A System for Automated Security Knowledge Extraction

Edward Choi

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-141
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-141.html

May 18, 2022

A System for Automated Security Knowledge Extraction

by Edward Choi

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Dawn Song
Research Advisor

05 / 11 / 2022

(Date)

* * * * * * *

Professor David Wagner
Second Reader

05 / 18 / 2022

(Date)

A System for Automated Security Knowledge Extraction

by

Edward Choi

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Dawn Song, Co-chair
Professor Peng Gao (Virginia Tech), Co-chair
Professor David Wagner

Spring 2022

A System for Automated Security Knowledge Extraction

Abstract

A System for Automated Security Knowledge Extraction

by

Edward Choi

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Dawn Song, Co-chair

Professor Peng Gao (Virginia Tech), Co-chair

Complex cyber attacks have highly impacted many high-profile businesses. To remain aware of the fast-evolving threat landscape, open-source Cyber Threat Intelligence (OSCTI) has received growing attention from the community. Commonly, knowledge about threats is presented in a vast number of OSCTI reports. Despite the pressing need for high-quality OSCTI, current OSCTI management systems have primarily focused on isolated, low-level Indicators of Compromise (IOC). On the other hand, higher-level concepts (e.g., adversary tactics, techniques, and procedures) and their relationships have been overlooked, which contain crucial knowledge about threat behaviors that is critical to uncovering the complete threat scenario. To bridge the gap, we propose ThreatExtractor, a system for automated security knowledge extraction. In particular, ThreatExtractor automatically collects a large number of OSCTI reports from a variety of sources, uses a combination of AI and NLP techniques to extract high-fidelity knowledge about threat behaviors, and uses this knowledge in the form of entities and relations to construct a security knowledge graph. ThreatExtractor also provides a GUI that supports various types of interactivity to facilitate knowledge graph exploration.

Contents

List of Figures

List of Tables

Acknowledgments

Doing research for the past three years has been an eye-opening and thrilling experience. I am grateful for Professor Dawn Song for advising me as an undergraduate and master's student. Her words of advice and guidance have been invaluable in my growth as a researcher.

I am amazingly lucky to have been mentored by Professor Peng Gao at Virginia Tech, who helped me start out my research journey. I am grateful for his support, commitment, and passion in guiding me these past few years and helping me through some of the toughest research challenges. His ability to do top-tier research while still being able to provide support and direction for his students is awe-inspiring. It has been an amazing journey and I wouldn't have been able to go this far without him.

I would also like to thank Professor David Wagner, for being my second reader on this thesis and giving constructive and detailed feedback.

I would like to thank my other collaborators, Xiaoyuan Liu, Sibo Ma, and Zhengjie Ji, who I have gotten learn from immensely these past couple of years.

Thank you to all of the friends I've made during my time at Berkeley. I wouldn't have made it through without their unwavering support.

Finally, I would like to thank my family for their unconditional love and support throughout my educational journey.

Chapter 1

Introduction

Complex cyber attacks have greatly impacted many high-profile businesses [11, 1, 3, 4, 8, 5]. As a way to gain visibility and insights into the fast-evolving threat landscape, open-source cyber threat intelligence (OSCTI) has garnered increasing attention from the security community. In particular, knowledge about threats is presented in a massive number of OSCTI reports in various forms (e.g., threat reports, security news and articles [23, 21]. Despite the pressing need for high-quality OSCTI, existing OSCTI management systems [26, 10, 13], however, have primarily focused on simple Indicators of Compromise (IOCs) [50] such as malicious file/process names, IP addresses, dmoain names, and signature of artifacts. Though effective in capturing isolated, low-level IOCs, these systems cannot capture higher-level threat behaviors such as adversary tactics, techniques, and procedures [19], which are aligned with the adversary's goals and thus much harder to change. Thus, IOCs lack the capability of uncovering the complete threat scenario and can only capture partial views of threats. To this end, there is a need for extraction of high-fidelity threat intelligence in an automated way.

However, currently there are several problems in extracting security knowledge from unstructured OSCTI reports. First, OSCTI reports are written in natural language texts, which means they are not suited for automated analysis. Second, the natural language text written in OSCTI reports also often has complex logical structures and deep semantics which may be hard to understand well enough for accurate threat extraction. Third, individual OSCTI reports also often only cover partial information about threat behaviors [49, 56], and so it is important to combine knowledge from multiple reports to get a complete view of threat behaviors.

1.1 Goals & Challenges

In this work, we design automated techniques to both extract comprehensive threat knowledge from a large number of unstructured OSCTI reports from a wide range of OSCTI sources and integrate the knowledge into a unified knowledge base that is able to provide a

complete view of threats.

Based on these goals, we identify four challenges for building such a system:

1. Other than IOCs, OSCTI reports also contain other types of knowledge entities and relations that capture threat behaviors. Thus, the system must cover a wide range of knowledge entity and relation types that can comprehensively model threats.

2. OSCTI reports come in diverse formats including reports that contain structured fields such as lists and tables (e.g., threat encyclopedia reports [27]) and reports that consist of unstructured natural language text (e.g., security blogs [63]). The system should be able to handle diverse report formats.

3. Threat reports from an OSCTI source may be irrelevant to threats (e.g., reports that advertise security products of the company which maintains the OSCTI source [50]). The system should be able to filter out these irrelevant reports.

4. Accurate extraction of threat knowledge natural language text is nontrivial because of massive nuances specific to the security context such as special characters (e.g., periods, underscores) in IOCs. These nuances limit the performance of existing natural language processing (NLP) modules (e.g., sentence segmentation, tokenization) and make current information extraction tools [28, 9] ineffective. In addition, some learning-based information extraction approaches require large annotated corpus for model training, which is expensive to obtain manually. Thus, the steps involved with programmatically obtaining annotations becomes another challenge.

1.2 Contributions

We propose ThreatExtractor, a system for automated open-source cyber security knowledge extraction. As part the system, ThreatExtractor collects a large number of OSCTI reports from various sources, uses a combination of AI and NLP based techniques to filter out irrelevant reports and extract high-fidelity threat knowledge as entities and relations, and uses the knowledge to construct a *security knowledge graph* based on a security knowledge ontology. In particular, to address the previously mentioned challenges, we make the following contributions:

1. *Fast & Robust OSCTI Report Crawlers:* We build and deploy a robust multi-threaded crawler framework that manages 40 brawlers for collecting OSCTI reports from major security websites including threat encyclopedias [27, 16], enterprise security blogs [63, 23, 14], influential personal security blogs [22, 17], security news [25, 24], etc.

2. *Security Knowledge Ontology:* The construction of the security knowledge graph follows a pre-defined security knowledge ontology to model cyber threats from multiple

dimensions. Our ontology covers both low-level entities such as IOCs (e.g., name/-path, IP, URL, domain) which provides details on threat behavior and high-level entities such as malware, vulnerabilities, threat actors, techniques, software, and security-related tools to provide a high-level threat context. Entities can have various types of relations (e.g., <ACTOR_A, *use*, MALWARE_A>, <MALWARE_A, *drop*, FILE_A>, <ACTOR_A, *reported_in*, REPORT_A>). Entities can also have attributes in the form of key-value pairs (e.g., type of a "MALWARE" entity). In comparison to other cyber ontologies (e.g., STUCCO [43], STIX [12]), our ontology covers a larger set of entities and relations (see Section 2.4).

3. *Parsers & Checkers:* For each OSCTI source, ThreatExtractor maintains a source-dependent parser which parses the structured fields (e.g., title, author, tables, lists, text fields) (see Section 2.2) to handle diverse formats of OSCTI reports (e.g., different HTML layouts). ThreatExtractor also maintains a set of rule-based and learning-based checkers to screen out OSCTI reports that are irrelevant to cyber threats.

4. *Rule-Based and Deep Learning-Based Techniques for Security Knowledge Extraction:* After the OSCTI report screening process, to extract further knowledge from the parsed unstructured texts, ThreatExtractor maintains a set of rule-based and AI-based extractors to extract a variety of security-related entities and relations. To accurately these entities and relations from natural language (NL) OSCTI texts, ThreatExtractor employs a combination of rule-based and deep learning-based techniques in its extractors. Specifically, to extract IOCs, ThreatExtractor has a rule-based IOC extractor that leverages a set of regular expression rules. Then to extract other types of entities, as they are hard to extract using fixed rules, ThreatExtractor employs a deep learning-based entity extractor that leverages a Bidirectional LSTM-CRF (BiLSTM-CRF) [46] model for neural named entity recognition. To annotate a large training corpus for the deep learning model, we leverage data programming [59] to programmatically synthesize annotations for targeted entities in natural language text in OSCTI reports. To extract relations, ThreatExtractor uses a dependency parsing-based relation extractor.

   During the IOC extraction process, we handle the nuances within the IOCs through pre-processing the texts by replacing IOCs with dummy words in natural language context (e.g., word "FILE" for a file IOC token), and restoring them after the tokenization procedure. This way we guarantee that the potential entities are complete tokens.

