International Journal of Cyber-Security and Digital Forensics (IJCSDF) 11(2): 20-33
The Society of Digital Information and Wireless Communications (SDIWC), 2022 ISSN: 2305

Enhanced Crime and Threat Intelligence Hunter with Named Entity Recognition and Sentiment Analysis

James H. Ng and Peter K.K. Loh
Singapore Institute of Technology, Singapore
{James.Ng, Peter.Loh}@SingaporeTech.edu.sg

ABSTRACT

Collecting cybercrime evidence on the Internet typically involves reconnaissance and analyses of information extracted. Scouring the Internet, especially the Deep Web, often requires manual effort and is time-consuming. Hence, it is imperative to have an efficient framework for an intelligent tool to gather and correlate cyber intel automatically according to programmed directives. In this paper, we present an updated design to our prior threat intelligence hunter [1]. We make use of machine learning and sentiment analysis for parsing and correlating linguistic intel.

KEYWORDS

Cybercrime, deep web crawler, machine learning, named entity recognition, sentiment analysis, threat intelligence.

1 INTRODUCTION

Cybercrimes and online threats like phishing, data breach, and ransomware or spyware obtained from the Deep Web, have risen significantly in recent times. Hence, intelligence gathering in the cyber world to identify potentially malicious actors has become practically important and significant to crime prevention as well as risk management and compliance. Collecting shreds of evidence on the Internet to narrow down the suspects may comprise of three stages. The first stage involves crawling the Internet (the Surface Web and the Deep/Dark Web) for information about a person or device, as metaphorically depicted in Figure 1. The second stage extracts named entities as well as conducts programmed training on malicious terms, for contextual information from the text gathered in the first stage. The third stage associates or correlates different informational entities to analyze sentiments to draw conclusions on potential suspicious actors.

The role of Artificial Intelligence (AI) and Machine Learning (ML) has gained prominence in cybercrime investigations recently [2]. They help to replace the error prone and time-consuming activity of manual labelling of information and minimize the efforts of incident responders by automatically sensing and tagging threat intelligence. The use of natural language processing is central to the development of cyber threat intelligence platforms. [3] discusses how the integration of ML helps to build a reliable model pertaining to malware detection, spam classification and network intrusion identification. The implementation of AI in turn helps to transform overwhelming threat intelligence generated by structured and unstructured sources to actionable data feeds. It suggests that such a model contains phases for data accumulation, feature extraction and language processing, and a Naïve Bayes classifier.

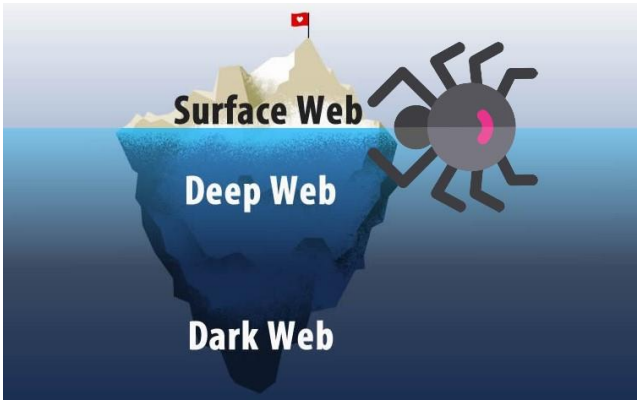

Figure 1. Crawling the Web

In an earlier work [4], we presented a survey of existing research on related tools and described the preliminary prototype design of our crime and threat intelligence hunter (CTIH). We demonstrated its capability with some test cases and concluded with demonstrated correlation effectiveness. Its performance improved with the

use of crime and threat scenario-focused feature values in the pre-defined machine learning models and training with larger, more scenario-focused datasets. As mentioned in the proposed future enhancements, we are also researching to improve the tool's performance and obtain better data classification.

In this paper, we evaluated our enhanced CTIH against more recent digital forensic tools, highlighting their comparative strengths and weaknesses. We have also enhanced our hunting tool design by incorporating sentiment analysis. This paper is organized as follows: Section I provides an introduction and background summary of our earlier research. Section II presents a review of existing digital forensic tools. Section III details the improved web user interface and the enhanced architectural design of our machine-learning- and sentiment-analysis-assisted crime and threat intelligence hunter. Section IV provides a critique overview of the new design and surfaces potential innovations. Section V concludes with the limitations encountered during our research and development work and the obstacles faced in future research.

2 SURVEY OF EXISTING CYBER INTELLIGENCE TOOLS

2.1 Maltego

Maltego [5] is an application that discovers related entities of a specified Entity and, in turn, other entities related to them. Some types of an Entity are devices, events, groups, infrastructure elements, locations, malware, ports and services, personal details, and social media tags. It visualizes the relationships between the entities discovered graphically in various layouts. It uncovers the connections and entities with the help of a Transform, which is a piece of code that searches for information related to an Entity. It allows querying of an API or database to show related information on the graph. Multiple transforms, filters, and actions are chained together into a Machine to discover more in-depth layered or filtered data. This design enables the automation of common and tedious tasks.

In a licensed Maltego Enterprise's Transform Hub, a user would have access to over 58 free and paid data sources. For additional transforms requiring paid data subscriptions, the user can purchase a subscription in Maltego's online shop or apply an existing API key. For our CTIH, we have designed the incorporation of third-party REST APIs based on the transforms Maltego provided and included the Shodan, Censys, and MacVendors APIs by default. The API keys used are for a free subscription and can be updated to a premium subscription.

Maltego automatically links and combines all information in one graph and visually maps out the relationships of the results returned by the transforms. Our CTIH also automatically generates a graph with information gathered from the open-source intelligence (OSINT) APIs as well as web pages returned by search engines.

