TwitterOSINT: Automated Cybersecurity Threat Intelligence Collection and Analysis using Twitter Data

Satyanarayan Raju Vadapalli
Computer Science
Norfolk State University
Norfolk, VA, USA
s.vadapalli@spartans.nsu.edu

George Hsieh
Computer Science
Norfolk State University
Norfolk, VA, USA
ghsieh@nsu.edu

Kevin S. Nauer
Computer & Network Security
Sandia National Labs
Albuquerque, NM, USA
ksnauer@sandia.gov

***Abstract*—To defend against the ever increasingly frequent, varied and sophisticated cyber-attacks, timely analysis of cybersecurity threat information is critical. Open Source Intelligence (OSINT) is a method of collecting information from publicly available sources and then analyzing it to derive actionable intelligence. This paper presents the design and implementation of a prototype system named TwitterOSINT for automating the collection and analysis of cybersecurity related information (e.g., threats and vulnerabilities) posted on Twitter which serves as an OSINT source. The prototype system was implemented in Java and it used Twitter Streaming API to download relevant tweets based on a set of user provided keywords. The selected tweets were next processed and analyzed by a Natural Language Processing (NLP) module which used the Stanford CoreNLP library and language models for foundational NLP capabilities, and the Stucco cyber domain-specific entity extraction library developed and provided by the Oak Ridge National Lab. The processed tweets were then stored in a JavaScript Object Notation (JSON) formatted file along with all the annotations produced by the NLP module for these tweets. Next, the TwitterOSINT system used the Elastic Stack (Elasticsearch, Logstash and Kibana) to collect, index, store, analyze, manage, and visualize the annotated tweets, and help derive intelligence using the built-in data analytic and machine learning capabilities provided by the Elastic Stack. A preliminary experiment was conducted to gain operational experiences and insights with this system.**

***Keywords*—Cyber intelligence, OSINT, Twitter, natural language processing, Elastic Stack**

I. INTRODUCTION

The impact of cyber-crime has necessitated the government and private-sector organizations across the world to tackle cyber threats. All sectors are now facing similar dilemmas of how to best mitigate against cyber-crime and implement best practices effectively. Extracting actionable and high value intelligence by harvesting public information is emerging rapidly as an important means for cyber defense. As the amount of information available from open sources rapidly increases, countering cyber-crime increasingly depends upon advanced software tools and techniques to collect, process, analyze and leverage the information in an effective and efficient manner.

Open Source Intelligence (OSINT) refers to intelligence that has been derived from publicly available sources such as news articles, blogs, and social media. Using automated OSINT collection and analysis tools and methods, government and private sectors alike can be better prepared to avert cyber-attacks before they hit them or mitigate and respond to incidents more effectively.

As an example, Twitter has become an obvious and very important source to gather cybersecurity information, as many professionals in this field use this social media to share information on threats and attacks as they happen, and to observe trends in the cybersecurity landscape. Twitter makes it possible and convenient for those interested to get on-the-fly notice of cyber events.

However, manual methods for cybersecurity OSINT collection and analysis are inadequate due to the volume of information to be analyzed and the speed with which threats arise. Unlike traditional cybersecurity products designed based on known attacks, next-generation security intelligence systems must provide real-time alerts and permit immediate intelligence to be gathered about a threat.

To help automate the cybersecurity OSINT collection and analysis process, we designed and implemented a prototype system named TwitterOSINT for gathering and analyzing cybersecurity related information (e.g., threats, vulnerabilities and attacks) posted on Twitter.

The prototype system was implemented in Java and it used Twitter Streaming API to download relevant tweets based on a set of user provided keywords. To process and analyze the tweets which are unstructured texts, the TwitterOSINT system implemented a Natural Language Processing (NLP) pipeline using the Stanford CoreNLP library and language models for foundational NLP capabilities, and the Stucco cyber domain-specific entity extraction library developed and provided by the Oak Ridge National Lab (ORNL) to extract cyber-domain entities from unstructured texts. Both libraries use advanced machine learning and statistical techniques to provide high performance text analytic capabilities.