In addition to the automated OSCTI report collection and threat extraction techniques, ThreatExtractor also constructs a security knowledge graph containing the threat knowledge in the form of entities and relations. Then on top of the constructed security knowledge graph, ThreatExtractor provides a web GUI that provides various types of interactivity to facilitate cyber threat knowledge exploration and acquisition (see Section 2.6). The system proposed in this thesis was published and presented at SIGMOD 2021 Demonstrations Track [36].

At the time of writing, ThreatExtractor has collected 149K+ OSCTI reports from 40+ major OSCTI sources including websites for threat encyclopedias [27, 16], security articles and blogs [7, 23, 14, 22, 17], security news [25, 24], etc. (see Section 2.1). The constructed security knowledge graph contains 347K+ entities and 1.73M+ relations.

1.3 Evaluation

We deployed ThreatExtractor on a physical testbed and evaluated its effectiveness. The evaluation results show that:

1. Our OSCTI report checkers are able to effectively filter out non-threat-related reports with an average 89.69% F1 and 84.27% accuracy.

2. Our threat knowledge entity extraction model based on BiLSTM-CRF is able to effectively extract a variety of entities from unstructured OSCTI texts and can generalize to an unseen OSCTI source with 99.95% accuracy and 99.94% F1.

3. Using ThreatExtractor's web GUI, our case studies on the wannacry ransomware and the cozyduke threat actor show that it is easy to search for and acquire threat knowledge.

Different from existing threat knowledge extraction techniques [68, 42, 61, 37], ThreatExtractor targets automatic threat knowledge extraction from OSCTI reports by using a combination of AI and NLP techniques. ThreatExtractor also represents the extracted threat knowledge which is in the form of entities and relations in a security knowledge graph and provides a convenient web GUI for cyber threat knowledge graph exploration and acquisition.

Chapter 2

System Design

In this chapter, we present the system design for ThreatExtractor. We first present the OSCTI report collection process, .

2.1 OSCTI Reports Collection

We built a robust multi-threaded crawler framework that manages 40 crawlers for continuously collecting OSCTI reports from major security websites (each crawler collects HTML files from one website), including: threat encyclopedias [27, 16], enterprise security blogs [63, 23, 14], influential personal security blogs [22, 17], security news [25, 24], etc. Table 2.1 shows the complete list of websites that ThreatExtractor currently covers.

As each website has a diverse format, the crawlers are capable of handling both static and dynamic web pages:

- *Static web pages*: Crawlers for OSCTI sources that have static web pages extract href links (using the Python BeautifulSoup4 library) that route to individual report URLs, which their HTML contents are then downloaded.
- *Dynamic web pages*: Crawlers for OSCTI sources with dynamic web pages requires user interaction in order to load more content and gather href links (e.g., pressing the "View More" button [63]. Thus, these crawlers leverage the Python Selenium library and the Chrome Webdriver to simulate a headless browser environment, which uses query selectors to locate the button and execute simulated clicks onto it to load more content.

Because the security websites' HTML templates can change unexpectedly, we also periodically monitor the crawlers and ensure they are most up to date.

Due to the sheer number of crawlers to manage and the multitude of OSCTI reports to handle, the crawler framework has a robust multi-threaded task scheduler to schedule the parallel execution for (1) running multiple crawlers, and (2) fetching multiple OSCTI reports

Table 2.1: OSCTI Report Source Names with Number of Reports and URLs

Source	Number of Reports	URL
apt_notes	539	https://github.com/aptnotes/data
attcybersecurity	244	https://cybersecurity.att.com
ciscoumbrella	478	https://umbrella.cisco.com
cloudflare	1,791	https://blog.cloudflare.com
crowdstrike	942	https://www.crowdstrike.com
csoonline	1,258	https://www.csoonline.com/
darknet	2,107	https://www.darknet.org.uk
fireeye	209	https://www.fireeye.com
forcepoint	1,190	https://www.forcepoint.com
hotforsecurity	9,496	https://hotforsecurity.bitdefender.com
kasperskydaily	3,350	https://www.kaspersky.com
krebsonsecurity	2,129	https://krebsonsecurity.com
malwarebytes	3,382	https://blog.malwarebytes.com
mcafee	6,295	https://www.mcafee.com
nakedsecurity	14,653	https://nakedsecurity.sophos.com
nccgroup	520	https://research.nccgroup.com
paloalto	3,284	https://blog.paloaltonetworks.com
recordedfuture	1,537	https://www.recordedfuture.com
rsa	71	https://www.rsa.com
securelist	5,630	https://securelist.com
shneieronsecurity	8,110	https://www.schneier.com
sophos	1,822	https://news.sophos.com
spiderlabs	1,401	https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/
symantecthreatintelligence	177	https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/
thehackernews	8,432	https://thehackernews.com
threatpost	5,427	https://threatpost.com/
trendmicro	2,393	https://blog.trendmicro.com
trendmicrosecurityintelligence	4,001	https://blog.trendmicro.com/trendlabs-security-intelligence
trustwave	571	https://www.trustwave.com/en-us/resources/blogs/trustwave-blog/
unit42_paloalto	645	https://unit42.paloaltonetworks.com/
webroot	1,438	https://www.webroot.com
welivesecurity	5,780	https://www.welivesecurity.com
zscaler	770	https://www.zscaler.com
malwarebytes	163	https://blog.malwarebytes.com
symantec_threats	37,588	http://asb-sngweb.symantec.com
symantec_vulnerabilities	7,431	http://asb-sngweb.symantec.com
kaspersky_threat	1,430	https://threats.kaspersky.com/en
kaspersky_vulnerability	1,968	https://threats.kaspersky.com/en
trendmicro_malware	534	https://www.trendmicro.com
trendmicro_spam	396	https://www.trendmicro.com
fsecure	4,083	https://www.f-secure.com

for each individual crawler. The scheduler is easily customizable allowing the user to specify the desired number of threads, and after the scheduler starts, each thread is assigned a task to execute. After the current batch of tasks are finished, the crawler framework will remain idle and schedule the execution of the crawlers again after a specified period of time (e.g., every 24 hours). The crawler framework is also resilient to failures (e.g., due to timeout or connection refusal), and crawlers that suffer failures will be scheduled for reboot. In addition, the framework also provides isolation because the underlying scheduler performs exception handling within each thread, meaning a single crawler's failure will not affect other crawlers. The crawler framework is able to achieve a throughput of approximately 350+ reports per minute on a single deployed host. At the time of writing, the crawler framework has collected $149,015$ reports in total so far.

2.2 Parsers and Checkers

Parsers. After the crawlers collect the OSCTI reports, we take them and convert them into *intermediate report representations* by grouping multi-page reports and adding metadata like ids, sources, titles, files locations, and timestamps. An example is shown in Figure 2.1.


```
{
  "payload":
"PCFET0NUWVBFIEhUTUw+CjwhLS0gUlNBLXd3dyBCYXNlIChCYXNlIFBhZ2UpOiBkYXRhLWNvbXBvbmVudC12ZXJza
W9uPSIxLjEwIiAtLT4KPGh0bWwgbGFuZz0iZW4tdXMiPgo8aGVhZD4KPCEtLSBSU0Etd3d3IGhlYWQgKEJhc2UgUGFnZ
Sk6IGRhdGEtY29tcG9uZW50LXZlcnNpb249IjEuMjEiIC0tPgo8dGl0bGU+SW50ZXJwcmV0YWJpbGl0eSBvZiBNYWNoa
W5lIExlYXJuaW5nIE1vZGVscyBmb3IgRnJhdWQgRGV0ZWN0aW9uPC90aXRsZT4KPG1ldGEgY2hhcnNldD0iVVRGLTgiLz4KPG1ldGEgaHR0cC1lcXVpdj0iWC1VQS
1Db21wYXRpYmxlIiBjb250ZW50PSJJRT1lZGdlIi8+CjxtZXRhIG5hbWU9InZpZXdwb3J0IiBjb250ZW50PSJ3aWR0aD1kZX
ZpY2Utd2lkdGgsIGluaXRpYWwtc2NhbGU9MS4wLCBtaW5pbXVtLXNjYWxlPTEiLz4KPG1ldGEgaHR0cC1lcXVpdj0iY2xlY
XJ0eXBlIiBjb...",
  "title": "cti_reports/rsa_html/2020-08:interpretability-of-machine-learning-models-for-fraud-detection",
  "source": "rsa",
  "create_on": "2022-04-26T17:51:47.992323",
  "remarks": {},
  "id": "eaab2fe7c6614b6aa051c74df281db46",
  "type": "HTML"
}
```


Figure 2.1: Example Intermediate Report Representation

We then have parsers that take advantage of prior knowledge of the OSCTI source structure to extract keys and values from report files. They are responsible for converting the list of intermediate report representations into a list of *intermediate security knowledge representations* (which we showed an example in Figure 2.2).