2.2 theHarvester

theHarvester [6] is a command-line tool available in Kali Linux designed to be used in the early stages of a penetration test or red team engagement to gather open-source intelligence (OSINT). It acts as a wrapper for a variety of search engines to find emails accounts, subdomain names, virtual hosts, open ports, and usernames related to a domain from different public sources. As it executes via commands in the terminal, a trained user well-versed with the parameter flags may perform searches very efficiently.

To gather intelligence on a company or domain, theHarvester employs third-party OSINT APIs such as Shodan and Google Dorks. It allows the user to specify other data sources ranging from Baidu and Censys to Twitter and VirusTotal too. Though our CTIH does not offer as many options for data sources, we make it user-friendly by providing a graphical user interface that simplifies and standardizes the search parameters accepted for the widely used crime and threat intelligence data sources.

2.3 ReconSpider and the OSINT framework

[7] gives an overview on the use of open-source intelligence in criminal investigation and states that the commonly used OSINT methodological tools for crime detection are lexical, textual content analysis, social network analysis, and geoinformatics analysis based on the user's social media tools. ReconSpider [8], an OSINT

Framework for scanning IP Addresses, Emails, Websites, and Organizations, aims to incorporate these tools such as for domain search and WHOIS, to find out the locational and personal information from different sources. However, it uses APIs such as Shodan.io and IP Stack that are subscription-based and have limited and restricted usage per month.

ReconSpider tries to cover as many aspects of the OSINT Framework [9] as possible, but it has not been updated for more than a year. Some features no longer function as the APIs used have changed in terms of the REST queries and results returned. Our CTIH adopts similar features and re-implements functionality with updated codes and API keys.

2.4 TorBot

[10] is an open-source intelligence tool developed in Python which primarily focuses on Dark Web content. It simplifies the process of identification and analysis of onion services and gathers intelligence about Dark Web services. It constitutes three modules: (1) a crawler that further identifies other links, emails along with metadata, and text; (2) an intelligence extractor that collects scripts, robots, files, emails, fuzz URLs and checks for basic web vulnerabilities; (3) a visualizer that can examine the relationship of links and creates an interactive tree graph to display to the user.

Setting up and running the TorBot is not as straightforward as the CTIH though. Besides having the dependency of having a Tor proxy service running, it also requires the GoTor [11] service to be run in tandem. We have bundled CTIH to be deployed with set Python packages that assist to set up the TOR proxy and download the browser driver and extension to access onion websites.

2.5 Valence Aware Dictionary and sEntiment Reasoner (VADER)

[12] proposes a methodology that combines information extracted from the Deep Web using smart crawlers and technical indicators of threats from the Surface Web. To get the processed texts for training its classification matrix, it scraps social media tweets and Dark Web hacker forums' posts for buzzwords, titles, and descriptions. With the features extracted, the texts are combined and transformed into a word embedding matrix. The matrix is provided in the Gensim Word2Vec format.

The design of CTIH is based on AllenNLP for machine learning and modularity, takes in the more commonly used GloVe word embeddings, thus limiting the importing of the new word embedding matrix. Hence, [12] can increase the portability of its matrix with the GloVe format.

Apart from recognizing domain-specific named entities from the web information extracted, natural language processing is also used to apply sentiment analysis to data from both the Surface Web and the Dark Web. [13] explores the link between community behavior and malicious activity by mapping the malicious events to hacker behavior. It introduces three sentiment models: VADER, LIWC, and SentiStrength, to compute standardized daily and running average sentiment scores. The scores provide for the computation of correlations between the texts and malicious event types to predict future events.

Out of the three sentiment analysis methods, VADER is the only Python-based implementation that our design referenced against. A rule-based sentiment model with a dictionary made robust by additional rules, it is tuned for social network context. Our sentiment analysis model would explore similar design with a dictionary for the cybersecurity domain.

LIWC15 is a stand-alone program that ignores context, irony, sarcasm, and idioms. SentiStrength has a Java implementation that though do better with social media, it cannot exploit indirect indicators of sentiment. Hence, they were not considered as part of our design reference.

2.6 Pattern

[14] is a web mining module for Python with tools for data mining, natural language processing (NLP), machine learning, and network visualization and analysis. For data mining, its 'pattern.web' module has various search engine APIs, web crawlers, and document parsers to find web pages and scraps them into textual data. Its 'pattern.en' and 'pattern.search' modules form a NLP toolkit with a search algorithm to retrieve sequences of words from

text annotated with word types.

For machine learning, the 'pattern.vector' module contains models that can be used for clustering, classification, and latent semantic analysis. Coupled with the modules for NLP, a particular piece of text is transformed into a vector, which a classifier uses it to predict the sentiment of the text. Then the 'pattern.graph' module provides a graph data structure that represents relations between terms.

Table 1. Comparison of OSINT Tools

Tool	Main Features/Functions
Maltego	Visualize the relationships between entities and uncovers new connections and entities
theHarvester	Command-line wrapper for search engines to find entities such as usernames, emails, and subdomains
ReconSpider	Use of a collection of subscription-based APIs such as Shodan and IP stack
TorBot	Identify and analyze onion services to gather Dark Web intelligence
VADER	A rule-based sentiment model with a dictionary and added rules for robustness
Pattern	Modules for data mining, natural language processing, machine learning, and network visualization

Our CTIH adopts a similar architecture and features data mining, NLP and graphical visualization. We query from multiple search engines, crawl and scrap the returned web links, and performed named entity recognition and sentiment analysis to display the extracted terms in a knowledge graph.

To summarize, the existing research above makes use of open-source intelligence tools and machine learning in the form of natural language processing to determine malicious activities or identify the actors of suspicious behaviors. A list of the OSINT tools is given in Table 1 to compare their main features. Our research intends to incorporate similar functionalities to these tools while trying to address any gaps these tools may have as well as supporting the various phases of cybercrime investigation. Table 2 shows the features of CTIH compared to the other OSINT tools.