The processed tweets were then stored in a JavaScript Object Notation (JSON), a lightweight and human readable data-interchange format, file along with all the annotations (e.g., date, username, text, and cyber-domain entities) produced by the NLP pipeline for these tweets.

Next, the TwitterOSINT system used the Elastic Stack (Elasticsearch, Logstash and Kibana) to collect, index, store, analyze, manage, and visualize the annotated tweets, and help derive intelligence using the built-in data analytic and machine learning capabilities provided by the Elastic Stack. A

This work was supported in part by DoD (award #FA8750-15-2-0120), and DoE/NNSA (award #DE-NAOO02686).

preliminary experiment was conducted to gain operational experiences and insights with the system.

The remainder of this paper is organized as follows. Section II provides background information about the key technologies used for the TwitterOSINT prototype system. Section III describes the design and implementation of the system. Section IV discusses how to run and use the TwitterOSINT system and our preliminary experiment. Section V concludes the paper with a brief summary and discussion of future work.

II. Background

In this section, we provide background information about Twitter Streaming API, natural language processing and Elastic Stack.

A. *Twitter Streaming API*

The TwitterOSINT system utilized the Twitter Streaming API [1], as illustrated in Fig. 1 [2], which provides low-latency access to Twitter's global stream of tweets and download those that were considered relevant to cybersecurity.

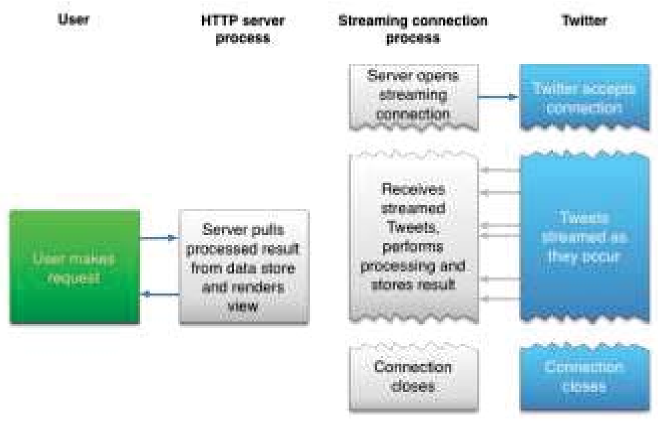

User
HTTP server process
Streaming connection process
Twitter
Server opens streaming connection
Twitter accepts connection
Receives streamed Tweets, performs processing and stores result
Tweets streamed as they occur
User makes request
Server pulls processed result from data store and renders view
Connection closes
Connection closes

Fig. 1. Twitter Streaming API [2].

A streaming client pushes messages indicating tweets and other events have occurred based on user-specified keywords. Connecting to the streaming API requires keeping a persistent HTTP connection open. The code for maintaining the streaming connection is typically run in a process separate from the process that handles HTTP requests. The streaming process gets input tweets and performs any parsing, filtering, and/or aggregation needed before passing the results to a data store. The HTTP handling process queries the data store for results in response to user requests.

B. *Natural Language Processing*

According to industry estimates, only 21% of the available data is present in structured form [3]. In order to produce significant and actionable insights from the unstructured text data, Natural Language Processing techniques and tools are developed to analyze, understand, and derive information from the text data in an intelligent and efficient manner.

The field of NLP aims to convert human language into a formal representation that is easy for computers to manipulate. NLP has been used to perform numerous automated tasks in a wide variety of applications including machine translation, automatic summarization, named entity recognition, sentiment analysis, etc. [3].

While complete semantic understanding by machine is still a distant goal, researchers have taken a divide and conquer approach and identified several NLP sub-tasks useful for application development and analysis. These range from the syntactic – such as Part-of-Speech (POS) tagging, chunking and parsing – to the semantic – such as word sense disambiguation, semantic-role labeling, named entity extraction and anaphora resolution [4].

Two problems in particular make NLP difficult and require techniques different from those associated with processing artificial languages (e.g., compiler construction). These problems are: the level of ambiguity that exists in natural languages; and the complexity of semantic information contained in even simple sentences.