OSCTI Report Checkers. After the list of intermediate security knowledge representations is built, we run checkers on it which works as a filter by screening out reports that are irrelevant to cyber threats (e.g., empty pages or ads) through conditional checks. As the crawlers only download HTML report files from the gathered URLs regardless of their contents, there could be reports that do not contribute valuable information to cyber threats (e.g., empty pages, ads, product promotions, irrelevant news). Keeping these OSCTI reports out of the processing pipeline saves on computational and storage resources for the downstream extraction and storage tasks.

We introduce two types of checkers; namely a rule-based checker and a learning-based checker:

1. The rule-based checker screens out empty or error HTML pages.

```
{
  "name": "cti_reports/rsa_html/2020-09:big-c-or-little-c-the-c-in-grc-quantifying-risk-compliance",
  "source_report_id": "27a5f66e541148d38cd51b8bbcf2e777",
  "source": "rsa",
  "source_url": "https://www.rsa.com/",
  "summary": null,
  "url": "https://www.rsa.com/en-us/blog/...",
  "author": "Chris Patteson",
  "publisher": null,
  "discovered_date": "Sep 29, 2020",
  "updated_date": "Sep 29, 2020",
  "ref_cve": null,
  "remarks": "{...}",
  "related_file_names": "[...]",
  "related_file_paths": "[...]",
  "related_ips": "[...]",
  "related_badactor": "[...]",
  "related_malware": "[...]",
  "related_technique": "[...]",
  "related_tool": "[...]",
  "related_mitigation": "[...]",
  "ioc_relations": "[...]",
  "relations": "[...]",
  "affected_systems": "[...]",
  "alias": null,
  "description_summary": null,
  "description": "\n\nImage: data-component-version=\"1.9\"  \n\n\n\n\n\nBlog Text: data-component-version=\"1.1\"  \n\nWhen talking about the “C” in GRC there is a big C and a little c. The big C is your compliance program, the little “c” is just another risk. How do you quantify the risk of being non-compliant? The stakes can be pretty high. \nTo kick off the conversation,...",
  "description_html": "<div class=\"grid-right parsys\">\n<div>\n<!-- Image: data-component-version=\"1.9\" -->\n<div class=\"c00 c00v0\">\n<img alt=\"Man looking at monitor with RSA Archer GRC tool on screen\" src=\"/content/dam/blog/2020-09/blog-image-big-c-or-little-c.jpg\" title=\"Big C or little c – The C in GRC, Quantifying Risk Compliance\"/>\n</div>\n</div>\n<div>...",
  "extracted": "[...]",
  "id": "6ad3f7d4252e4693b55843baa373c458",
  "blog_title": "Big C or little c – The C in GRC, Quantifying Risk Compliance",
  "blog_subtitle": "",
  "categories": [],
  "tags": [
    "Cyber Risk Quantification",
    "Risk & Compliance (GRC)",
    "IRM"
  ],
  "related_blog_titles": [],
  "author_about": null,
  "type": "BLOG"
}
```


Figure 2.2: Example Intermediate Security Knowledge Representation

2. The learning-based checker screens out ads and other irrelevant reports by performing binary classification to predict whether a given report is relevant to cyber threats or not. To train the classifier, we extract a set of useful features from a given report,

including:

- *Keyword Count & Density in the Report Body:* We obtain the list of keywords from MITRE ATT&CK [19] for examples from important threat-related categories such as threat actor, malware, tool, and technique.
- *Keyword Count & Density in the Report Title:* We scan the same list of keywords in the report title, as the title typically reflects the overall theme of the report.
- *IOC Count & Density in the Report Body:* The IOCs are extracted using regex rules (see Table 2.2). We didn't compute these metrics for the report title as the title typically does not contain IOC details.
- *Report Article Length:* Based on our observations, a longer report is more likely to contain threat contexts and threat behavior details (e.g., indicators).
- *TF-IDF Values for Tokens:* We prioritize the frequent, unique tokens in the training report corpus by calculating the TF-IDF [58] value for each token in the report.

Table 2.2: Representative IOC regex rules

IOC Type	**Regex**
Windows Filepath	\b[A−Z]:\\[A−Za−z0−9−_\.\\]+\b
Linux Filepath	\b(/[^/]∗[^/.,() %%\n]+)+/?\b
URL	\b([a−z]{3,}\:\/\/[\S]{16,})\b
IP	\b(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}) (:\d{1,5})?\b
Email	\b([a−z][_a−z0−9−.]+@[a−z0−9−]+\.[a −z]+)\b
MD5	\b([a−f0−9]{32}|[A−F0−9]{32})\b
SHA1	\b([a−f0−9]{40}|[A−F0−9]{40})\b
SHA256	\b([a−f0−9]{64}|[A−F0−9]{64})\b
CVE	\b(CVE\−[0−9]{4}−[0−9]{4,6})\b
Registry	\b((HKLM|HKCU)\\[\\A−Za−z0−9−_]+)\b
Filename	\b([A−Za−z0−9−_\.]+\.(EXE|exe|dll| bat|sys|htm|html|js|jar|jpg|png|vb|scr|pif| chm|zip|rar|cab|pdf|doc|docx|ppt|pptx|xls| xlsx|xlsm|swf| gif |txt|ps1|so|apk))\b

OSCTI reports collected from different sources have different structures, writing styles, and focused topics. Thus, considering the distributional shift in the training data, a classifier might benefit more from data within the same source compared with other sources. Our evaluation results in Section 3.2.1 validate this for a variety of machine learning models, including: Logistic Regression, Random Forest, Linear SVM, SVM with RBF Kernel, XGBoost [32], and LightGBM [44].

2.3 Security Knowledge Extraction

The ThreatExtractor extraction process involves further refining the intermediate security knowledge representations by extracting information (e.g., IOCs, malware names) using entity recognition and relation extraction and putting them into the corresponding fields.

2.3.1 Security-related Entity Recognition

Table 2.3: Entity tags for NER in IOB2 format

Tags	Description
B-BADACTOR	Beginning of a BADACTOR entity
I-BADACTOR	Part of BADACTOR entity
B-MALWARE	Beginning of a MALWARE entity
I-MALWARE	Part of MALWARE entity
B-TOOL	Beginning of a TOOL entity
I-TOOL	Part of TOOL entity
B-TECHNIQUE	Beginning of a TECHNIQUE entity
I-TECHNIQUE	Part of TECHNIQUE entity
B-MITIGATION	Beginning of a MITIGATION entity
I-MITIGATION	Part of MITIGATION entity
O	Non-targeted entity

ThreatExtractor has a rule-based IOC extractor that utilizes a set of regular expression rules (shown in Table 2.2) to extract IOC entities. During the IOC extraction process, we handle the nuances within the IOCs through pre-processing the texts by replacing IOCs with dummy words in natural language context (e.g., word "FILE" for a file IOC token), and restoring them after the tokenization procedure. This way we guarantee that the potential entities are complete tokens.

Since other types of entities are hard to extract using fixed rules, ThreatExtractor also uses a deep learning-based entity extractor that runs separately from the rule-based

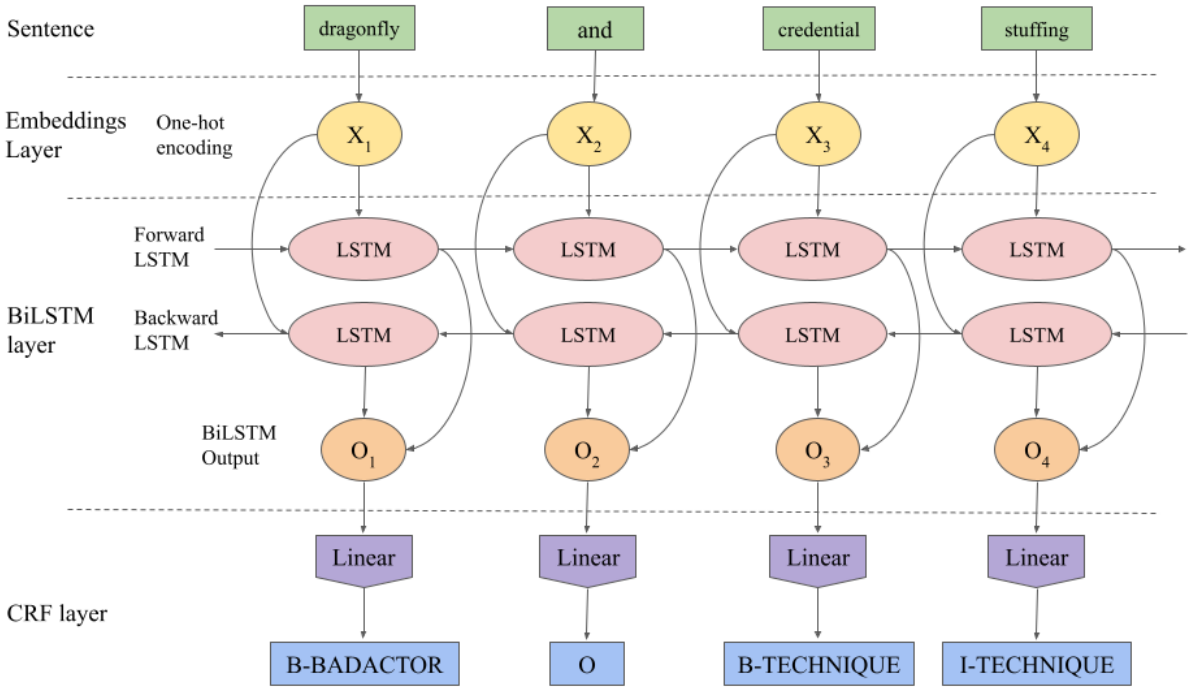

Figure 2.3: BiLSTM-CRF for security-related entity recognition

IOC extractor. Specifically, we construct a Bidirectional LSTM-CRF (BiLSTM-CRF) model architecture [46] to perform neural named entity recognition (NER) over natural language OSCTI texts. Compared to conventional NER approaches like HMM [67] and CRF [45], deep learning-based approaches avoid the time-consuming feature engineering stage and can better understand deep semantics of OSCTI texts and capture hidden patterns, leading to more accurate entity recognition.