The following section details the streamlined architectural design of our machine-learning based crime and threat intelligence hunter prototype. It minimizes the amount of search parameters an investigator needs to input and automates the steps the investigator takes. The information obtained is then fed into the digital crime learning analytics system that we are developing. It forms part of a larger smart digital forensics platform that is funded by the National Research Foundation of Singapore.

3 ARCHITECTURAL DESIGN

The new design of the Dark Web crime and threat intelligence hunter (CTIH) starts from simplifying the web user interface to streamline the workflow. We want to strike a balance between a minimal number of user input parameters and substantial search coverage for optimized performance, reducing wait times.

We have reduced the number of input fields by removing the search parameters of name, email and MAC address from our original design. The name and email fields now form a part of the free text search while the MAC address is included in the IP/domain/URL search

Table 2. CTIH compared to OSINT Tools

	CTIH	Maltego	theHarvester	ReconSpider	TorBot	VADER	Pattern
Graphical User Interface	✓	✓	×	×	×	×	×
Third-party APIs for IP/MAC query	✓	✓	✓	✓	×	×	×
Query by search engines	✓	✓	✓	×	×	×	✓
Dark Web crawling and scraping	✓	✓	×	×	✓	×	×
NLP for Named Entity Recognition (NER) and Sentiment Analysis (SA)	✓	✓	×	×	×	✓	✓
Knowledge Graph Analysis	✓	✓	×	×	×	×	✓

as depicted in Figure 2. As every search performed is tied to an investigation, the investigated case's crime type, if available, is added to the free text search automatically.

The user can choose to search the Surface Web or the Dark Web or both at the same time for the free text input field by selecting the corresponding radio buttons. This functional flow is represented in Figure 3. As for the username search, we currently support the querying of profiles from a comprehensive list of more than 70 social media websites and forums.

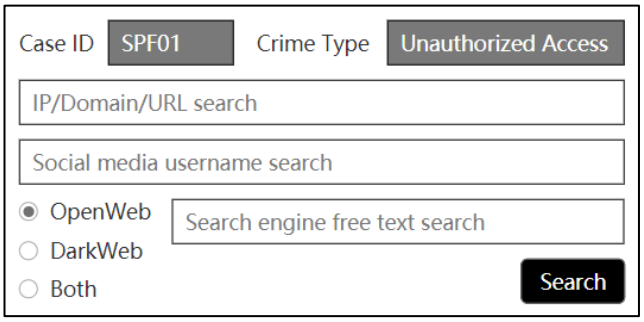

Figure 2. Search Parameters

3.1 Crime Type Searching

The crime type of a selected case ID is a default search parameter included along with the other free text search parameters. It is also used for crawling the MITRE ATT&CK® [15] website's tactics and techniques for related cybersecurity terms. These related terms will in turn be used to ascertain entities extracted via NER.

3.2 Domain/IP Searching

If this input field is entered, we will retrieve network information such as WHOIS details, domain/IP locations, and service providers. These network and website information are saved into a database, along with other URLs associated with the domain.

3.3 Social Media Username Searching

If this input field is entered, we will search for profiles from Facebook, Twitter, LinkedIn, and more, that are tied to the username provided. The profile pages are scraped to extract user details such as name, email, and organization, using a pre-trained named entity recognition model. The details are stored into the database.

3.4 Free Text Searching

The full search query is built by taking the user input as it is and concatenate with the crime type as described earlier. The terms are joined based on basic search operators wherever a search engine allows, and we will not filter out any other operators entered by the user.

For the Surface Web, we make use of Google, Yahoo! and Bing to return pages of search results containing links to websites. Duplicated results are removed before being stored into the database.

As for the Deep Web, the search engines used are VisiTOR, HayStacks, Tor66 and DarkSearch. This list may be updated whenever better search engines are known and to replace retired ones. Similarly, for the Surface Web search, each search engine will return a page of onion links that are saved to the database after duplicates have been removed.

3.5 Web Crawling and Scraping

Once the search has completed, the saved web and onion links are retrieved from the database and accessed individually. These links are separately searched via a multi-threaded function that crawl and scrap relevant text data from the HTML pages. The text data is in turn processed into sentences, with the removal of newlines. Scraping is an important module as it filters and extracts out potential intelligence inputs for named entity recognition and sentiment analysis.

3.6 Named Entity Recognition

In our previous design, we used spaCy's [16] pre-trained statistical model, specifically the 'en_core_web_sm' and a Resume-Parser [17], to extract the following list of information:

- Emails
- Phone Numbers
- Job Titles
- Schools
- Academic Qualifications
- Organizations
- Locations