Typically, language processors deal with a large number of words, many of which have alternative usages, and large grammars that allow different phrase types to be formed from the same string of words. NLP is made more complex because of the irregularities of language and the different kinds of ambiguity that can occur [5].

B. *NLP Tasks*

The most common tasks in the NLP pipeline, as shown in Fig. 2 [6], are:

- POS Tagging (POS): labels each word with a unique tag that indicates its syntactic role (e.g., plural noun, adverb).
- Chunking: labels segments of a sentence with syntactic constituents such as noun phrase (NP) or verb phrase (VP). Each word is assigned only one unique tag.
- Named Entity Recognition (NER): labels atomic elements in the sentence using categories such as PERSON, COMPANY, or LOCATION.
- Semantic Role Labeling (SRL): assigns a semantic role to each syntactic constituent of a sentence.
- Language Models: traditionally estimate the probability of a word *w* being the next word in a sequence.
- Semantically Related Words (e.g., synonyms, homonyms, hypernyms): are measured using the WordNet databases as ground truth. This task involves predicting whether two words are semantically related.

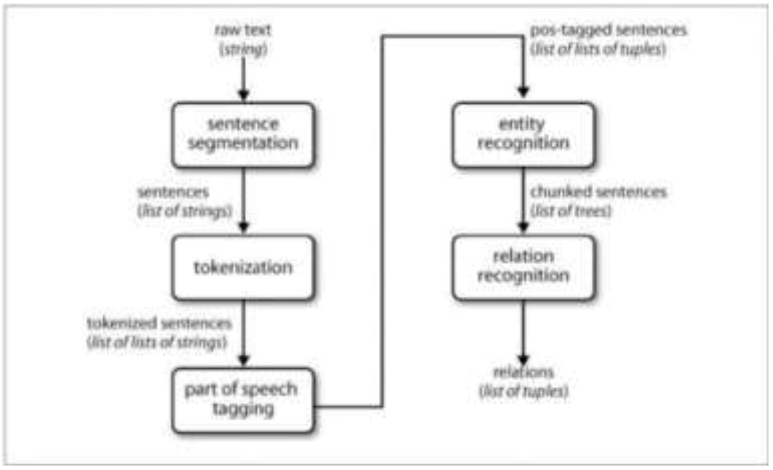

raw text (string)
sentence segmentation
sentences (list of strings)
tokenization
tokenized sentences (list of lists of strings)
part of speech tagging
pos-tagged sentences (list of lists of tuples)
entity recognition
chunked sentences (list of trees)
relation recognition
relations (list of tuples)

Fig. 2. NLP Pipeline [6].

C. Stanford CoreNLP

The Stanford CoreNLP [7] is one of the most widely used NLP software worldwide. It provides a set of human language technology tools. It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and syntactic dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract particular or open-class relations between entity mentions, get the quotes people said, etc. [7].

The Stanford CoreNLP is designed to make it very flexible, extensible, and easy to apply a collection of linguistic analysis tools to a piece of text. A tool pipeline can be run on a piece of plain text with just two lines of code. Stanford CoreNLP integrates many of Stanford's NLP tools, including the part-of-speech (POS) tagger, the named entity recognizer (NER), the parser, the coreference resolution system, sentiment analysis, bootstrapped pattern learning, and the open information extraction tools. Moreover, an annotator pipeline can include additional custom or third-party annotators. CoreNLP's analyses provide the foundational building blocks for higher-level and domain-specific text understanding applications [7].

D. Entity Extraction for Cyber Domain

Although the Stanford CoreNLP is a state-of-the-art NLP software, it did not provide very good performance for entity extraction from unstructured text in the cyber domain because it was not trained with cyber domain-specific data.

The Stucco Entity-Extractor library, developed by ORNL, was designed to identify and label cyber-domain entities from unstructured text [8] [9] [10]. This library makes use of the Apache OpenNLP [11] library, and the Stanford CoreNLP library to tokenize, part-of-speech tag, and build the parse trees of the unstructured text.

