automatically collected from the 600 iocterms through tokenizing dictionary-word elements within each iocterm and by removing common terms like "item" . We further used *Semanticlink* [14] to recover other semantic related terms, e.g., "portable executable" (related to PE).

Altogether, we gathered 5,283 context terms in our research, which were found to indeed provide good coverage of the common terminologies used by technical blogs. In our research, we gathered 80 public IOC files and their corresponding blog articles (according to their description tags) from labeled dataset. By manually inspecting the sentences carrying the IOCs, we found that all such sentences also contain at least one context term and all such terms are on the list we created.

3.2 Relation Checking and IOC Creation

Although context terms and regexes can find us the sentences likely involving IOCs, they are *insufficient* for detecting true IOCs with high accuracy. For example, Figure 2 shows four sentence pairs collected from two articles posted on AlienVault. As we can see, each pair contains both context terms, like "download", and the strings fitting the descriptions of their corresponding IOCs, such as "3344"; however, the fifth one does not include any real IOC while the third one does. Fundamentally, the coincidence of relevant tokens (context term and IOC token) does not necessarily indicate the presence of an IOC-related description. Most important here is the consistency between the *relation* of these tokens and what is expected from the narrative of IOCs: in the above example, `3344` is supposed to be the *offset*, not the *PE* file. Actually, in the case that such a relation is incorrect, even when the IOC extracted is indeed an attack indicator, its context can be wrong and as a result, the OpenIOC record created can be erroneous. The third sentence in the figure is such an example.

As mentioned earlier, identifying IOCs is essentially an NER problem and connecting them to their context is an RE issue. In the NLP community, solutions to the problems are pipelined: individual name entities within a sentence are first recognized (i.e., NER) and then their relation is established (i.e., RE). For our problem, however, this pipeline becomes unnecessary, since the putative tokens for an IOC and its context are already located in a sentence, and all we need to do is check the consistency of their relation with what is expected to happen between them. In other words, we are in the position to address both the NER and RE together. On the other hand, existing RE techniques are inapplicable here, because they are designed to work on the nominal relation between two nouns, whereas the links between an IOC and its context terms are more diverse, including nominal, verb and adjective (e.g., "attachment", "download" and "injected"). To handle such diverse relations, we came up with an idea that models the analysis on the grammatical connection between the IOC candidate and its context as a *graph mining* problem [23]. This allows us to apply graph similarity comparisons to detect the presence of the desired relation, which achieves both a high accuracy (95%) and a high coverage (above 90%) that the more generic RE techniques cannot attain [24]. Below we present how the approach works.

Relation representation. To analyze the relation between an IOC candidate and a context term, our approach first uses a *dependency parser* to transform a sentence into a dependency graph (DG). A DG describes the grammatical connections between different linguistic tokens, with verbs being the structural center of a clause and other tokens directly or indirectly depending on the center. Such a dependency describes the grammatical connection among tokens such as direct object, determinant, noun compound modifier, etc. Formally, a DG is a directed and weighted graph $g = (V, E, W)$, where words in the sentence are nodes V, two related nodes are connected by an edge E and the specific grammatical relation linking them together is modeled as an edge weight W. In our research, the DG was constructed using the Stanford dependency parser [20], the state-of-the-art open-source tool that converts text to simplified grammatical patterns suitable for relations extraction.

Unlike the more generic RE techniques, which work on the DG of the whole sentence, our approach takes advantage of the known anchors to focus on the smallest dependency graph g_{n_i,n_c} connecting an IOC candidate n_i and a context term n_c together. This is because the information carried by this subgraph (called *core*) is most relevant to the understanding of the relations between the anchors, which is all we care about (see an example in Figure 4). Note that we also add negation dependencies as child nodes or sibling nodes of the nodes on the core to capture IOC-related negative descriptions. As an example, in the fourth sentence in Figure 2, the relation between the context term "modify" and the IOC token "`AndroidManifest.xml`" is affected by the word "not", which adds a negation dependency on the verb "modify".

Similarity comparison. Over a dependency subgraph g_{n_i,n_c} modeling the relation between a context term and an IOC candidate, we

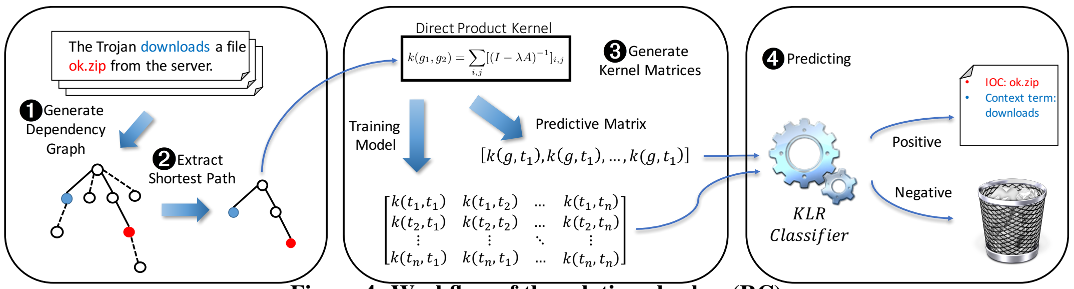

The Trojan downloads a file ok.zip from the server.
1 Generate Dependency Graph
2 Extract Shortest Path
Direct Product Kernel
$k(g_1,g_2)=\sum_{i,j}[(I-\lambda A)^{-1}]_{i,j}$
3 Generate Kernel Matrices
Training Model
Predictive Matrix
$[k(g,t_1),k(g,t_1),\ldots,k(g,t_1)]$
$\begin{bmatrix} k(t_1,t_1) & k(t_1,t_2) & \ldots & k(t_1,t_n) \\ k(t_2,t_1) & k(t_2,t_2) & \ldots & k(t_2,t_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(t_n,t_1) & k(t_n,t_1) & \ldots & k(t_n,t_n) \end{bmatrix}$
4 Predicting
KLR Classifier
Positive
Negative
IOC: ok.zip
Context term: downloads