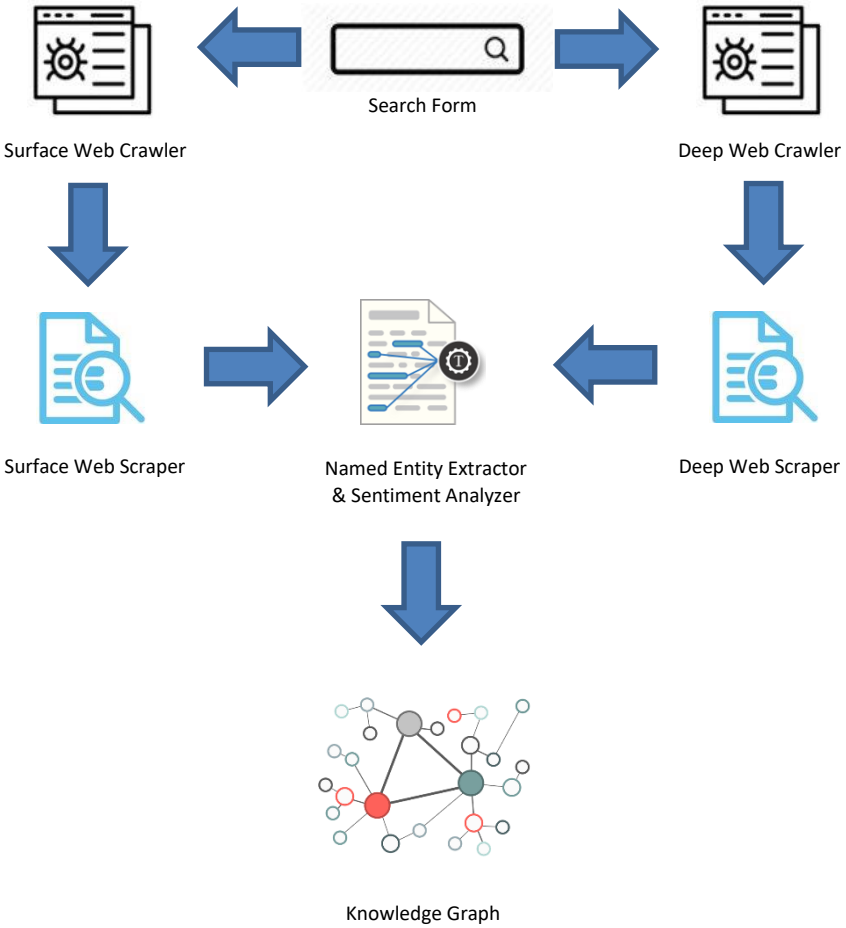

Figure 3. Dedicated Surface and Deep Web crawler and scrapper

In our new design, we have swapped spaCy with AllenNLP [18] to extract the above-named entities. It provides an ELMo-based NER model that was trained on the CoNLL-2003 NER dataset. It uses a Gated Recurrent Unit (GRU) character encoder as well as a GRU phrase encoder, and it starts with pretrained GloVe vectors for its token embeddings.

The text data scrapped in the previous task is passed into this model sentence by sentence. It recognizes terms related to persons, locations and organizations, as well as some miscellaneous types such as religions, nationalities and languages.

However, to extract problem domain-related terms, we must train our own custom NER model using datasets that have words labeled with tags created from and for the specified domain. AllenNLP allows us to do just that easily by updating its training configuration file.

The configuration file is in Jsonnet format. The filename is supplied to the AllenNLP command as a parameter for execution. In this file, we specify the data files, change the dataset reader, swap the model's word embedder or encoder, and update the trainer's settings. Our extension to AllenNLP's Conll2003 dataset reader that caters to datasets with UTF8 encoding, as shown in Figure 4, can be used simply by replacing the type "conll2003" in the config with its registered name "cyber2003".

3.7 Sentiment Analysis

The objective of performing sentiment analysis on the websites crawled is to better identify the information context and potentially help to narrow down whether the extracted entities contribute to the identification of malicious actors. This feature is currently in experimental mode as we are still researching and evaluating relevant existing works. The relevant works we have explored thus far include Intent Analysis/Classification, Contextual Semantic Search/Analysis, and Sentiment Summarization.

```
@DatasetReader.register("cyber2003")
class Cyber2003DatasetReader(Conll2003DatasetReader):
    """
    Registered as a `DatasetReader` with name "cyber2003".
    """

    @overrides
    def _read(self, file_path: PathOrStr) -> Iterable[Instance]:
        # if `file_path` is a URL, redirect to the cache
        file_path = cached_path(file_path)

        with open(file_path, "r", encoding="utf8") as data_file:
            logger.info("Reading instances from lines in file at: %s", file_path)
```

Figure 4. Dataset Reader for UTF8-encoded Data

For Intent Analysis, its use cases are geared towards scenarios like targeting a specific group of users to try to deliver the right advertisement to them. It does not fit into our use case as its classification is also meant more for chat bots. As for Contextual Semantic Analysis, it is a bidirectional LDA LSTM structure to generate embeddings to capture the semantics of the article and its search's use cases are more towards web search based on specific keywords. Sentiment Summarization is summarizing the text and then followed by a Sentiment Analysis on the summarized text. This is in contrary to what we want in getting as many sentiments out of an article as possible.

To determine the processes required in the sentiment analysis function, we referred to the whitepaper by Lexalytics [19] and based our design with the following steps:

a. Break the text crawled from the website into sentences.
b. Identify sentiment-bearing phrases and components.
c. Assign a sentiment score to each phrase or component.
d. Combine the scores to determine the overall sentiment of the text.
e. Remove the named entities extracted from the same text from the saved NER results if the sentiment score is low.

Our initial implementation of the sentiment analysis is separate from the NER implementation whereby each sentence or piece of text is determined to have a positive or negative score. Subsequently, we piggybacked it on the NER implementation such that for each of the sentences that contains the NER word, we run it against an SA model to get the sentiment of the sentence. A further enhancement has the model analyzing a whole paragraph instead of individual sentences to better determine the overall sentiment of the scraped article. We are expecting this to provide an accurate prediction of whether the concerned subject in the article is of malicious nature. Figure 5 depicts the program flow as described above and Figure 6 gives a sample snapshot of a table of NER and SA results.

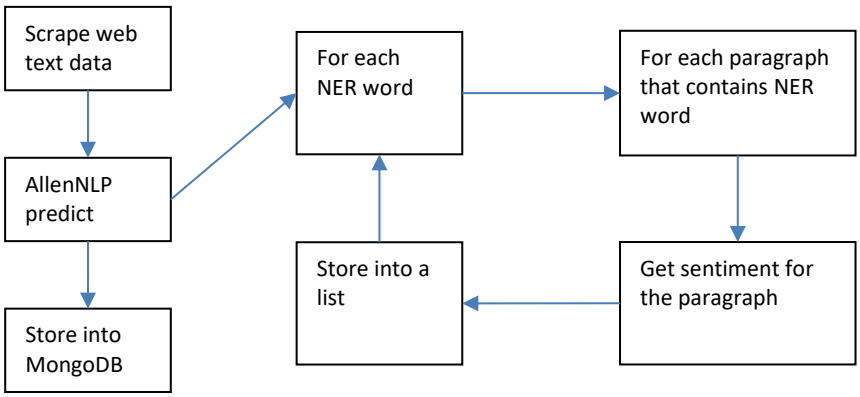

Figure 5. Sentiment Analysis Program Flow

NER	Sentence	Sentiment
TechRepublic	Procuring software packages for an organization is a complicated process that involves more than just technological knowledge. There are financial and support aspects to consider, proof of concepts to evaluate and vendor negotiations to handle. Navigating through the details of an RFP alone can be challenging, so use TechRepublic Premium's Software Procurement Policy to establish ...	POSITIVE
Comparitech	Though the final price for a cybercriminal's services is usually negotiated, personal attacks are the most expensive, says Comparitech.	NEGATIVE