The entity types identified and labelled by the Stucco Entity-Extractor library include [10]:

- Software: Vendor, Product, Version
- File: Name
- Function: Name
- Vulnerability: Name, Description, CVE, MS

For an unstructured text, the Entity-Extractor library produces an Annotation object that represents the unstructured text as a map which includes:

- Text: original raw text
- Sentences: list of sentences
    - o Sentence: map representing one sentence
        - Token: word within the sentence
        - POSTag: part-of-speech tag
        - CyberEntity: cyber domain label for the token
    - o ParseTree: sentence structure as a tree

As an example, Fig. 3 [8] shows a part of the National Vulnerability Database (NVD) text description of the Common Vulnerability Enumeration CVE-2012-0678 with automatically generated labels. Note the cyber-domain annotations such as Software Vendor: Apple and Software Product: Safari.

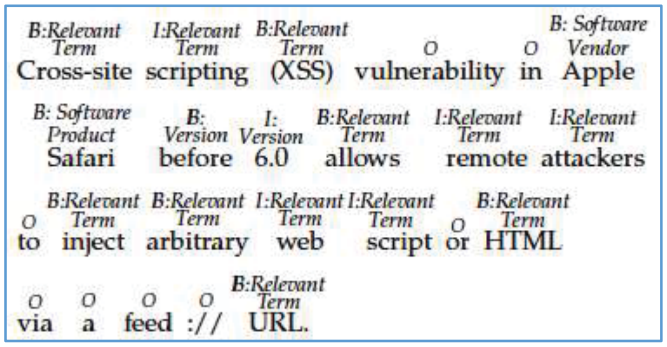


Fig. 3. Example text with automatically generated labels [8].

After a tweet was analyzed by the NLP pipeline, the TwitterOSINT system appended the tweet's text along with its user name, date, and all the annotations to a JSON file, ready for further processing and analysis.

E. Elastic Stack

The TwitterOSINT system utilized the Elastic Stack [12] which is a scalable, flexible, and feature rich platform enabling users to reliably and securely take data from any source in any format and search, analyze, and visualize it in real time.

Fig. 4 shows the key components of the Elastic Stack:

- Logstash: an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to a "stash" (which is Elasticsearch if using the Elastic Stack).
- Elasticsearch: a distributed, RESTful search and analytics engine for indexing, storing, and analyzing data.
- Kibana: UI to visualize Elasticsearch data and navigate the Elastic Stack.

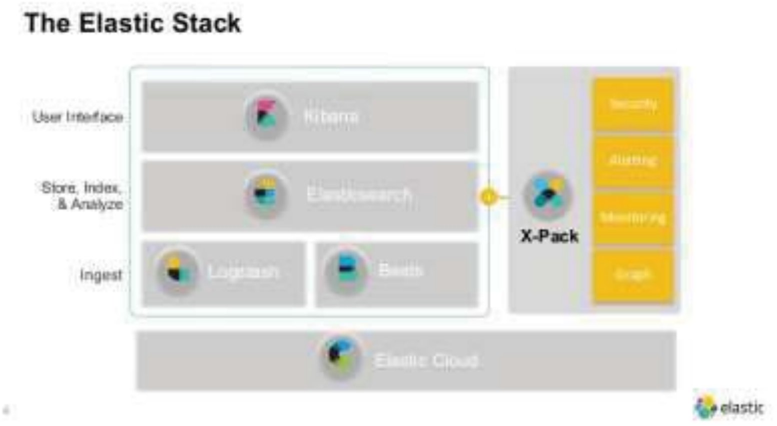


Fig. 4. Key components of the Elastic Stack.

The TwitterOSINT system used the Logstash to ingest the JSON file created for the selected tweets, and feed the content to Elasticsearch which then indexed and stored the data into the centrally managed data store.

To search and analyze the indexed tweets to gain further insights and derive actionable intelligence, a human analyst can use Elasticsearch and Kibana to execute search queries or create visualizations. The X-Pact for the Elastic Stack can be further utilized to provide additional data analytic and task automation capabilities including machine learning, graph analytics, and alerting.