Figure 4: Workflow of the relation checker (RC).

want to find out whether another subgraph is similar, which indicates that the same relation also exists between its anchors. This similarity comparison is important, since it is the foundation for classifying a sentence, determining whether IOC relations are indeed there to bind anchors or they are not. Given the fact that now we only need to work on simple subgraphs with a few labeled nodes (see Figure 4), the focus is to compare the paths linking corresponding nodes (with identical labels) across the subgraphs. For this purpose, we customize a well-known graph mining technique, called *direct product kernel*, a function that measures the similarity of two graphs by counting the number of all possible pairs of arbitrarily long random walks with identical label sequences [27].

Specifically, let $g_1 = (V_1, E_1, W_1)$ and $g_2 = (V_2, E_2, W_2)$ be directed weighted graphs. The direct product between g_1 and g_2 is a directed weighted graph $G = g_1 \times g_2 = (V, E, W)$, where

$$V = \{(v_1, v_2) \in V_1 \times V_2\}$$

$$E = \{(u_1, u_2), (v_1, v_2) \in V \times V : (u_1, v_1) \in E_1, (u_2, v_2) \in E_2\}$$

$$W = \begin{cases} 1, & \text{if } W_1(u_1, v_1) = W_2(u_2, v_2) \\ 0, & otherwise \end{cases}$$

In other words, for each pair of nodes (u_1, u_2) and (v_1, v_2) in the new graph G, they are adjacent (connected by a directed edge) if and only if an edge with the *same* direction and weight exists both between u_1 and v_1 in g_1 and between u_2 and v_2 in g_2. It is important to note that we bring the weights into the product to compare the *type* of the grammatical relation between two nodes (words): only when the adjacent word pairs in two sentences have the same grammatical relation, will that relation be preserved in the new graph. Usually, it takes $O(|V|^4)$ to compute this direct product. However, all the subgraphs we consider are just "paths" (after removing the directions) between a context term and an IOC candidate, whose direct product can be computed in $O(|V|^2)$.

Over the direct-product graph G, we calculate the similarity between the two subgraph g_1 and g_2 as follows:

$$k(g_1, g_2) = \sum_{i,j}[\sum_{l=0}^{\infty} \lambda^l A^l]_{i,j} = \sum_{i,j}[(I - \lambda A)^{-1}]_{i,j}$$

where each entry of A^l is the number of the walks of length l from v_i to v_j in the direct product graph $G = g_1 \times g_2$, hence A^l can be calculated as the l-th power of the adjacency matrix of G; λ is the decay constant and $\lambda \leq \frac{1}{min(d_i, d_o)}$ with d_i, d_o being the maximum in-degree and out-degree of G, representatively. In our research, we set $\lambda = \frac{1}{min(d_i, d_o)}$.

Classification. Based on our customized direct-product kernel, we run a classifier (the relation checker) to determine the presence of IOC relations between a context term and an IOC candidate within a sentence. The classifier is trained over `DS-Labeled` with 1500 positive instances and 3000 negative instances. From the dataset, a model is learned to work on a kernel vector generated from the features of a subgraph g_i: $(m_1, \cdots, m_j, \cdots)$, where $m_j = k(g_i, t_j)$ and t_j is the subgraph for the jth instance in the training set. In other words, each new instance g_i is classified into the positive set (with the IOC relation) or the negative set (without the relation) based on its similarity with every instance in the labeled set. This classification is executed efficiently with multi-threading in our implementation.

The classifier can be trained with different kinds of machine learning algorithms. Our implementation utilizes logistic regression, since the algorithm works well on a small labeled dataset. More specifically, it optimizes the log likelihood function to determine a probability that is a logistic function of a linear combination of the training dataset, and every training point has a certain influence on the estimated logistic regression function. In our research, we compared the recall and precision of five classification models on the labeled dataset through a 5-fold cross-validation. With the regularization parameter set to 3.0, our logistic regression classifier yielded the best results. On unknown set `DS-Unknown` , this classification model achieved a high accuracy (with a precision of 95% and recall of 90%).

IOC creation. After identifying the IOC and its corresponding context terms, our IOC generator can automatically convert the CTI content of a technical blog to the OpenIOC record. Specifically, each indicator item in the record is created by filling in the `search` attribute in the `Context` tag with a context term and the `Content` tag with its corresponding IOC. The content of other fields on the item can be derived from these two fields. For example, the `type` attribute of the `Content` tag and the `document` attribute of the `Context` tag are actually the iocterms of the IOC discovered.

For the header of the record, the IG generates the content for the `Description` tag using the open-source automatic summarization tool `TextRank` to create a summary for the blog article. Also, the original blog link address is used to fill the `link` tag, and the content of `authored_by` and `authored_date` tag are set to our prototype name and the file generation time.