Figure 6. Sample NER and SA results

3.8 Knowledge Graph Generation

The knowledge graph that is generated shows the associativity between the extracted entities (duplicates removed) as well as with the search parameters. The graph will start building its nodes from the search query and grows outwards based on the search results. It is formulated algorithmically by first adding the search parameters as individual root nodes. For IP and MAC addresses, they are connected to network information such as country, organization, and ISP. For other parameters, they are connected to the URLs returned by search engines which in turn are connected to named entities extracted by the NER model.

The blue node indicates the starting node representing the web crawler pointing to the search query terms marked by the yellow nodes. The red nodes are the returned web pages being scrapped and the grey nodes are the named entities extracted from those web pages.

The nodes would help the cybercrime investigator to recognize related entities that could potentially identify a malicious actor. One such example is Kevin Mitnik, a hacker who is associated with social engineering and phone phreaking. He would appear in the graph in relation to the terms he is linked to, such as the organizations with his involvement in cybercrime as shown in Figure 7.

Another example is to scour the Dark Web for information related to a specific hacking service or malware sale. A free text search on the phrase ‘hacking services’ using a dark web search engine can return an onion link that contains related named entities as depicted in Figure 8.

The web crawler and scraper in the backend system are written in Python, whereas the user

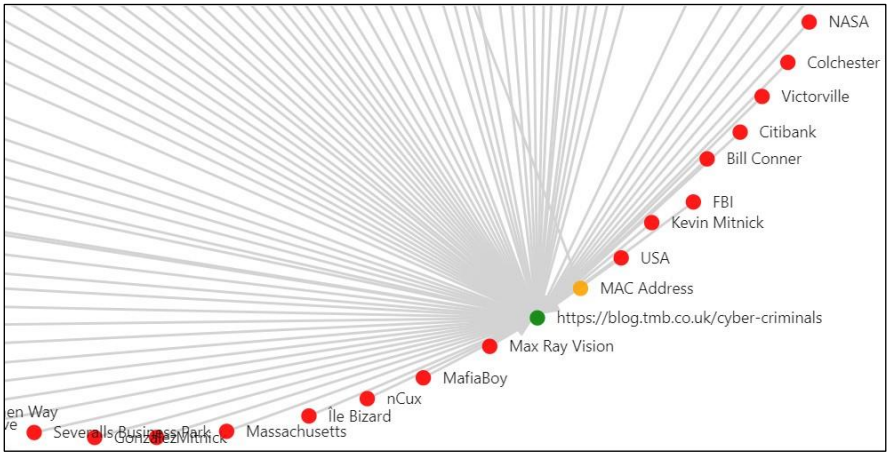

Figure 7. Knowledge Graph from Open Web search

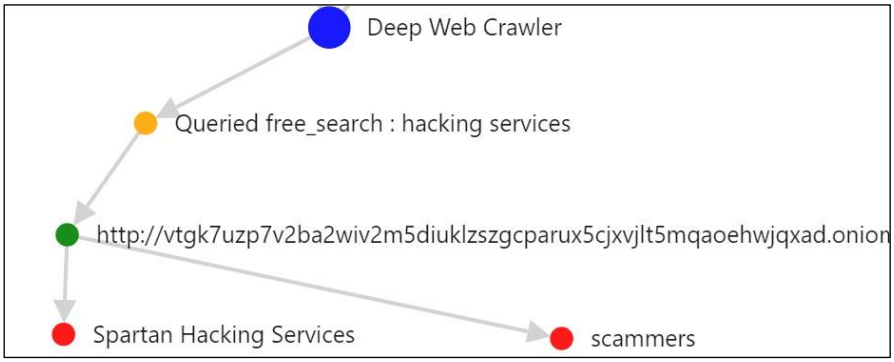

Figure 8. Knowledge Graph from Dark Web search

interface is created in ReactJS. To summarize, the various application packages and Python modules/libraries incorporated into the CTIH are as recorded in Tables 3 and 4, respectively. This is a real-time system processing real-world data with no simulation framework employed.

Table 3. Application packages used for development

Application Package	Feature/Function
MongoDB	NoSQL database
Tor for Windows	To access the Deep Web
ChromeDriver	For automation on the Chrome browser
React d3 graph	React component for drawing interactive graphs
Ant Design	For user interface design

Table 4. Python libraries used for development

Module/Library	Feature/Function
Beautiful Soup	For extracting data from HTML
Socks	For Tor proxy connection
Selenium	For scraping onion links, especially useful for dynamic websites
markdown	Converts HTML data to Markdown
html2text	Converts HTML to text
phonenumbers	Extracts phone numbers from text
AllenNLP	For NER to extract named entities

3.9 Machine Learning Tools

As briefly described in section 3.6 on Named Entity Recognition and section 3.7 on Sentiment Analysis, we have deployed the natural language processing library AllenNLP in place of the spaCy and Resume Parser libraries used in our previous design. AllenNLP allows for solving natural language processing tasks in PyTorch.

Besides having pre-trained NLP models such as a ELMo-based Named Entity Recognition model and a LSTM binary classifier with GloVe embeddings, it also provides a framework to train custom models based on our own datasets. In section 4.3, we will elaborate on using domain specific datasets to train different NER models, and in sections 7.1 and 7.2, we will discuss on refining the datasets and reconfiguring the models further.