III. TWITTEROSINT DESIGN AND IMPLEMENTATION

In this section, we describe the design and implementation of the TwitterOSINT prototype system.

A. System Architeture

TwitterOSINT brought together NLP, machine learning techniques and the Elastic Stack to provide a comprehensive data analytic environment to collect, store, analyze, and visualize tweets that were considered relevant to cybersecurity. The flowchart in Fig. 10 at the end of the paper shows the architecture of the TwitterOSINT system and the transformation of Twitter data at each step.

B. Development Environments and Tools

The TwitterOSINT prototype system was developed using Eclipse Oxygen Release 4.7.0 IDE, and Java Version 8 Update 151. It was initially implemented on a Dell Power Edge 730xd enterprise-grade server running CentOS 7. It was later ported onto Windows 7 workstations.

The system was built with the following major application-specific dependencies:

- twitter4j-4.0.4
- stanford-corenlp-3.4.1
- stanford-corenlp-3.4.1-models
- entity-extractor-1.0.0
- opennlp-tools-1.6.0
- jackson-databind-2.7.0

C. TwitterOSINT Java Application

TwitterOSINT was the main Java application comprised of two main class objects: Stream and annotator.

- **Stream.java** contains the main class that streams the tweets using the Twitter Streaming API, based on the cybersecurity keywords specified in a text file. It removes special characters and URLs and then passes the result onto the annotator class for further NLP and entity extraction. The Stream class is also responsible for creating JSON objects with the tweet date, username, text and annotations returned by the NLP and Entity Extraction pipeline. These JSON objects are next written into a file named twitDB.json.
- **annotator.java** contains the class that uses the entity-extraction module for tokenizing, POS tagging and adding annotations to each token. The output from the annotator class is a JSON object with key-value pairs, where keys are the field labels (e.g. token, POS, CyberAnnot) and values are the field values.

IV. RUNNING AND USING THE SYSTEM

In this section, we first discuss how to run the TwitterOSINT system along with a few examples to show how to use the system for cybersecurity operations. We then discuss the preliminary experiment conducted to gain operational experiences and insights about the system and its capabilities, and our observations.

A. Running TwitterOSINT System

To run the TwitterOSINT system, start the Stream.java application, Logstash, Elasticsearch and Kibana services. Keep them running.

The Stream.java application maintains a connection to Twitter, to download the relevant tweets. It then uses the NLP pipeline to analyze the tweets, and store the texts with the automatically generated annotations in the twitDB.json file.

The Logstash service is configured to digest the twitDB.json file and forward the data to Elasticsearch which then indexes and stores the tweets along with their annotations in the centrally managed data store.

The UI provided by the Kibana service is used to create visualizations of the data stored in Elasticsearch, navigate the Elastic Stack, and execute commands.

B. Sample Data

The textbox below shows a sample tweet (with the date, username, and the text of the tweet) which was processed by the TwitterOSINT system.

```
▸September 4th 2017, 03:55:11.078       cyberinbox
   Google removed almost 300 Android apps involved in
   DDoS attack ps://t.co/UpZNV96PRK
```

A section of the annotations generated for this tweet's text is shown below using the JSON format. The text is broken down into tokens. Each token is tagged with three fields: CyberAnnot (which denotes the cyber entity: vulnerability, software vendor, software product, etc. or "O" if not a cyber-related entity), POS (which denotes the part-of-speech), and the token itself.

```
"annotations":[{"CyberAnnot":"Sw.vendor","POS":"NNP",
"token":"Google"},{"CyberAnnot":"O","POS":"VBD","token":
"removed"},{"CyberAnnot":"O","POS":"RB","token":"almost"}
,{"CyberAnnot":"Sw.product","POS":"JJ","token":"Android"}
...
{"CyberAnnot":"vuln.description","POS":"NN","token":"DDoS
"},{"CyberAnnot":"vuln.description","POS":"NN","token":
"attack"}
...
```

C. Data Visualization

Manually examining a large number of indexed tweets is not practical for an analyst. To illustrate how the power and ease-of-use of the Elastic Stack can be utilized to help analysts define and execute tasks of high interest and value to them in an automated manner, the Elastic Stack was used to create five visualizations to make it easier to gather and view the aggregated or related information.