4. EVALUATION

4.1 Settings

In our study, we ran our implementation of iACE to automatically analyze 71,000 real-world technical articles, on an R730xd server with 40 of Intel Xeon E5-2650 v3 2.3GHz, 25M Cache CPUs and 16 of 16GB memories. Here we explain the datasets used in the study and the parameter settings of the system.

Datasets. We utilized two datasets in our study: a labeled set for training our topic classifier (Section 3.1) and relation checker (Section 3.2), and an unknown set for evaluating our technique.

• *Labeled dataset (`DS-Labeled`).* The dataset contains 80 IOC files and their corresponding blog articles. These IOC files were collected in our research from two public feeds, iocbucket [29] and openiocdb [1], both providing threat intelligence through OpenIOC items. Under the `description` tags of these items, we found the links pointing to these items' sources and recovered 150 articles from 22 blogs, including their HTML pages and image files. Also, we manually gathered, from the same blogs, 300 other articles. Each of them was manually checked to ensure that they do *not* have any IOCs but contain some IOC-like strings (e.g., IP addresses, MD5, etc.). Most of them are technical news or articles for product promotion.

From these articles, we further extracted two kinds of sentences, those with IOCs (*true IOC sentences*) and those without but involving IOC-like strings (*false IOC sentences*). More specifically, the 1,500 true IOC sentences were identified using the context terms and IOC tokens specified in the OpenIOC items we collected, and further manually inspected to ensure their correctness. The 3,000 false IOC sentences were those matched by the regular expressions and context terms used by iACE (Section 3.1) but did not include any IOCs, as confirmed by a manual check.

• *Unknown set* (`DS-Unknown`). As mentioned earlier, the dataset used for evaluating our system was gathered from 45 security-related technical blogs. These blogs are well recognized to be the leading sources for CTI collection, which includes AlienVault, Malwarebytes, and others (see supplementary material for the full list). On these blogs, we ran a crawler that scraped 71K articles posted there between 2003/04 and 2016/05. Figure 5a shows the increase of the number of the articles per month on these blogs over 13 years.

Parameter settings. In the experiments, the parameters of our prototype system were set as follow:

• *Kernel decay constant* (λ). The decay constant is a parameter for calculating a direct product when the impact of a long random walk needs to be discounted (Section 3.2). It was set according to the convention of evaluating the kernel function: $\lambda = 0.9$ if $\frac{1}{min(d_i,d_o)} = 1$ and $\lambda = \frac{1}{min(d_i,d_o)}$ otherwise.

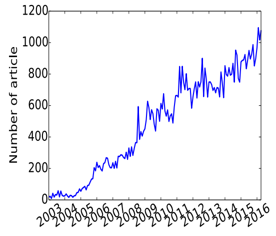
• *Inverse of regularization strength* (C). Regularization is a parameter for reducing the over-fitting to the labeled data when we built the relation models using logistical regression. In our implementation, we utilized a $C = 3.0$, which gave the best performance among other C values from 0.3 to 15.

• *Threshold of sentence length* (l). The current dependency parser is limited by its capability to process long sentences. When a sentence grows longer, the accuracy of parsing goes down and its overhead goes up. In our implementation, we set the maximum length of the sentence to 200 words. Given IOCs may exist in sentence length larger than 200 words, we directly extract IOCs from sentences longer than 200 words without building dependency graphs. In our implementation, we directly extract the IOCs from sentences longer than 200 words if the sentence has two continuous IOC tokens or has more than five IOC tokens which were split by short terms (less than 5 characters) (see supplementary material).

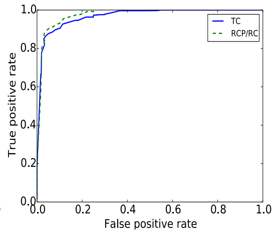
4.2 Results

Accuracy and coverage. In our study, we first evaluated the topic classifier (i.e., TC) over the 450 IOC and non-IOC articles, and the relevant content picker and the relation checker (i.e., RCP/RC) over 1,500 true IOC sentences and 3,000 false IOC sentences in `DS-Labeled`, both using a five-fold cross-validation. Our prototype achieved a precision of 98% and a recall of 100% in finding IOC articles, and a precision of 98% and a recall of 92% in identifying true IOCs and its context. Figure 5b illustrates the ROC curve of the RCP and RC.

Further, we ran our system over `DS-Unknown` across all 71K articles. Altogether, iACE automatically extracted IOC tokens and their context terms, and further converted them into OpenIOC items. To understand the accuracy and coverage of the information extracted, we sampled the unknown set using two different methods: we first grouped the articles in the unknown set according to their publication time (4 consecutive months per group), and then their publishers (45 blogs), and in each case, we randomly picked up a few articles from each group. Altogether, in this validation step, we manually inspected 820 articles and in total, 25K reported sentences, and concluded that iACE achieved a precision of 95% (23K out of 25K reported IOCs were correct), and a recall of 90% (across the articles, 90% of IOCs were reported).



(a) **The number of the articles per month.**



(b) **ROC curve of iACE.**

Figure 5: The increase in the number of articles over 13 years and the effectiveness of iACE.

Table 3: The number of samples and average accuracy and recall in each method.