3.10 Database Schema

We employ MongoDB [20] as our database to utilize its cross-platform and document-oriented nature. It uses JSON-like documents with optional schemas; hence we can dynamically add fields whenever a need arises. We have two document collections to store the results obtained from the web crawling and scraping, along with the recognized entities and analyzed sentiments. Table 5 gives the fields of a document for the raw data collected from web crawling while Table 6 gives the fields of a document for the scraped texts and the named entities extracted from them. These data are sufficient to generate the knowledge graph on-the-fly.

Table 5. Raw Data

Field	Description
Query	Array of search parameters
Date	Start date of search
Category	Specify crawling in open or dark web
Username	Array of web links returned from social media username search
Email	Array of web links returned from email search
Free_search	Array of web links returned from search engines
URL	Array of web links returned from URL
IP_add	IP information returned from third-party APIs
MAC_add	Device information returned from third-party API

Table 6. Scraped Data

Field	Description
Query	Array of search parameters
Date	Start date of scrapping
Category	Specify scrapping in open or dark web
Username	Array of named entities and sentiments from web pages returned from social media username search
Email	Array of named entities and sentiments from web pages returned from email search
Free_search	Array of named entities and sentiments from web pages returned from search engines
URL	Array of named entities and sentiments from web pages returned from URL
IP_add	IP information returned from third-party APIs
MAC_add	Device information returned from third-party API

4 SYSTEM ANALYSIS AND POTENTIAL INNOVATIONS

In this section, we are going to discuss about the developmental hurdles we have encountered, and the technological gaps we try to overcome, for the functions described in the architectural design in the previous section. We will also provide findings to the various methodologies explored and experimental runs using some of the technologies mentioned above.

4.1 Interfacing with Multiple Data Sources

Our new streamlined design of the CTIH is crafted to function as a standalone tool as well as an API module for compatible enterprise applications for cybersecurity analysis and digital forensics. It aims to incorporate not only manual inputs, but also auto-populated data obtained from other investigative tools or forensic sub-system modules. It can also export the resultant knowledge graph in JSON format for future analysis.

To pre-populate search parameters into the input fields of the search form, the system reads from the database for the availability of pre-existing form field data such as domain names, IP addresses, MAC addresses, etc. This information may be saved and tagged under a primary identifier by a separate forensic module. The identifier is then used by the system to query for the parameters to be populated and subsequently archive search results under it.

Currently, any module or application exchanging data with CTIH accesses its database directly to query or insert records. We are developing a REST API layer for third-party programs that will facilitate simpler integration and better automation.

4.2 Scraping the Deep Web

One of the difficulties in accessing a webpage in the Deep/Dark Web programmatically is in the determination of the wait time for the browser to connect to the Tor proxy and load the webpage. We set an estimated delay for the webpage to load before proceeding to scrape it. After numerous trial runs, we observed that a delay of three seconds is sufficient for most times, though we recommend a delay of five seconds as optimal.

For a browser to connect to the Tor proxy to access a deep webpage, it requires a corresponding driver for Selenium as well as an extension or plugin to enable the Tor connection. The readily available drivers and extensions are for Chrome, Firefox, and Microsoft Edge.

Though the Tor Project has its own browser with a built-in Tor proxy, it does not provide a method to enable it programmatically or allow Selenium to access it. Hence, we tested the various browser drivers and extensions, and opted to go with Chrome as we found our implementation with it to be more stable.

The program flow for scraping the deep webpage, as depicted in Figure 9, starts with instantiating the Chrome driver through a Selenium service. Upon getting the driver instance, we add the “Onion Browser Button” extension to it. As Chrome does not, and has no intention to, support extensions in headless mode, we must minimize the display window.

Unlike Firefox add-ons which are given different URLs during installations, Chrome extensions have static URLs that we can set in our code or configuration. We can retrieve the “Onion Browser Button” extension without having to find out its address and activate the HTML element to connect to the Tor proxy. This is the main reason we chose to go with the Chrome browser instead of Firefox.

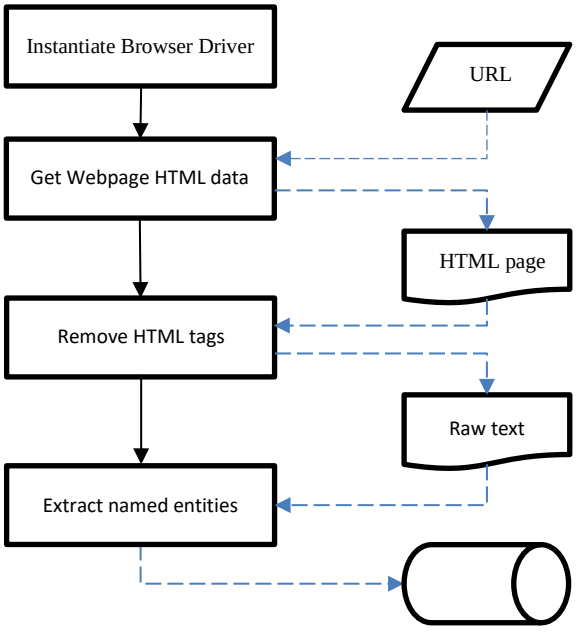

Figure 9. Web Scraping Program Flow

4.3 Training the NER Model

In the initial trial runs to familiarize with the use of AllenNLP, we depended on its demo configuration as a template. We would update the configuration to train domain-specific datasets with and without word embeddings.

AllenNLP also provides an NER model pretrained with ELMo on the standard CoNLL 2003 dataset. It has a F1 test score of 92.51 which we would use as a baseline indicator.

The F1 score is a machine learning metric that can be used to gauge, in our case, the accuracy of NER as well as SA models. It improves on the direct calculation of the percentage of correct predictions for accuracy, which suffers from imbalanced data. It is made up of two simpler performance metrics.