Elasticsearch provides a feature named *Significant Terms Aggregation* [13] which is an aggregation that returns interesting or unusual occurrences of terms in a set. The terms selected are not simply the most popular terms in a set. They are the terms that have undergone a significant change in popularity when measured between a *foreground* (e.g., tweets indexed today) set and a *background* (tweets indexed until but not including today) set.

For example, suppose the term *Ransomware* existed in only 5 out of 10 million tweets indexed previously, and yet it is found in 4 of the 100 tweets that were indexed today. Such a big shift in frequency (5/10,000,000 or 0.000005% versus 4/100 or 4%) is significant, and it likely indicates a need for further investigation [13]. It is common that a current threat, or a software product affected by a threat, would appear in the day's tweets with a higher than normal frequency.

We built a pie chart [14], as shown in Fig. 5, to show the distribution of the token words in the tweets aggregating on the Most Significant Terms of a day. This chart can help analysts to more quickly spot significant increases in frequencies of cyber threat and vulnerability terms appearing in tweets, e.g., on a daily basis.

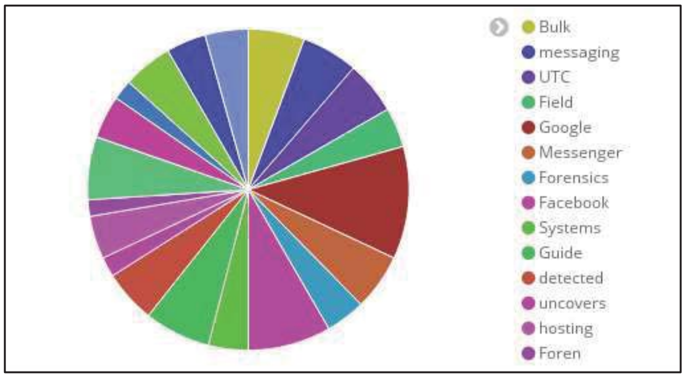


Fig. 5. Significant terms distribution (pie chart).

A *tag cloud* is a visual representation of free-form text [14]. Tags are usually single words, and the importance of each tag is shown with font size and/or color as determined by the *metrics* aggregation. Significant Term Aggregation was used to build a tag cloud for TwitterOSINT. The purpose of this tag cloud is to represent the most significant words of the day, and to provide one-click access to all tweets that contain a particular word of interest. For example, in the tag cloud shown in Fig. 6, clicking on *Google* would display all the indexed tweets that contained the word *Google* on that day.

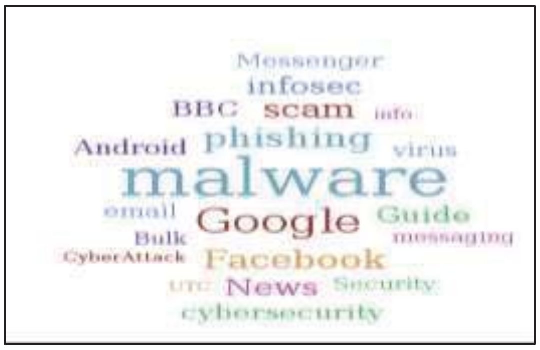


Fig. 6. Significant terms aggregation (tag cloud).

A *count* aggregation returns a raw count of the elements in the selected index pattern. This visualization provides a quick view of the number of relevant tweets of the day. When a word is clicked on the tag cloud to display the tweets containing that word, the Count matrix displays the number of tweets returned by the query, as illustrated in Fig. 7.