Method	# of Groups	Samples per Group	Precision	Recall
Time span	39	10	93%	92%
Blog	45	10	96%	91%

To compare our approach with state-of-the-art alternatives, we ran iACE against the top-of-the-line NER tool Stanford NER [26] and the commercial IOC identifier integrated within AlienVault OTX [17]. Note that none of them (actually none of the existing systems we are aware of) can also identify the context for an IOC and therefore generate machine-readable OpenIOC items. In our experiment, Stanford NER was trained on our labeled *true/false IOC sentences* as describe in Section 4.1 (the same set for training iACE). For AlienVault OTX, we utilized its API to submit articles to their web service and retrieve the IOC tokens identified. This study shows that our relation-based approach is indeed much more effective. Specifically, we ran all three systems on 500 randomly selected articles from 25 blogs in `DS-Unknown` and compared their findings: iACE extracted the IOC items across 427 OpenIOC categories, such as `FileDownload HistoryItem/FileName`, `Email/ReceivedFromIP`, with a precision of 98% and a recall of 93%, while OTX could only find the IOCs in 8 categories (IP, hash value, domain, etc.) with a precision of 72% and a recall of 56%, a performance mirrored by Stanford NER. Both OTX and Stanford NER tend to introduce a lot of false positives: for example, OTX treated the reference links of articles as malicious URLs. Table 4 summarizes the result of this study.

Table 4: Accuracy and coverage comparison of iACE, AlienVault OTX and self-trained Stanford NER.

Tool	Precision	Recall
iACE	98%	93%
AlienVault OTX	72%	56%
Stanford NER	71%	47%

Performance. To understand the performance of iACE, we measured the time it spent on each real-world article in the unknown set and the breakdowns of the overhead in each analysis stage, BP, RCP, RC, and IG. In the experiment, our prototype was running on our R730xd server, using 40 threads. On average, it took around 0.25 second to inspect one article, as illustrated in Table 5. This result provides strong evidence that iACE can easily scale to the level expected for processing a massive amount of CTI generated every day.

Table 5: Running time at different stages.

Stage	average time (ms/article)	Std Deviation (ms)
BP	5	-
RCP	20	2.5
RC	252.5	37.5
IG	2.5	-
total	278.6	-

In the meantime, considering the DG parser is largely affected by sentence length, we measure the performance of the RC model in different sentence lengths. Figure 6 illustrates the average running time on sentences of different lengths. We found that the running time of iACE gradually increases when sentence length increases. This is because iACE only extracts the shortest paths linking each pair of a context token and the IOC token, which mitigates the impact of sentence length .

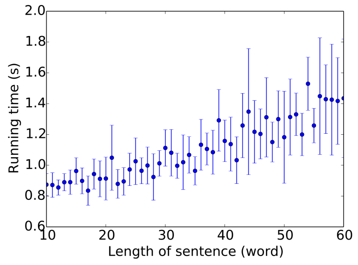


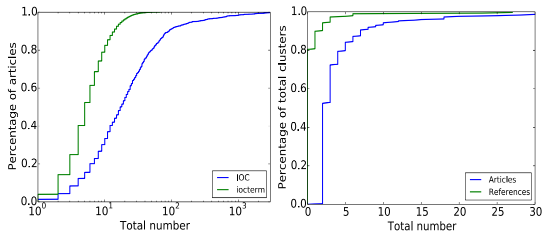
Figure 6: Average running time on the sentences in different lengths.

5. MEASUREMENT AND ANALYSIS

The IOCs automatically extracted from the 71,000 articles on 45 blogs present to us a comprehensive view of the cyber threats that the world has been facing in the past 13 years. By analyzing these IOCs, looking at their relations and evolution over a large time frame, across thousands of articles, our study brings to light new findings of attacks' strategies and evolution, as well as new insights into the impact of open-source intelligence. Particularly, we found that hundreds of apparently independent attacks were actually related, sharing unique IOCs such as IP address, register's email and domain, etc. Among them, a set of command and control (C&C) servers were reported by 396 articles (with little reference among them) and linked to over 7,000 unique IOCs over a four-year span, indicating that these separate attacks could all be part of a massive, previously unknown campaign. Further, through correlating different articles, we observe that the same vulnerability has been continuously utilized for a long period of time. Also, the IOCs intensively reported tend to be short-lived, demonstrating the possible impacts of the CTI on the adversaries. On the defender side, the response to the IOCs seems less timely than one hopes, taking days to get IOCs into malware scanners, blacklists etc. Also, our study reveals the quality of the IOCs reported by different blogs and the ways these blogs react to emerging threats, which helps better understand the effectiveness of such intelligence sources. Below we elaborate on the details of this study.

5.1 Landscape

Our study shows that these technical blogs are indeed a gold mine for threat intelligence gathering: altogether, 900K IOCs and their context were recovered from 71,000 articles (20K IOC articles), including 20K exploit hash values, 55K registry key, 58K malicious IPs, 180K FQDNs etc. Table 6 presents the 10 IOC types (described by their corresponding iocterms) with the most instances found by iACE. We observe the largest amount of IOCs with the type PortItem/remoteIP, which was related to the popularity of drive-by-download, Phishing and other web attacks in the wild.



(a) **Cumulative distributions for the numbers of IOCs and iocterms per article.** (b) **Cumulative distribution of the numbers of articles and reference per cluster.**

Figure 7: Distribution of IOCs across different articles and clusters across different blogs.