The first part of the F1 score is 'Precision' and it calculates the percentage of true positives that is correct within all predicted positives. The second part of the F1 score is 'Recall' and it calculates the percentage of true positives the model succeeds in finding within all cases that are correct (i.e. true positives and false negatives). The F1 score combines the two metrics by computing the average of precision and recall as shown in the formula below:

$$F1\ score = 2 * \frac{Precision * Recall}{Precison + Recall} \qquad (1)$$

We trained our models using cybersecurity-related datasets obtained from [21] and [22]. Each dataset has its own custom tags/labels for IP, URL, malware, vendor, etc. [21] provides a corpus of auto-labeled cybersecurity domain text. It includes all descriptions from CVE/NVD entries starting in 2010. [22] provides Cyber Threat Intelligence (CTI) reports dataset for NER.

As the raw datasets were given in JSON form with only the terms and named entity tags, we had to convert them to the CoNLL format for AllenNLP's dataset reader, filling null values for the part-of-speech tags and syntactic chunk tags. Then we split a dataset into three sets for training, validating, and evaluating, in the proportion of 7:1:2.

As recorded in Table 7, we could only manage to get F1 scores of 0.46 and 0.36 for [21] and [22] respectively. Since we are concerned with cybersecurity entities as whole terms without the need of context, we retrained our models without the ELMo word embeddings and gotten F1 scores of 0.45 and 0.27.

We are not able to cross evaluate the datasets due to the difference in the labels and the low scores for the domain related datasets could be the results of insufficient tagged data. Hence, to improve on the results will lead to future work that require building related up-to-date cybersecurity datasets to fine-tune the models.

Table 7. Evaluation of NER models based on datasets

(Pre-)Trained Model Dataset	F1 Measure
Standard CoNLL 2003	0.92
Auto Labeled Corpus	0.46
CTI Reports	0.36
Auto Labeled Corpus (w/o ELMo)	0.45
CTI Reports (w/o ELMo)	0.27

5 LIMITATIONS

We have encountered two defining limitations during our research and development work that affect the accuracy and scope of extracting relevant information. Our design also lacks the support for analyzing data from mobile platforms and network streams.

More research is also needed on the sentiment analysis portion of the machine learning module. We want to further eliminate extracted named entities that will falsely identify an actor as malicious, thus improving identification accuracy.

5.1 Lack of Substantial Training Dataset

Our named entity extractions rely on a natural language processing model that is trained for the cybersecurity domain. However, we are facing difficulties in obtaining timely-tagged, domain-specific datasets of sizable volume. The dated training datasets of [21] and [22] do result in misclassified data or missed identification at times.

To keep pace with the emergence of terms referring to new vulnerabilities, malware, and malicious attacks, we will be looking into possibly utilizing text annotation tools such as Doccano [23] to create our own labelled data.

We will also be exploring further on the use of domain-specific word embeddings such as the one given in [24] to improve the accuracy.

5.2 Volatility of Deep Web Links

Navigating the Deep Web is a challenge, as the onion services and websites are not indexed by the usual search engines or regulated by any organizations. We make use of dark net search engines that do index onion sites as well as directories that list commonly known sites to access sites which we deploy a smart crawler on.

However, the nature of Deep Web sites wanting to preserve anonymity and secrecy means the links change their addresses frequently. The results returned by the search engines will often be outdated. Even the search engines themselves such as NotEvil, Candle, and Torch are unreachable at the point of writing. Only Ahmia and DuckDuckGo are functioning. Directories like dark.fail will indicate whether some websites are offline or not but the validity of that information is subjected to how recently the directories are updated.

Furthermore, version 2 of onion sites is now being deprecated so we must prepare for an update to the hardcoded search engines' links to version 3 in our web crawler soon.

5.3 No Mobile Platform and Network Data Stream Support

Unlike [25], our design is currently unable to differentiate data attributes generated on mobile platforms and is not capable of being deployed on mobile devices. We also are unable to process live network data streams to observe for distributed denial of service (DDOS) attacks as performed by [26].

6 CONCLUSION

In this paper, we have presented a preliminary prototype of a crime and threat intelligence hunting engine that makes use of machine learning to progressively improve the accuracy of its information acquisition. Preliminary testing indicates the need for more comprehensive datasets to train the machine learning module.

Its streamlined design with the use of open-source technologies makes it a cost-effective, plug-and-play framework for evidence acquisition and threat prediction. Its primary use is as an automated intelligence acquisition and risk management support for smart digital forensics and incident response.

Table 8. AllenNLP-based scoring for CTIH models

	NER	SA
Accuracy	0.95-0.97	0.64
Precision	0.67	-
Recall	0.73	-
F1-measure	0.70	-
Loss	224.70	0.65

7 FURTHER WORK

Due to the lack of substantial training datasets as mentioned in the previous section, we have proceeded to generate our own domain-specific datasets to train customized NER and SA models for cybersecurity using AllenNLP. Table 8 summarizes the evaluation results of the trained models, and the following sub-sections describe how we generate the datasets to train them.

7.1 NER Dataset and Model

To create a dataset tailored specifically to train a NER model for the cybersecurity domain, we first must determine the entity types we want to extract. We came up with a list of 12 types for the entities, namely PER for person, ORG for organization, LOC for location, APT for advanced persistent threat, and CYBERSEC for non-specific cybersecurity terms, along with the self-explanatory EXPLOITS, MALWARE, VENDOR, DEVICES, NETWORK, PATH, and COMMANDS tags.

By collecting textual information from open web knowledge bases such as MITRE ATT&CK® as well as dark web forums, we managed to scrape a sufficiently large amount of raw data to divide the dataset into training, validation, and evaluation sets in the ratio of 7:1.5:1.5. The dataset is tagged in the CoNLL2003 IOB format for a count of 7206 I-tags and B-tags out of 66475 tags. The breakdown of the tags in terms of the different entity types is tabulated in Table 9.