The BiLSTM-CRF model takes as input sentences, which are broken down into word tokens, from OSCTI report texts, and outputs an entity tag for each word token. The entity tags are based on the IOB2 format, which assigns tags using three types of prefixes: (i) B- prefix, used for a token in the beginning of an entity chunk, (ii) I- prefix, used for a token inside a chunk, and (iii) O- prefix, used for a token outside a chunk. The set of entity tags that we use for the BiLSTM-CRF model are shown in Table 2.3.

Figure 2.3 shows our BiLSTM-CRF model architecture, which consists of the following layers:

1. *Embedding Layer*: Sentences come in as initial inputs to the model. Then, each sentence is broken down into tokens (words) via tokenization and each token is then transformed into an embedding vector using one-hot encoding.

2. *BiLSTM Layer*: The embeddings are then forwarded as input to the bidirectional LSTM (BiLSTM) layer. A BiLSTM is a sequence processing model that consists of two Long-Short Term Memory (LSTM) networks: one taking the input in the forward direction, and the other in the backward direction. An LSTM [41] is a variant of recurrent neural networks that combats the gradient vanishing and exploding problems, and is better at capturing long-term dependencies of tokens in the sequence. A single LSTM network can only remember and process information from the past context. However, for tasks like NER which require understanding the context of a word (token) through past and future contexts, an additional LSTM network is needed. In BiLSTM, the first LSTM processes a sequence of tokens in the forward direction and induces a representation of tokens in the past context, while the second LSTM processes the sequence in the backward direction and induces a representation of tokens in the future context. The BiLSTM network is able to understand the information provided to it in a bidirectional manner, which improves the contextual relationship of tokens. In essence, the BiLSTM layer extracts semantic features from the sequence and provides the features to the CRF model for the NER task.

3. *CRF Layer*: The output vectors from the BiLSTM layer are forwarded as input to a linear layer, which maps the features extracted by the BiLSTM from feature space into tag space. After the mapping, the output is sent to the Conditional Random Field (CRF) model which decodes the input sequence and outputs the tag predictions. Essentially, the CRF model is responsible for finding the most optimal sequence path for our tags, which does so by maximizing the probability of seeing the given tag at every state. Thus, we use the Viterbi algorithm [35] to decode the input sequence and calculate the tag predictions.

As part of the training process for the BiLSTM-CRF model, we need to label the OSCTI text corpus first. Specifically, we use the IOB2 format [60] to assign entity tags to every token in the corpus.

A major challenge for the training process is the need to annotate a large training corpora. Thus, to address this challenge, we programmatically synthesis annotations using data programming [59]. In particular, we create labeling functions that are based on our curated lists of entity names. For example, the list of threat actors, malware, techniques, and tools are constructed from MITRE ATT&CK [19].

When training the BiLSTM-CRF model, we tuned a set of hyperparameters, including: epochs, batch size, maximum report length, hidden units, embedding dimension, LSTM layers, and learning rate.

2.3.2 Security-related Relation Extraction

To extract relations between entities recognized by our BiLSTM-CRF model, since it is relatively difficult to synthesize annotations for relations, we take an unsupervised approach.

In particular, we leverage the IOC relation extraction approach proposed in [37], which uses dependency parsing to analyze the grammatical structure of a sentence and constructs a dependency tree and then uses a set of dependency grammar rules to locate the subject-verb-object relations between IOCs to finally extract the relation verb. According to [37], this approach has achieved high extraction accuracy on a large OSCTI corpus. We further extend [37] to support the extraction of relations between IOCs and other entities recognized by our BiLSTM-CRF model (e.g., <MALWARE_A, *drop*, FILE_A>, <ACTOR_A, *use*, MALWARE_A>, <ACTOR_A).

2.4 Security Knowledge Ontology

We use the high-fidelity security knowledge represented by entities and relations found through entity recognition and relation extraction to build a security knowledge graph containing the entity-relation triplets. However, before we construct the security knowledge graph, we design a security knowledge ontology that is able to model the wide range of high-level and low-level threat entities and relations. Figure 2.4 shows our security knowledge ontology.

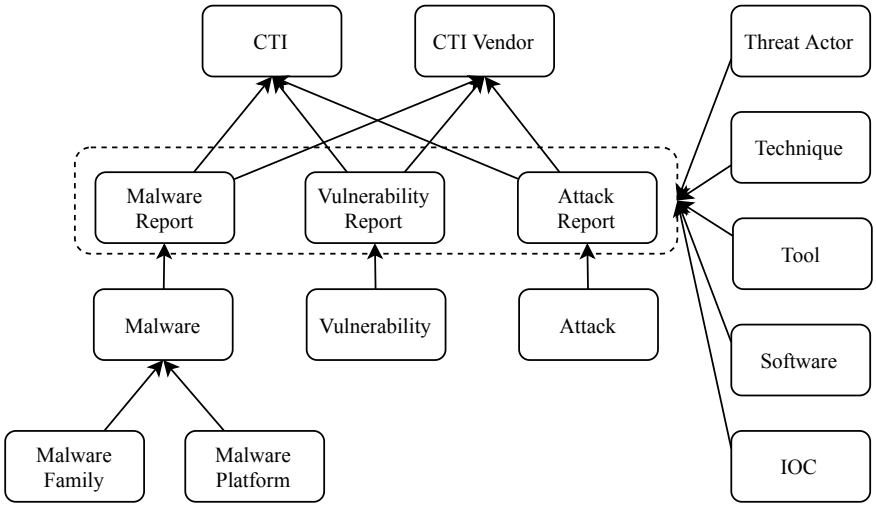

Figure 2.4: Security knowledge ontology

Based on our observations of various OSCTI data sources, we classify OSCTI reports into three types: malware reports, vulnerability reports, and attack reports. These reports are created by specific CTI vendors and contain information including:

1. malware (e.g., "BlackEnergy" Trojan [15])
2. vulnerabilities (e.g., CVEs)

3. threat actors (e.g., “CozyDuke” group [6])
4. techniques (e.g., “credential stuffing” [19])
5. vulnerable software products (e.g., “Microsoft Word”)
6. security-related tools (e.g., “Mimikatz”)
7. IOCs (e.g., file name, file path, IP , URL, email domain, registry, hashes)

We create entities out of these concepts as well. Entities have relationships between them (e.g., <ACTOR_A, *use*, MALWARE_A> describes a "use" relationship between an "ACTOR" entity and a "MALWARE" entity). Entities also have attributes in the form of key-value pairs. By using such a security knowledge ontology, we are able to capture different types of security knowledge in the system.

Compared to other cyber ontologies (e.g., STIX [12], STUCCO [43]), our ontology covers a larger set of entities and relations.

2.5 Security Knowledge Graph Construction

After the completion of the security knowledge extraction process, ThreatExtractor constructs the security knowledge graph from the entity-relation triplets. ThreatExtractor stores the security knowledge graph in the database backend through database connectors to persist the threat knowledge. The connector merges the intermediate security knowledge representations, which is refined with the extracted entities and relations, into the corresponding storage backend by refactoring them to match our security knowledge ontology. Currently, ThreatExtractor uses a Neo4j [20] database for its storage, with nodes being entities and edges being relations. Each node is associated with a category (e.g., malware or threat actor), a unique name (e.g., specific malware or threat actor name) and a collection of attributes.

2.6 Frontend Web GUI

To facilitate security knowledge graph exploration and threat knowledge acquisition, we built a web GUI using React and Elasticsearch. Figure 2.5 shows an example subgraph of security knowledge graph in the GUI. Currently, the GUI interacts with the Neo4j database through a Neo4j JS driver, and provides various interactivity, which we describe next.