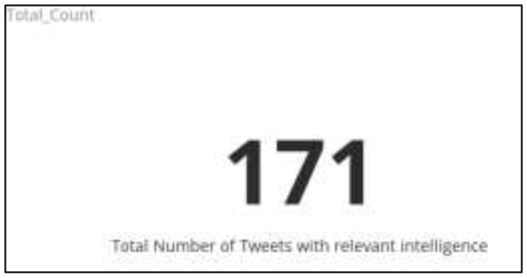
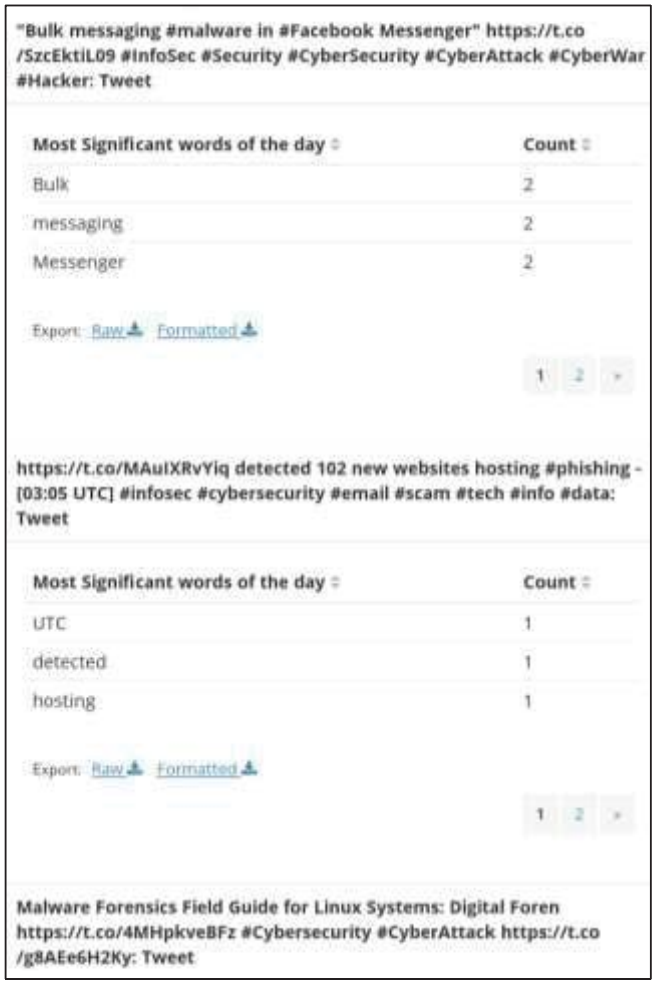


Fig. 7. Total count of tweets with relevant information.

The last two Kibana-generated visualizations that we built were displayed as tables. The first table, as shown in Fig. 8, displays the tweets containing the top 20 significant terms of the day. The second table, as shown in Fig. 9, contains all the relevant tweets of the day. These tables provide a quick view of the day's tweets. The tables could be modified to show the tweets of interest, based on clicking a word in the tag cloud.



"Bulk messaging #malware in #Facebook Messenger" https://t.co/SzcEktiL09 #InfoSec #Security #CyberSecurity #CyberAttack #CyberWar #Hacker: Tweet

Most Significant words of the day	Count
Bulk	2
messaging	2
Messenger	2

Export: Raw Formatted

1 2 »

https://t.co/MAuIXRvYiq detected 102 new websites hosting #phishing - [03:05 UTC] #infosec #cybersecurity #email #scam #tech #info #data: Tweet

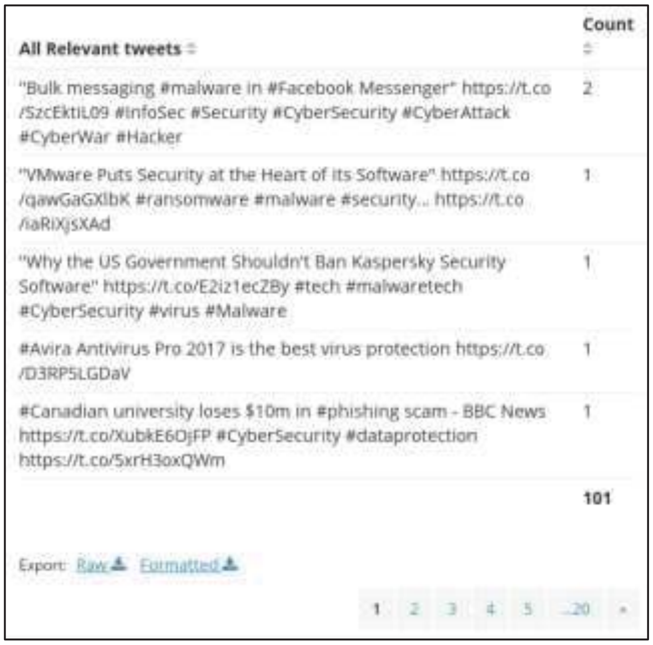
Most Significant words of the day	Count
UTC	1
detected	1
hosting	1