We passed the model through AllenNLP's evaluate function with the evaluation dataset and gotten an overall F1 score of 0.70.

Table 9. Breakdown of tag count in NER dataset

Entity Types	I-Tag	B-Tag	Total
PER	305	955	1260
ORG	82	359	441
LOC	14	7	21
EXPLOITS	1076	881	1957
MALWARE	219	263	482
VENDOR	38	141	179
DEVICES	196	436	632
NETWORK	444	504	948
PATH	49	166	215
COMMANDS	110	158	268
APT	65	49	114
CYBERSEC	320	369	689

7.2 SA Dataset and Model

Like collecting raw data in NER, we once again turned to the knowledge bases and web forums such as Reddit to get text snippets for the training of our SA model. The text snippets were tokenized into words, keywords, phrases, or symbols. This was done to facilitate the normalization stage whereby the data is sanitized to remove special characters and stop words not needed for analysis. This would aid in the process of calculating the sentiment values.

To start labeling the sentences and phrases as either having positive or negative sentiments, we first identified a dictionary of positive and negative keywords. This made it easy for us to label some of the simpler sentences or phrases before proceeding to the harder samples with a better comprehension of the context. If a phrase is too short to derive the context and thus assign a sentiment, it is grouped with the phrases before and/or after.

We have extracted a total of 20278 samples: 7980 positives and 12298 negatives. The dataset is divided into training, validation, and evaluation sets in the ratio of 7:1.5:1.5. Then we passed the trained model through AllenNLP's evaluate function with the evaluation dataset and gotten an accuracy score of 0.64.

7.3 Potential Enhancements

We continue to fine-tune the models by enriching the datasets given user feedback and new texts from recurring scraping of the web. We would also retrain the models with different configurations such as varying the dataset format, the token embedder, token indexer, or the number of epochs.

ACKNOWLEDGEMENTS

The above research is funded in part by the National Research Foundation of Singapore - National Cybersecurity Research and Development Grant GC2018-NCR-0008.

REFERENCES

1. Ng, J., Loh, P.: Enhanced Crime and Threat Intelligence Hunter. ICSCS 2022: 2nd International Conference on Soft Computing for Security Applications. (2022)
2. Perlman, A.: The Growing Role of Machine Learning in Cybersecurity. https://www.securityroundtable.org/the-growing-role-of-machine-learning-in-cybersecurity/
3. Dutta, A., Kant, S.: An Overview of Cyber Threat Intelligence Platform and Role of Artificial Intelligence and Machine Learning. In: Kanhere S., Patil V.T., Sural S., Gaur M.S. (eds) Information Systems Security. ICISS 2020. Lecture Notes in Computer Science, vol 12553. Springer, Cham. (2020)
4. Shameel, M., Ng, J., Loh, P.: A Machine Learning – Assisted Crime and Threat Intelligence Hunter. ICCSM 2021: International Conference on Cybersecurity and Security Management. (2021)
5. Maltego Technologies, https://www.maltego.com
6. Martorella, C.: theHarvester: E-Mails, Subdomains and Names Harvester - OSINT. https://github.com/laramies/theHarvester
7. Nyeste, P.: The use of the open source intelligence in the criminal investigations. CASOPIS NAUOA-SERIA PRAVO 21.1: 1-10. (2020)
8. Chahal, B.: Reconspider: Most Advanced Open Source Intelligence (OSINT) Framework For Scanning IP Address, Emails, Websites, Organizations. https://github.com/bhavsec/reconspider
9. OSINT Framework, https://osintframework.com/
10. Narayanan, P. S., Ani, R., King, A.: TorBot: Open Source Intelligence Tool for Dark Web. Inventive Communication and Computational Technologies: 187-195. (2020)
11. King, A., Narayanan, P. S.: Gotor: This Program Provides Efficient Web Scraping Services For Tor And Non-Tor Sites. https://github.com/DedSecInside/gotor
12. Adewopo, V., Gonen, B., Adewopo F.: Exploring Open Source Information for Cyber Threat Intelligence. 2020 IEEE International Conference on Big Data (Big Data), pp. 2232-2241. (2020)
13. Deb, A., Lerman, K., Ferrara E.: Predicting cyber-events by leveraging hacker sentiment. Information

9.11: 280. (2018)
14. CLiPS: Pattern, https://github.com/clips/pattern
15. MITRE ATT&CK®, https://attack.mitre.org/
16. spaCy · Industrial-strength Natural Language Processing in Python, https://spacy.io/
17. Rajwani, K., Njoroge B.: Resume_Parser. https://github.com/kbrajwani/resume_parser
18. AllenNLP, https://allennlp.org/
19. Sentiment Analysis | Lexalytics, https://www.lexalytics.com/technology/sentiment-analysis
20. MongoDB, https://www.mongodb.com/
21. Bridges, Robert A., et al.: Automatic labeling for entity extraction in cyber security. arXiv preprint arXiv:1308.4941 (2013)
22. Kim, Gyeongmin, et al.: Automatic extraction of named entities of cyber threats using a deep Bi-LSTM-CRF network. International journal of machine learning and cybernetics 11.10: 2341-2355. (2020)
23. Wolff, Clemens et al.: Doccano: Open Source Annotation Tool For Machine Learning Practitioners. https://github.com/doccano/doccano
24. Mumtaz, Sara, et al: Learning word representation for the cyber security vulnerability domain. 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020
25. Vivekanandam, B. "Design an Adaptive Hybrid Approach for Genetic Algorithm to Detect Effective Malware Detection in Android Division." Journal of Ubiquitous Computing and Communication Technologies 3, no. 2 (2021): 135-149
26. Mugunthan, S. R. "Soft computing based autonomous low rate DDOS attack detection and security for cloud computing." J. Soft Comput. Paradig.(JSCP) 1, no. 02 (2019): 80-90