We designed features to simplify the user view. The user can zoom in/out and pan the canvas. Node names and edge types are displayed by default for convenient threat identification. Nodes are also colored according to their labels (e.g., IOC, malware, threat actor). When a node is hovered over, its detailed information is displayed (e.g., URL of a report node).

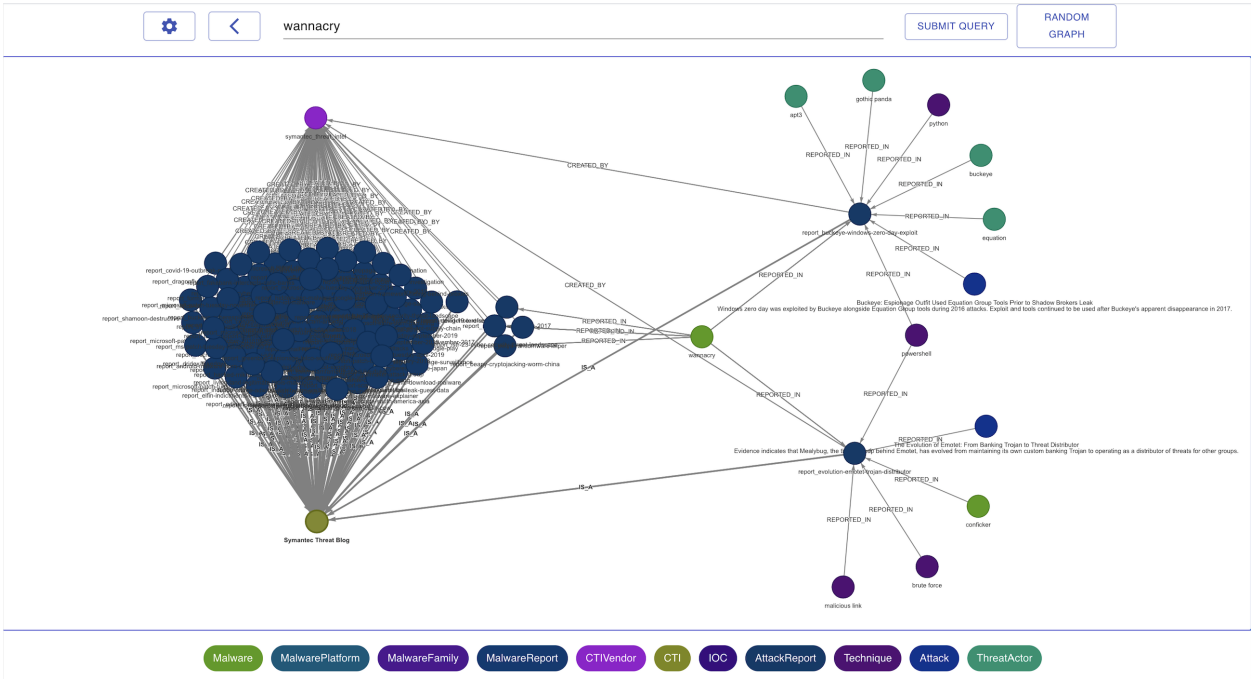

Figure 2.5: The web GUI of ThreatExtractor

We also designed features to facilitate threat search and security knowledge graph exploration. First, the GUI provides multilingual query support so that the user can search information using keywords (through Elasticsearch) or Cypher queries (through Neo4j Cypher engine), which enables the user to easily identify targeted threats in the large graph. Second, the user can drag nodes around on the canvas. The GUI actively responds to node movements to prevent overlap through an automatic graph layout using the Barnes-Hut algorithm [30], which calculates the nodes' approximated repulsive force based on their distribution. The dragged nodes will lock in place but are still draggable if selected. This feature helps the user define custom graph layouts, while still obeying the constraints of the graph layout algorithm described above, making the graph visualizable. Third, the GUI supports inter-graph navigation. This means that when a node is double-clicked, if its neighboring nodes have not appeared in the view yet, these neighboring nodes will automatically spawn. On the contrary, once the user is done investigating a node, if its neighboring nodes or any downstream nodes are shown, double clicking on the node again will hide all its neighboring nodes and downstream nodes, which also helps reduce clutter in the view. This node expansion/collapse feature is essential for convenient graph exploration.

We also designed features that provide flexibility to the user. The user can configure the number of nodes displayed and the maximum number of neighboring nodes displayed for a node. The user can view the previous graphs displayed by clicking on the back button. The

user can also fetch a random subgraph for exploration.

We note that the Neo4j database also offers Neo4j Browser, a graphical tool which allows users to interact with their graph databases. However, our UI differs from Neo4j Browser in the following ways:

1. Unlike Neo4j Browser that can only performed structured Cypher query search, our GUI also offers fuzzy keyword search powered by Elasticsearch, which is easier to use and facilitate quick exploration

2. Although our current implementation relies on the Neo4j graph database as the underlying data storage system, our UI is adaptable enough such that switching to a different data storage connector is still capable of providing the same UI functionality.

Chapter 3

Evaluation

We built ThreatExtractor based on the following tools: Python for the system, BeautifulSoup and Selenium for OSCTI reports collection, scikit-learn and Ray Tune (hyperparameter optimization) for the OSCTI report checkers, and PyTorch for the BiLSTM-CRF model for threat entity recognition.

In the evaluations, we answer the following research questions:

- **RQ1**: What is the performance of the article checker in predicting whether a given article is related to the threat context or not? Will the article checker perform better if it is trained on all OSCTI data sources combined or if it is trained on each individual OSCTI data source?
- **RQ2**: What is the performance of our security-related entity recognition model? How can data programming help? How is the performance of the entity recognition model when generalizing to unseen OSCTI data sources?
- **RQ3**: What is the statistical information for the number of collected OSCTI reports and the constructed security knowledge graph? How many new OSCTI reports are collected on a daily basis? How effective is the web GUI in threat investigation?

3.1 Evaluation Setup

Our experiments were carried out on a single virtual machine instance running Ubuntu 20.04 with an AMD EPYC 7282 CPU @ 2.80GHz and an Nvidia GRID T4-16Q GPU with 16GB RAM.

3.2 Evaluation Results

3.2.1 RQ1: Article Checker Performance

To construct the dataset, we randomly selected three OSCTI sources and a random subset of articles from each source. This led to a dataset of 755 articles from three sources: Securelist, Symantec Threat Intelligence, and Webroot. The dataset was then manually labeled. The binary labels indicate whether the article is threat-related or not. There are 517 threat-related articles and 238 non-threat related articles. We ran two experiments on this dataset to investigate RQ1: (1) For each source, we train a source-specific classifier and evaluate its performance on its own source; (2) We combine all sources to train a universal classifier and evaluate performance on each source individually.

Table 3.1: Source-specific checker results

	Symantec Threat Intelligence				Securelist				Webroot			
Models	**Accuracy**	**F1**	**FPR**	**FNR**	**Accuracy**	**F1**	**FPR**	**FNR**	**Accuracy**	**F1**	**FPR**	**FNR**
Logistic Regression	94.29%	96.00%	18.18%	0.00%	80.26%	87.18%	56.52%	3.77%	80.00%	87.10%	61.54%	0.00%
Random Forest	94.29%	96.00%	18.18%	0.00%	81.58%	88.33%	60.87%	0.00%	77.50%	85.71%	69.23%	0.00%
Linear SVM	94.29%	96.00%	18.18%	0.00%	80.26%	87.18%	56.52%	3.77%	80.00%	87.10%	61.54%	0.00%
Kernel SVM	88.57%	92.31%	36.36%	0.00%	82.89%	88.89%	52.17%	1.89%	80.00%	87.10%	61.54%	0.00%
LightGBM	94.29%	96.00%	18.18%	0.00%	82.89%	88.70%	47.83%	3.77%	77.50%	85.25%	61.54%	3.70%
XGBoost	94.29%	96.00%	18.18%	0.00%	78.95%	85.71%	47.83%	9.43%	75.00%	83.87%	69.23%	3.70%
Average	93.33%	95.38%	21.21%	0.00%	81.14%	87.67%	53.62%	3.77%	78.33%	86.02%	64.10%	1.23%

Source-Specific Checkers. We trained binary classifiers individually for each OSCTI source. For each of the three OSCTI sources, we created a train/dev/test split of 70%, 10%, and 20%. The models were then trained using the best hyperparameters found using the dev set and finally evaluated on the test set. The results are shown in Table 3.1. We define false positives as non-threat-related articles (e.g., ads, production promotions, irrelevant news) which were classified as threat-related. We define false negatives as threat-related articles which were classified as non-threat-related. We observe that the average F1 scores are above 86% and the average false negative rates (FNRs) are below 3.77%. The false positive rates (FPRs) are high, especially when trained on the Securelist and Webroot datasets, but we deemed this as acceptable as long as the FNR remains low. The goal is to extract as much information as possible without overlooking threat-related articles.