Table 6: Top-10 iocterms with the largest number of IOCs.

iocterm	# of IOCs	general type
PortItem/remoteIP	18,243	IP
RegistryItem/ValueName	12,572	string
UrlHistoryItem/HostName	12,020	URL
RegistryItem/Path	11, 777	string
FileDownloadHistoryItem/SourceURL	10,324	URL
ServiceItem/serviceDLLmd5sum	9,908	hash
PortItem/remotePort	9,831	int
FileDownloadHistoryItem/FileName	8,720	string
Email/ReceivedFromIP	8,225	IP
FileItem/FileExtension	8,100	string

We looked into the distribution of IOCs across different articles and blogs, as illustrated by the cumulative distributions for the numbers of IOCs and iocterms displayed in Figure 7a. On average, each article contains 52 IOCs and 70% of the articles have more than 10 IOCs. Particularly, the blog *hpHost* has 350 IOCs per article, the largest one among the blogs we inspected. Also, an article on *CyberCrime* talking about a Google redirection Spam reported 3,417 IOCs. When it comes to the diversity of IOC context, we found that on average, each article includes 6 different iocterms and 30% of articles have more than 10 different iocterms. Also interestingly, the blog with more IOCs does not necessarily come with more diverse IOC types: for example, even though *hpHost* has the largest number of IOC per article, each article only has 8 iocterms.

5.2 Understanding Threats

Through mining the IOCs across articles and blogs, we gained new insights into reported attacks and discovered connections never known before. Such connections shed new light on the way the adversaries organize their campaigns and adapt their techniques. Further linking the reported IOCs to auxiliary data sources reveals the impact of such CTI on the responses to emerging threats.

Correlation analysis. To understand the relations across different attack campaigns reported by articles, we studied the sharing of critical attack resources in these campaigns, through measuring their common infrastructure-related IOCs, including IP, register's email and domain. Specifically, using these IOCs, we were able to group all the articles into 527 clusters (each group with more than three articles): two articles were put in the same cluster if they share at least one IOC IP, email or domain. Note that we removed IP addresses in the private address ranges (10.*.*.*, 172.16.*.*, 192.168.*.*). To find out whether those in the same cluster actually refer to a common source, so they essentially talk about the same attack instance, we also looked at the URLs in each of the articles to identify those pointing to the 45 blogs and other well-known intelligent sources. Figure 7b illustrates the distribution of the clusters with various percentages of the articles involving such references. It turns out that many clusters (including thousands of articles) have low reference percentages: in other words, the authors of these articles apparently did not realize that the attacks they were documenting were related to other instances.

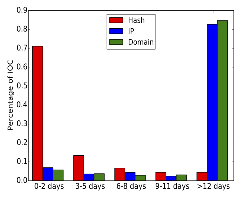
We further picked out 5 clusters with at least 25 articles but at most 5% of them include references to others. None of such reference-carrying articles were found to mention the relations among the attack instances reported by other articles in the same clusters. Also, a sampling of these articles did not show any references to other blogs not on our list. This indicates that the infrastructure relations linking all these attacks together have never been reported before, which was confirmed by our additional search for related literature across the Internet. Table 7 provides the information about those clusters. Most interestingly, we found that for the IOC "132.248.49.112", a shared IP reported by 19 blogs, turned out to point to a C&C campaign's name server. We believe that all the independently documented attacks actually belonged to a massive campaign whose scale was not known before. Note that this correlation effort is highly important because it informs us of the critical resources attackers share, which are likely to be their weakest link.

Table 7: The 5 Clusters.

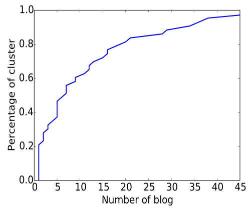
Cluster	# of articles	# of IOCs of mal. infra.	# of IOCs of total	# of references
1	396	7,363	10,533	21
2	178	4,271	8,110	3
3	30	215	960	0
4	28	897	1,302	0
5	25	897	1,222	0

Evolution. Looking into the C&C campaign reported by 396 articles and related to 7,000+ unique IOCs, we were surprised to find that it lasted for a long time (2009-2013), and continued to adapt its attack strategies and techniques. Specifically, it distributed malware by sending Spam emails to users, compromising legitimate websites, and others, and the vulnerabilities it exploited evolved from CVE-2010-1885 to CVE-2013-0422. In the meantime, the campaign utilized a small set of IPs for its C&C servers, and some of them share the same register email "gkook@checkjemail.nl". Apparently, taking down these servers could significantly affect the effectiveness of this long lasting, large-scale attack.

Another step we took to correlate the attacks reported by different articles was to cluster them according to the indicators of their attack vectors, including malware hash, vulnerability CVE and the content of the registries. The purpose is to understand the evolution of the vectors with regard to the releases of related IOCs. To this end, we looked at the shortest period of time, in terms of the number of consecutive months, during which a specific IOC (e.g., 132.248.49.112) was continuously covered by new articles. This period, which we call "decay time", demonstrates how long the attack instances related to the IOC continue to pop up before they can be stopped (at least temporarily). As illustrated in Table 8, in general, we found that the IOCs reported by a large number of articles tend to disappear quickly, indicating that either the related problems were quickly fixed or the adversaries reacted to the reports to change their strategies. However, there are long-lasting IOCs even though they are well known: as a prominent example, a buffer overflow vulnerability CVE-2012-0158 kept showing up in blog articles for four months, and after a few recesses, continues to appear in other attacks over a four year period! Specifically, it was used in APT attacks on the military and aerospace industry (2012/04),



(a) **Distributions of the duration after an IOC was released and before it was uploaded for a scan.**



(b) **Cumulative distributions for the numbers of first-reported blog per cluster.**

Figure 8: IOC impacts and timeliness of blog.

and then on political organizations (2012/09), generic malware distribution (2013), and more recently Spear Phishing attacks on the film industry (2014) and banks (2015). This clearly shows that organizations have not done their due diligence to adequately respond to the problem.