Universal Article Checker. We also trained a universal classifier which combined the three OSCTI sources. We then performed a train and dev split of 87.5% and 12.5%, respectively. We note that to preserve the ground truth across experiments, the test sets were the same for both the single source classifiers and universal classifier experiments. The results for this experiment are shown in Table 3.2.

Comparing the F1 scores for both experiments, we found that the performance of universal checkers doesn't benefit from more training data because similar to the domain generalization problem in natural language processing, the distributional shift of articles from different sources prevents the ML models from being able to learn the semantics across ar-

Table 3.2: Universal checker results

	Symantec Threat Intelligence				Securelist				Webroot			
Models	**Accuracy**	**F1**	**FPR**	**FNR**	**Accuracy**	**F1**	**FPR**	**FNR**	**Accuracy**	**F1**	**FPR**	**FNR**
Logistic Regression	94.29%	95.83%	9.09%	4.17%	84.21%	89.09%	34.78%	7.55%	82.50%	88.52%	53.85%	0.00%
Random Forest	94.29%	96.00%	18.18%	0.00%	76.32%	85.48%	78.26%	0.00%	70.00%	81.82%	92.31%	0.00%
Linear SVM	97.14%	97.96%	9.09%	0.00%	85.53%	90.43%	43.48%	1.89%	72.50%	83.08%	84.62%	0.00%
Kernel SVM	97.14%	97.96%	9.09%	0.00%	72.37%	83.20%	86.96%	1.89%	75.00%	84.37%	76.92%	0.00%
LightGBM	91.43%	93.88%	18.18%	4.17%	82.89%	88.29%	39.13%	7.55%	70.00%	81.25%	84.62%	3.70%
XGBoost	91.43%	93.88%	18.18%	4.17%	78.95%	85.45%	43.48%	11.32%	72.50%	81.97%	69.23%	7.41%
Average	94.29%	95.92%	13.64%	2.08%	80.04%	86.99%	54.35%	5.03%	73.75%	83.50%	76.92%	1.85%

ticles from different sources. Thus, based on our empirical experiment results, we believe that in practice with larger datasets, training classifiers to check different sources separately leads to better overall accuracy.

3.2.2 RQ2: Entity Recognition Performance

Table 3.3: BiLSTM-CRF OSCTI dataset statistics

OSCTI Source	Number of Reports
APTnotes	538
attcybersecurity	229
ciscoumbrella	406
cloudflare	1,545
crowdstrike	641
csoonline	1,126
fireeye	93
forcepoint	955
hotforsecurity	4,342
kasperskydaily	2,961
krebsonsecurity	1,931

We gathered a dataset corpus of 14,767 reports from APTnotes and 10 other OSCTI report sources. The source names and report counts are shown in Table 3.3. Then, to train the BiLSTM-CRF model, we labeled the OSCTI corpus using data programming. With this dataset construction, we created an a train/test split of 80% and 20% and performed two experiments: (1) We trained the BiLSTM-CRF model and evaluated it on a test set from the same sources. (2) We then evaluated the trained model on a new OSCTI source, Spiderlabs, which was not part of the original dataset. These experiments evaluate the generalizability of our BiLSTM-CRF neural network architecture. The hyperparameters and their values we chose for the experiments are: 45 epochs, batch size of 64, maximum report length of 2,000,

Table 3.4: Entity recognition on test set from same sources

	Precision	**Recall**	**F1**	**Support**
B-BADACTOR	99%	79%	87%	1,140
B-MALWARE	99%	95%	97%	5,294
B-MITIGATION	97%	99%	98%	467
B-TECHNIQUE	100%	100%	100%	22,357
B-TOOL	100%	100%	100%	18,699
I-BADACTOR	93%	66%	78%	346
I-MALWARE	85%	96%	90%	211
I-MITIGATION	97%	99%	98%	651
I-TECHNIQUE	96%	98%	97%	1,713
I-TOOL	100%	100%	100%	12
O	100%	100%	100%	4,600,467
Accuracy	99.98%		**Weighted F1**	99.98%

Table 3.5: Entity recognition on Spiderlabs OSCTI source

	Precision	**Recall**	**F1**	**Support**
B-BADACTOR	100%	100%	100%	38
B-MALWARE	100%	87%	93%	1,459
B-MITIGATION	98%	96%	97%	179
B-TECHNIQUE	99%	99%	99%	7,155
B-TOOL	100%	100%	100%	2,599
I-BADACTOR	67%	100%	80%	2
I-MALWARE	100%	88%	94%	182
I-MITIGATION	94%	85%	89%	52
I-TECHNIQUE	93%	89%	91%	577
I-TOOL	0%	0%	0%	1
O	100%	100%	100%	891,349
Accuracy	99.95%		**Weighted F1**	99.94%

256 hidden units, embedding dimension of 512, 2 LSTM layers, the Adam optimizer, and a learning rate of 1e−2.

Performance on Test Data from the Same Sources. We evaluated the model performance on the test set from the same sources. As shown in Table 3.4, we are achieving high weighted F1 score (99.98%) and accuracy (99.98%), which is practical in real-world use.

Performance on Test Data From Spiderlabs OSCTI Source. We evaluate the performance of the model trained in the previous experiment on a new OSCTI source, Spiderlabs, which contains 1,349 articles. The goal of this experiment is to evaluate the generalizability of the model. The results for the experiment are shown in Table 3.5. While we expect a performance drop in most NLP domain generalization tasks, here we see that the overall

accuracy is still 99.95% while the weighted F1 score is 99.94%.

In order to explain the high F1 score, we compared the data used for the two experiments and observe a high similarity between text in the training set and the text from the unseen source. Specifically, 96% of the tokens that have a targeted entity type exist in both the training dataset and the new Spiderlab OSCTI source. Because the model performs well on seen entities, it explains the high generalizability across different sources. This suggests that different from the article checkers where it is more practical to train separate classifiers for different OSCTI sources, training a universal entity recognizer achieves better results.

We also observe that the entity `I-TOOL`, which only has one instance, achieved a 0% F1 score. This is because the model has a hard time identifying the entity type when a token is used in isolation compared to when it is combined with other tokens. For example, on its own the *powershell* token has a correct entity type of `B-TECHNIQUE` but when used with other tokens such as *powershell empire*, the correct entity types should be <`B-TOOL I-TOOL`>. Uncoincidentally, this exact token phrase consists of the only instance of `I-TOOL` in the Spiderlabs dataset, but the model is unable to predict the correct entity tags on this instance and predicts the token pair as <`B-TECHNIQUE B-TOOL`> instead.

3.2.3 Measurement Study & Case Study

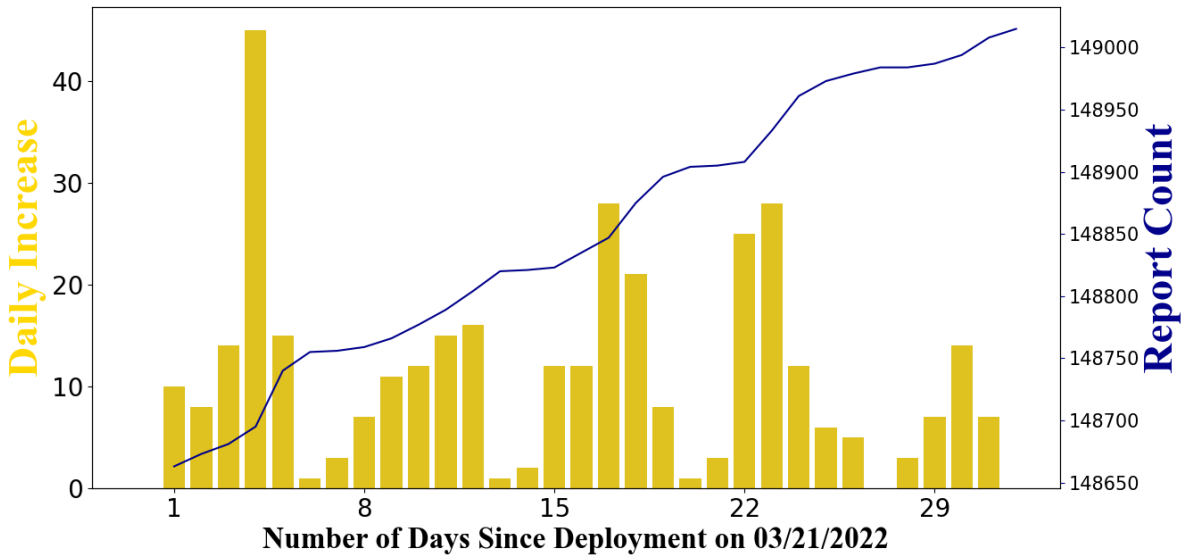

Figure 3.1: OSCTI report collection update frequency

Measurement Study. We perform a measurement study to provide statistical information on both the collected OSCTI reports and the final constructed knowledge graph: (1) We