Table 8: Decay time of IOCs.

Decay time (month)	Avg. # of articles per month	% of total IOCs.
0-1	68	92%
1-2	23	5%
2-3	8	1%
3-4	6	1%
>4	3	1%

IOC impacts. Further, we studied the impact of such open-source intelligence on security protection: whether the security industry (e.g., anti-virus service providers) quickly responded to the reports of IOCs. To this end, we estimated the possible time intervals between the first release of the IOCs and their adoption by anti-virus (AV) tools and web scanners. In our research, we submitted a set of IOCs, including IPs and domains of malicious sources, and the hashes of malware to *VirusTotal* [11], a platform that hosts 56 mainstream AV scanners and CleanMX [22], an IP/URL scanner. From VirusTotal, we collected the time stamps for the first time when the submitted IOCs were seen by at least one of the AV systems. For CleanMX, we fetched its IP/URL blacklist database archived from 2009/01 to 2015/04. We found that 47% of the IOCs were updated to these systems before they were reported by the blogs. For the rest of them, Figure 8a shows the distributions of the duration after such an IOC was released and before it was first used for a scan. We observed days of delay before the IOCs were put in place for protection, if indeed the uploading time (or the time when the IOCs were first used for a scan) was close to the moment when the IOCs (IP, domains, hashes) were added to the systems. Particularly, for IPs and domains, the whole process often took more than 12 days. On the other hand, the malware hashes were often quickly added, in most cases within 2 days.

5.3 Understanding Intelligence Sources

The availability of the longitudinal data (the IOCs collected over a span of 13 years) also enables us to investigate the qualities of the indicators produced by different sources and their timeliness against new threats, as reported below.

Timeliness. Using the aforementioned attack clusters (see Table 7), we analyzed the distribution of the articles first reporting the attacks over different blogs, as shown in Figure 8b. We found that 10 blogs were responsible for the first report of 60% the clusters (each cluster likely to be a campaign). For example, the blog *Dancho Danchev* first report 12 clusters, each time involving 45 IOCs on average, which later also showed up on other blogs.

Table 9: Quality of selected intelligence sources (10 out of 45)

Blog	% of covered IOCs	% of covered iocterms	% of timely IOCs	% of robust IOCs
Dancho Danchev	42%	62%	14%	84%
Naked Security	43%	55%	54%	45%
THN	38%	38%	41%	51%
Webroot	54%	**79%**	13%	84%
ThreatPost	26%	37%	52%	29%
TaoSecurity	**57%**	61%	31%	68%
Sucuri	34%	35%	43%	52%
PaloAlto	39%	44%	15%	**87%**
Malwarebytes	32%	48%	26%	72%
Hexacorn	49%	57%	**59%**	76%

Table 9 shows the average percentage of IOCs first reported among all the IOCs finally discovered from a cluster (i.e., the number of first-reported IOCs and those only reported once vs. the total number of IOCs in a cluster). We found that most IOCs reported first by *Hexacorn* and *Naked Security* were also mentioned by other blogs later. Also, they provide a large amount of IOCs not documented by other blogs. We observed that even though *Webroot* only has an average of 13% of the earliest-reported IOCs for the clusters we monitored, 84% of its IOCs were not reported by other sources.

Completeness. On the other hand, the early reports often only contain a small portion of IOCs. In our study, we measured the percentages of the IOC tokens and their iocterms for different attack clusters that were included in the first report: 6 blogs reported more than 40% of the IOC tokens and 9 blogs covered more than 50% of iocterms (related to attack behavior) per cluster. We further checked the blogs whose articles give the most complete descriptions of attack clusters. Altogether, *TaoSecurity* were found to have the largest number of such articles.

Robustness. In our research, we compared the robustness of different IOC tokens, in terms of their stability across the whole period of an attack cluster (the clusters in Table 7). From the data mentioned above, we found that the name server, C&C server, registry email are the most robust indicators, which remained unchanged in 10 to 30 percent of the clusters we analyzed (see Table 9). Using such information, we further measured the blogs likely to report these tokens: the top blogs providing most of such tokens (i.e., the number of robust IOCs vs. the number of total IOCs in a cluster) are in Table 9. Interestingly, looking into the IPs of these servers, we found that many of them actually shared the same IP prefixes, which makes us believe that they might all come from a small set of malicious Autonomous Systems.

6. DISCUSSION

Our study shows that iACE makes an important step toward fully automated cyber threat intelligence gathering. This is significant not only for the convenient collection of information, but also for effective analysis of such information, as demonstrated by our measurement study. With a large amount of IOCs automatically recovered from the wild and converted into a machine-readable form, their intrinsic relations can be quickly discovered and effectively utilized to counter emerging threats. For example, knowing the sharing of C&C servers across multiple attack instances could enable the defender to disable or block the servers to stop the attacks. On the other hand, our current design is still preliminary. Here, we discuss the limitations of our systems and potential follow-up research.