Table 3.6: ThreatExtractor Security Knowledge Graph Entity Names and Counts

Entity Name	**Count**
AttackReport	87,153
Attack	81,074
IOC	58,429
Malware	23,359
BlogTag	23,305
MalwareReport	22,993
MalwareFamily	21,328
Vulnerability	9,136
VulnerabilityReport	9,136
BlogCategory	1,663
Technique	1,449
ThreatActor	248
MalwarePlatform	147
Tool	72
Mitigation	70
CTIVendor	36
CTI	3
Total	339,601

Table 3.7: ThreatExtractor Security Knowledge Graph Relation Types and Counts

Relation Type	**Count**
COOCCUR	620,886
REPORTED_IN	546,318
HAS_TAG	155,300
CREATED_BY	119,282
IS_A	119,282
HAS_CATEGORY	77,466
ACTION	26,230
BELONGS_TO_FAMILY	22,894
ON_PLATFORM	21,044
Total	1,708,702

deployed the system for 1 month (starting 03/21/2022 until 04/21/2022) and recorded the total number of OSCTI reports collected daily. We started with a total number of 148,663 reports on day 1 and ended with 149,015 reports in the system on the final day. Table 2.1 shows the final number of reports for each OSCTI source along with their names and URLs. We also present the update frequency of the collected reports in Figure 3.1. We observe that there are more reports collected on weekdays (i.e., days 1-5, 8-12, etc) compared to the weekend (i.e., days 6-7, 13-14, etc). (2) At the time of writing this paper, the final constructed knowledge graph contains 347K+ entities and 1.73M+ relations. We provide a count breakdown of the entity names and relation types in Table 3.6 and Table 3.7.

Security Case Study. To demonstrate the effectiveness of the GUI, we walk through 2 threat investigation case studies with one using Cypher query search and the other using keyword search:

- *Cypher query search on "wannacry"*: We investigate the wannacry ransomware to observe its behavior and impact. After submitting the Cypher search query `MATCH(n) WHERE n.name = "wannacry" RETURN n`, which takes 2.93 seconds to execute, we see a *Malware* node named `wannacry`. Double clicking on this node will reveal direct neighbors which enables convenient inter-graph navigation and facilitates simpler knowledge graph exploration. From the unveiled neighboring nodes, we see that there are many OSCTI reports that discuss this ransomware. Users are able to drag these nodes around and pan/zoom in and out of the canvas to facilitate user interaction. Users are also able to hover over nodes which will display key information. For example, in the report nodes, information such as the name, title, node type, and URL are shown. If we select one of the report nodes, which has the title *report_ buckeye-windows-zero-day-exploit*, and double click, we will see relevant nodes that contain information related to the Wannacry ransomware, including threat actors (e.g., gothic panda, apt3, and buckeye) and techniques used during the process (e.g., Python and Powershell). Besides this report node, there are also other report nodes, including one named *report_ petya-ransomware-wiper*. If we double click on this node and reveal its neighbors, we will see additional information regarding wannacry including related malware such as petya and IOCs (e.g., rundll32.exe). From this case study, we see that ThreatExtractor's GUI provides convenient interactive features to help users explore the security knowledge graph and acquire threat knowledge.

- *Keyword search for "cozyduke"*: We use keyword search supported by Elasticsearch to explore the threat actor cozyduke. After submitting the keyword search query `cozyduke`, which takes 60 milliseconds to execute, we see the `cozyduke` node which is linked to 1 OSCTI report node named *report_ forkmeiamfamous-seaduke-duke*. Similar to the case study above, the report node reveals information such as related threat actors (e.g., `cozy bear`, `the dukes`) and techniques used by `cozyduke` and these threat actors including `file deletion` and `Powershell`. From this case study, we see that ThreatExtractor's GUI provides convenient keyword-based threat search and knowledge graph exploration.

Chapter 4

Related Work

In this chapter, we survey five categories of related work.

4.1 OSCTI Analysis and Management

Besides existing standards and platforms for OSCTI gathering and management [26, 10, 13, 2, 12, 18], research progress has developed to better analyze OSCTI reports, including extracting IOCs [50], extracting threat action terms from semi-structured Symantec reports [42], understanding vulnerability reproducibility [56], and measuring threat intelligence quality [49, 33]. Research has also proposed to leverage IOC information extracted from individual OSCTI reports for cyber threat hunting [37]. ThreatExtractor distinguishes from all these works in the sense that it targets the automated extraction of security-related knowledge from OSCTI reports using a combination of AI and NLP techniques. Then it uses the extracted knowledge in the form of entities and relations to build a security knowledge graph.

4.2 CTI Ontologies

STIX[12] is an open standard CTI format for exchanging threat intelligence by providing a flexible and expressive representation language, but lacks support for inference and cyber investigation [31, 52]. There are some ontologies [55, 64, 40] with inference functions, but they only focus on sub-domains of threat intelligence, such as IDS and malware behavior. The STUCCO ontology [43] is designed to integrate both structured and unstructured data sources but lacks support for high-level threat knowledge such as techniques. Unlike the ontologies mentioned above, the ontology of ThreatExtractor is a hierarchical structure that includes both high-level and low-level threat intelligence, such as IOCs, malware, threat actors, and techniques. Such comprehensive threat intelligence enables better threat knowledge inference. For example, two attacks using the same tools are likely to use similar techniques and have similar attack traces. MITRE ATT&CK[19] is a manually curated

knowledge base by security experts for adversary behaviors based on real-world observations, but it does not contain IOC relations and also does not focus on automated knowledge extraction from unstructured reports as done in ThreatExtractor.

4.3 Threat Knowledge Extraction

There have been several works proposed for threat knowledge extraction from OSCTI reports. iACE[50] extracts IOCs from security articles using a graph mining technique. ChainSmith [68] is an IOC extraction system which classifies the IOCs into different attack campaigns (e.g., baiting, exploitation, installation and C&C) using neural networks. TTPDrill [42] extracts threat actions from Symantec reports and maps them to pre-defined attack patterns. EXTRACTOR [61], ThreatRaptor [37], and HINTI [66] use various NLP techniques to extract IOC entities and relations. These works primarily focus on IOCS or IOC relations, while ThreatExtractor covers a wider range of entities (e.g., threat actors, techniques, tools) and relations. In addition, these works only extract knowledge from a single OSCTI report while ThreatExtractor automatically extracts knowledge from a large volume of OSCTI reports and represents the knowledge in the form of entities and relations in a security knowledge graph.

4.4 Knowledge Graphs

A knowledge graph is a large-scale semantic network composed of entities and their relationships. It plays an increasingly important role in intelligent recommendation, hidden information mining, etc. There are a number of knowledge graphs [53, 29, 65, 62, 51, 54] that are designed for storing and representing general knowledge, such as people, locations, and organizations. Different from them, ThreatExtractor targets automated extraction of high-level cyber threat knowledge with more threat behavior details gathered from OSCTI reports and builds a security knowledge graph for the security domain. In this way, ThreatExtractor provides a chance to uncover valuable threat knowledge, such as the goals of attackers and attack stages in multi-step threat scenarios. Thus, by leveraging ThreatExtractor, a variety of downstream security applications (e.g., threat hunting, attack investigation, intrusion detection) can be further empowered.

Chapter 5

Empowering Downstream Security Applications

ThreatExtractor is capable of empowering many existing downstream security applications while supporting new applications that were not previously possible. Since ThreatExtractor automatically extracts structured knowledge from unstructured OSCTI reports, systems and platforms that previously benefit from the structured OSCTI can also benefit from the knowledge provided by ThreatExtractor. For example, the knowledge extracted by ThreatExtractor can be converted into open formats like STIX [12], exchanged in platforms like AlienVault OTX [13], and integrated in existing intrusion detection systems [57, 47, 48] that take IOC and STIX feeds as input.

Other works like [37] have proposed to use the knowledge extracted from individual OSCTI reports to guide threat hunting. Thus, with the automated extraction of threat knowledge by ThreatExtractor combined with the constructed security knowledge graph, a new way of threat hunting can be enabled. For example, we can reduce the efforts of manual query construction in threat hunting, by synthesizing or suggesting queries based on the security knowledge graph and partial user input. The knowledge extracted from OSCTI reports can also be used to empower other types of defenses like attack detection [38] and attack investigation [39, 34]

Chapter 6

Conclusion

We have presented ThreatExtractor, a system for automated cyber threat knowledge extraction. ThreatExtractor collects a large number of OSCTI reports from various sources, uses a combination of AI and NLP based techniques to extract comprehensive threat knowledge from the reports, constructs a security knowledge graph, and persists the knowledge in the database. ThreatExtractor also provides a web GUI that facilitates security knowledge graph exploration and cyber threat knowledge acquisition. ThreatExtractor has the potential to empower a variety of downstream security applications.

Bibliography

[1] SolarWinds Cyberattack. https://www.zscaler.com/resources/security-terms-glossary/what-is-the-solarwinds-cyberattack.

[2] The History of OpenIOC, 2013. https://www.fireeye.com/blog/threat-research/2013/09/history-openioc.html.

[3] Home Depot Confirms Data Breach At U.S., Canadian Stores, 2014. http://www.npr.org/2014/09/09/347007380/home-depot-confirms-data-breach-at-u-s-canadian-stores.

[4] Target Data Breach Incident, 2014. http://www.nytimes.com/2014/02/27/business/target-reports-on-fourth-quarter-earnings.html?_r=1.

[5] The Sony Pictures hack, explained, 2014. https://www.washingtonpost.com/news/the-switch/wp/2014/12/18/the-sony-pictures-hack-explained/.

[6] The cozyduke apt, 2015. https://securelist.com/the-cozyduke-apt/69731/.

[7] Symantec threat intelligence, 2017. https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence.

[8] The Marriott data breach, 2018. https://www.consumer.ftc.gov/blog/2018/12/marriott-data-breach.

[9] Open IE 5, 2018. https://github.com/dair-iitd/OpenIE-standalone.

[10] Threatcrowd, 2019. https://www.threatcrowd.org/.

[11] The Equifax Data Breach, 2020. https://www.ftc.gov/equifax-data-breach.

[12] Structured Threat Information eXpression, 2021. http://stixproject.github.io/.

[13] AlienVault OTX, 2022. https://otx.alienvault.com/.

[14] AT&T Alien Labs Research Blog, 2022. https://cybersecurity.att.com/blogs/labs-research/.

[15] Blackenergy apt attacks in ukraine, 2022. https://www.kaspersky.com/resource-center/threats/blackenergy.

[16] Kaspersky threat encyclopedia, 2022. https://threats.kaspersky.com/.

[17] Krebs on security, 2022. https://krebsonsecurity.com/.

[18] MISP - Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing, 2022. https://www.misp-project.org/.

[19] Mitre att&ck, 2022. https://attack.mitre.org.

[20] Neo4j, 2022. http://neo4j.com/.

[21] PhishTank, 2022. https://www.phishtank.com/.

[22] Schneier on security, 2022. https://www.schneier.com/.

[23] SecureList, 2022. https://securelist.com/.

[24] Sophos News, 2022. https://news.sophos.com/en-us/.

[25] The Hacker News, 2022. https://thehackernews.com/.

[26] ThreatMiner, 2022. https://www.threatminer.org/.

[27] Trend micro threat encyclopedia, 2022. https://www.trendmicro.com/vinfo/us/threat-encyclopedia/.

[28] Angeli, G., Johnson Premkumar, M. J., and Manning, C. D. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)* (July 2015), pp. 344–354.

[29] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. Dbpedia: A nucleus for a web of open data. In *The semantic web*. 2007.

[30] Barnes, J., and Hut, P. A hierarchical o (n log n) force-calculation algorithm. *nature* (1986).

[31] Casey, E., Barnum, S., Griffith, R., Snyder, J., van Beek, H., and Nelson, A. The evolution of expressing and exchanging cyber-investigation information in a standardized form. In *Handling and Exchanging Electronic Evidence Across Europe*. Springer, 2018, pp. 43–58.

[32] Chen, T., and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (2016), p. 785–794.

[33] Dong, Y., Guo, W., Chen, Y., Xing, X., Zhang, Y., and Wang, G. Towards the detection of inconsistencies in public security vulnerability reports. In *28th USENIX Security Symposium (USENIX Security)* (2019), pp. 869–885.

[34] Fang, P., Gao, P., Liu, C., Ayday, E., Jee, K., Wang, T., Ye, Y. F., Liu, Z., and Xiao, X. Back-propagating system dependency impact for attack investigation. In *Proceedings of the 31st USENIX Security Symposium* (2022), SEC '22.

[35] Forney, G. The viterbi algorithm. *Proceedings of the IEEE 61*, 3 (1973), 268–278.

[36] Gao, P., Liu, X., Choi, E., Soman, B., Mishra, C., Farris, K., and Song, D. *A System for Automated Open-Source Threat Intelligence Gathering and Management*. Association for Computing Machinery, 2021, p. 2716–2720.

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

[38] Gao, P., Xiao, X., Li, D., Li, Z., Jee, K., Wu, Z., Kim, C. H., Kulkarni, S. R., and Mittal, P. SAQL: A stream-based query system for real-time abnormal system behavior detection. In *USENIX Security* (2018).

[39] Gao, P., Xiao, X., Li, Z., Xu, F., Kulkarni, S. R., and Mittal, P. AIQL: Enabling efficient attack investigation from system monitoring data. In *USENIX ATC* (2018).

[40] Grégio, A., Bonacin, R., Nabuco, O., Afonso, V. M., Lício De Geus, P., and Jino, M. Ontology for malware behavior: A core model proposal. In *2014 IEEE 23rd International WETICE Conference (WETICE)* (June 2014), pp. 453–458.

[41] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

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

[43] Iannacone, M., Bohn, S., Nakamura, G., Gerth, J., Huffer, K., Bridges, R., Ferragut, E., and Goodall, J. Developing an Ontology for Cyber Security Knowledge Graphs. In *Proceedings of the 10th Annual Cyber and Information Security Research Conference* (Apr. 2015), CISR '15, pp. 1–4.

[44] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)* (2017), p. 3149–3157.

[45] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)* (2001), pp. 282–289.

[46] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)* (2016), pp. 260–270.

[47] Lee, W., and Stolfo, S. J. Data mining approaches for intrusion detection. In *7th USENIX Security Symposium (USENIX Security)* (1998), p. 6.

[48] Lejonqvist, G., and Larsson, O. Improving the precision of an intrusion detection system using indicators of compromise : - a proof of concept -. Master's thesis, Luleå University of Technology, Department of Computer Science, Electrical and Space Engineering, 2018.

[49] Li, V. G., Dunn, M., Pearce, P., McCoy, D., Voelker, G. M., and Savage, S. Reading the tea leaves: A comparative analysis of threat intelligence. In *28th USENIX Security Symposium (USENIX Security)* (2019), pp. 851–867.

[50] Liao, X., Yuan, K., Wang, X., Li, Z., Xing, L., and Beyah, R. Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)* (2016), pp. 755–766.

[51] Mahdisoltani, F., Biega, J., and Suchanek, F. M. Yago3: A knowledge base from multilingual wikipedias. In *7th biennial conference on innovative data systems research (CIDR)* (2013).

[52] Mavroeidis, V., and Bromander, S. Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence. In *2017 European Intelligence and Security Informatics Conference (EISIC)* (Sept. 2017), pp. 91–98.

[53] Miller, G. A. Wordnet: a lexical database for english. *Communications of the ACM 38*, 11 (1995), 39–41.

[54] Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., et al. Never-ending learning. *Communications of the ACM 61*, 5 (2018), 103–115.

[55] More, S., Matthews, M., Joshi, A., and Finin, T. A Knowledge-Based Approach to Intrusion Detection Modeling. In *2012 IEEE Symposium on Security and Privacy Workshops (S&P Workshop)* (May 2012), pp. 75–81.

[56] Mu, D., Cuevas, A., Yang, L., Hu, H., Xing, X., Mao, B., and Wang, G. Understanding the reproducibility of crowd-reported security vulnerabilities. In *27th USENIX Security Symposium (USENIX Security)* (2018).

[57] Pasquier, T., Han, X., Moyer, T., Bates, A., Hermant, O., Eyers, D., Bacon, J., and Seltzer, M. Runtime analysis of whole-system provenance. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)* (2018), pp. 1601–1616.

[58] Rajaraman, A., and Ullman, J. D. *Mining of massive datasets*. Cambridge University Press, 2011.

[59] Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems (NeurIPS)* (2016), pp. 3567–3575.

[60] Sang, E. F., and Veenstra, J. Representing text chunks. *arXiv preprint cs/9907006* (1999).

[61] Satvat, K., Gjomemo, R., and Venkatakrishnan, V. N. Extractor: Extracting attack behavior from threat reports. arXiv.

[62] Speer, R., Chin, J., and Havasi, C. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence (AAAI)* (2017), pp. 4444–4451.

[63] Symantec. Symantec, 2017. `https://www.symantec.com/`.

[64] Undercofer, J., Joshi, A., Finin, T., and Pinkston, J. A Target-Centric Ontology for Intrusion Detection. *Workshop on Ontologies in Distributed Systems* (2003).

[65] Vrandečić, D., and Krötzsch, M. Wikidata: A free collaborative knowledgebase. *Communications of the ACM 57*, 10 (2014), 78–85.

[66] Zhao, J., Yan, Q., Liu, X., Li, B., and Zuo, G. Cyber threat intelligence modeling based on heterogeneous graph convolutional network. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)* (Oct. 2020), pp. 241–256.

[67] Zhou, G., and Su, J. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (2002), ACL '02, p. 473–480.

[68] Zhu, Z., and Dumitras, T. Chainsmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)* (2018), pp. 458–472.