Error/missing analysis. Our study shows that iACE has a high accuracy and coverage, well beyond what standard NLP techniques can achieve. However, still our technique introduces some false discoveries and misses some IOCs. These problems mostly come from the limitations of underlying tools we use and abnormal ways of presentation. Specifically, Tesseract, the optical character recognizer, is less than perfect, and its accuracy affects the outcome of our analysis. Also, the state-of-the-art dependency parser still cannot maintain its accuracy when sentences become too long. Even though iACE only works on the shortest path between an IOC and its context token, which mitigates the problem, still there are sentences too long for the parser to understand the dependencies between words correctly. Also, adding to the complication are typos: as an example, in the case that one forgets to put a space after the period, a sentence becomes stuck with the follow-up one, which could cause an error in IOC sentence identification. Another interesting observation is that in some articles, authors deliberately misspell URLs to prevent the readers from inadvertently clicking on them: e.g., changing "`http`" to "`hxxp`" or add "[]" among dot in a URL. iACE includes a list of typical obfuscation tricks to recognize such transfers. However, there are always approaches we do not recognize, making IOC tokens fall through the cracks. Furthermore, a large amount of polluted original contents of articles might also lead to false discoveries. For example, an active attacker can compromise the blog websites and inject fake IOCs into the articles, so as to trigger iACE to report false IOCs. Further effort is needed to better address these issues.

Other intelligence sources and standards. The current design of iACE is for gathering threat intelligence from technical blogs, based on the unique ways that IOCs are described. We believe that it will also work well on other equally or more formal sources, such as white papers and other technical articles (e.g., research papers), though further study is certainly needed here. What is less clear is the technique's effectiveness on less formal sources, like technical forums (e.g., Google groups [5], SecurityFocus [7]). The writing styles there are bit different, particularly, the use of more diverse context terms and the sentences with irregular grammatical structures. Extending iACE to this setting needs further effort. Also, the intelligence sources we use to feed iACE are all English articles. Considering the intelligence sources of other languages, iACE should import new modules for language translation of context terms and re-trained dependency parser of different languages. Further, as mentioned earlier, iACE is meant to support the OpenIOC CTI model. Although there are other models such as STIX [8] and yara [12], tools exist to convert the information across these standards.

7. RELATED WORK

Threat intelligence exchange. To help the organizations and security community defend against the fast-evolving cyber attacks, there have been great efforts on threat intelligence sharing. Facebook ThreatExchange [25] and Defense Industrial Base voluntary information sharing program (dibnet) [3] are platforms developed for exchanging IOCs between certified participants. Meanwhile, AlienVault OTX [17], OpenIOC DB [1] and IOC Bucket [29] are established to share public (unclassified) IOCs. Regardless of the type of platform, public sources like blogs still contribute a big portion of IOCs. Our approach, *iACE*, will contribute to the establishment of a fast tunnel between these public sources and exchange platforms and help the participated organizations receive IOC updates timely. As far as we know, AlienVault [17] and Recorded Future [13] are the only two IOC providers that support automatic IOC extraction. Even though Recorded Future (which does not provide public services) utilized NER techniques [40], both tools are simply looking for IOC entities without linking them to attack context, and therefore cannot generate machine-readable OpenIOC items. Instead, *iACE* customized graph mining techniques to capture the relation between entities, which produces high-quality IOCs and their attack context with high accuracy. IOC extraction from other sources were also studied recently. Catakoglu et al. [19] demonstrated a framework to extract external components (web IOCs) from web pages served in honeypots, which compared with our approach, is more in line with the work on automatic signature generation. Sabottke et al. [43] developed a warning system to alert the user of the ongoing attacks reported by tweets (i.e., looking for the tweets with "CVE"). This is different from our work, which aims at generating machine-readable IOCs from attack reports.

NER/RE. NER today mainly relies on a sequence of words to identify the presence of pre-defined entities (such as PERSON, LOCATION). For example, Stanford NER [26] utilize a Hidden Markov model to find the most likely sequence of entities from unstructured text. Other examples include Illinois NER [21] (based on supervised learning) and LIPI [16] (based on n-gram character language models, etc.). When it comes to RE, today's approaches use the tree kernel with SVM [24], heuristic matches with self-supervised learning [46], open pattern templates [44] and other techniques to detect specific relations between two known entities. By comparison, iACE leverages the unique features of IOC-related articles, using the relation detection to help identify true IOCs and their context. This combines both NER and RE steps together, which has never been done before. Our customized graph mining algorithm also enriches the RE techniques.

NLP for security and privacy. Compared with its application in other areas (e.g., bioinformatics), NLP has only been recently used for security and privacy research. Prior work utilized NLP for analyzing web privacy policies (by extracting its key terms) [49], generating privacy policies for Android apps [48], analyzing app descriptions to infer required permissions by Android apps [38, 36], detecting compromised websites [31] and identifying sensitive user input from apps [28, 34]. Our work showcases a new application of NLP, demonstrating that innovative NLP techniques need to be developed to address real-world security challenges.

8. CONCLUSION

In this paper, we present *iACE*, a novel technique for automatic extraction of IOCs from unstructured text. iACE is designed to specialize NLP techniques to threat intelligence gathering, combining the NER and RE steps together based on the unique features of IOCs and the technical articles describing them. By anchoring a sentence with putative IOC tokens and context terms, our approach can efficiently validate the correctness of these elements using their relations, through a novel application of graph similarity comparison. This simple technique is found to be highly effective, vastly outperforming the top-of-the-line industry IOC analyzer and NER tool in terms of precision and coverage. Our evaluation of over 71,000 articles released in the past 13 years further reveals intrinsic connections across hundreds of seemingly unrelated attack instances and the impacts of open-source IOCs on the defense against emerging threats, which highlights the significance of this first step toward fully automated cyber threat intelligence gathering.

9. ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation (grants CNS-1223477, 1223495, 1527141 and 1618493). We thank our anonymous reviewers for their useful comments.

10. REFERENCES

[1] A community OpenIOC resource. https://openiocdb.com/.
[2] Beautiful Soup. https://www.crummy.com/software/BeautifulSoup/.
[3] Defense Industrial Base Cybersecurity Information Sharing Program. http://dibnet.dod.mil/.
[4] GIMP. https://www.gimp.org/downloads/.
[5] Google groups. https://groups.google.com/.
[6] Open Source OCR Engine. https://github.com/tesseract-ocr/tesseract.
[7] SecurityFocus. www.securityfocus.com/.
[8] Structured Threat Information eXpression. https://stixproject.github.io.
[9] The OpenIOC Framework. http://www.openioc.org.
[10] topia.termextract 1.1.0. https://pypi.python.org/pypi/topia.termextract.
[11] Virustotal. https://www.virustotal.com/.
[12] YARA. http://plusvic.github.io/yara/.
[13] Real-Time Threat Intelligence. https://www.recordedfuture.com/, 2016.
[14] Semantic Link: find related words. http://semantic-link.com/, 2016.
[15] Abiword. Enchant. http://www.abisource.com/projects/enchant/, 2010.
[16] Alias-i. Lingpipe 4.1.0. http://alias-i.com/lingpipe, 2008.
[17] AlienVault. Open Threat Intelligence. https://otx.alienvault.com/, 2016.
[18] N. Bach and S. Badaskar. A review of relation extraction. *Literature review for Language and Statistics II*, 2007.
[19] O. Catakoglu, M. Balduzzi, and D. Balzarotti. Automatic extraction of indicators of compromise for web applications. In *WWW 2016*, 2016.
[20] D. Chen and C. D. Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750, 2014.
[21] J. Clarke, V. Srikumar, M. Sammons, and D. Roth. An nlp curator (or: How i learned to stop worrying and love nlp pipelines). In *LREC*, 5 2012.
[22] CleanMX. http://lists.clean-mx.com/cgi-bin/mailman/listinfo/viruswatch/.
[23] D. J. Cook and L. B. Holder. *Mining graph data*. John Wiley & Sons, 2006.
[24] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of ACL'04*.
[25] Facebook. https://developers.facebook.com/products/threat-exchange.
[26] J. R. Finkel, T. Grenager, et al. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL'05*.
[27] T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*. 2003.
[28] J. Huang, Z. Li, and X. Xiao. Supor: Precise and scalable sensitive user input detection for android apps. In *USENIX Security'15*.
[29] IOCbucket. IOCbucket. https://www.iocbucket.com/, 2016.
[30] Josh Grunzweig. Alina: Casting a Shadow on POS. https://www.trustwave.com/Resources/SpiderLabs-Blog/Alina--Casting-a-Shadow-on-POS/, 2013.
[31] X. Liao, K. Yuan, X. Wang, et al. Seeking nonsense, looking for trouble: Efficient promotional-infection detection through semantic inconsistency search. In *Proceedings of S&P'16*.
[32] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. Association for Computational Linguistics, 2004.
[33] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.
[34] Y. Nan, M. Yang, Z. Yang, et al. Uipicker: User-input privacy identification in mobile applications. In *USENIX Security'15*.
[35] L. Obrst, P. Chase, and R. Markeloff. Developing an ontology of the cyber security domain. In *STIDS*, pages 49–56, 2012.
[36] R. Pandita, X. Xiao, et al. Whyper: Towards automating risk assessment of mobile applications. In *USENIX Security'13*.
[37] PhishTank. https://www.phishtank.com/.
[38] Z. Qu, V. Rastogi, and X. Zhang. Autocog: Measuring the description-to-permission fidelity in android applications. CCS '14.
[39] J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *First international workshop on mining graphs, trees and sequences*, pages 65–74. Citeseer, 2003.
[40] Recorded Future. http://info.recordedfuture.com/Portals/252628/resources/cyber-anatomy-white-paper.pdf.
[41] Recorded Future. Recorded Future at SITA. https://go.recordedfuture.com/hs-fs/hub/252628/file-2607572540-pdf/case-studies/sita.pdf, 2015.
[42] Rob McMillan. Open Threat Intelligence. https://www.gartner.com/doc/2487216/definition-threat-intelligence, 2013.
[43] C. Sabottke, O. Suciu, et al. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *USENIX Security'15*.
[44] M. Schmitz, R. Bart, S. Soderland, et al. Open language learning for information extraction. In *Proceedings of the JCEMNLP'12*.
[45] B. Settles. Abner: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005.
[46] F. Wu and D. S. Weld. Open information extraction using wikipedia. In *Proceedings of ACL'10*.
[47] L. Yi, B. Liu, and X. Li. Eliminating noisy information in web pages for data mining. In *Proceedings of ACM SIGKDD'03*.
[48] L. Yu, T. Zhang, X. Luo, and L. Xue. Autoppg: Towards automatic generation of privacy policy for android applications. SPSM '15, 2015.
[49] S. Zimmeck and S. M. Bellovin. Privee: An architecture for automatically analyzing web privacy policies. In *USENIX Security 14*, 2014.