Export: Raw Formatted

1 2 »

Malware Forensics Field Guide for Linux Systems: Digital Foren https://t.co/4MHpkveBFz #Cybersecurity #CyberAttack https://t.co/g8AEe6H2Ky: Tweet

Fig. 8. Tweets with top 20 significant terms.



All Relevant tweets	Count
"Bulk messaging #malware in #Facebook Messenger" https://t.co/SzcEktiL09 #InfoSec #Security #CyberSecurity #CyberAttack #CyberWar #Hacker	2
"VMware Puts Security at the Heart of its Software" https://t.co/qawGaGXlbK #ransomware #malware #security... https://t.co/iaRIXjsXAd	1
"Why the US Government Shouldn't Ban Kaspersky Security Software" https://t.co/E2iz1ecZBy #tech #malwaretech #CyberSecurity #virus #Malware	1
#Avira Antivirus Pro 2017 is the best virus protection https://t.co/D3RP5LGDaV	1
#Canadian university loses $10m in #phishing scam - BBC News https://t.co/XubkE6OjFP #CyberSecurity #dataprotection https://t.co/5xrH3oxQWm	1
	101

Export: Raw Formatted

1 2 3 4 5 ...20 »

Fig. 9. All relevant tweets.

D. Experiment

We conducted a very preliminary experiment to assess the efficacy of the TwitterOSINT system in terms of tagging tweets with information relevant to the cyber domain.

During this experiment, three groups consisting of 500 randomly chosen tweets each, were chosen. The tweets were manually studied to identify the properties of a tweet with relevant data. The properties are depicted by the presence or absence of cybersecurity entities in the form of annotations within the JSON objects. Results are summarized in TABLE 1.

TABLE 1. TWEET SAMPLING RESULTS

	No of tweets	*tweets with relevant information (a)*	*Relevant tweets with all 3 Cyber entities (b)*	*Relevant tweets without one or more Cyber entities (c)*	*No of tweets in (c) where the information is covered in (a)*
	500	18	14	4	2
	500	12	9	3	1
	500	21	17	4	2
Total	1500	51	40	11	5
Percentages	100%	3.4% - of sample 100% - of relevant tweets	78.43%	21.56%	9.8%

The experiment observed that 51 of the 1500 (3.4%) sampled tweets were relevant, meaning they contained some interesting information related to a vulnerability. 78% (40/51) of those tweets had a vulnerability description, a software vendor and/or a software product which is affected by the vulnerability. In the remaining 22% of the tweets with relevant information, one or more of these fields was missing; however, about half the time the information they did contain was already covered in the other 78% of relevant tweets.

For this experiment, the observed average daily unfiltered tweet count was about 10,000 tweets. We developed a rule in Kibana to filter down daily tweets to show only those tweets that contain the three entities: *vulnerability*, *software vendor* and/or *software product*. This rule removed most of the irrelevant tweets, so that on average 130 tweets remained on the filtered-down list. The rule also removed a small percentage (~12%) of tweets that did in fact contain relevant information. Also, there was a small percentage of *false-positive tweets* in the filtered down list. False-positives are tweets that do not actually contain information pertaining to cybersecurity, or the information they contain is simply informational and not about a recent cybersecurity threat.

V. Summary

This paper described the design and implementation of a TwitterOSINT prototype system that can help analysts gather and derive cybersecurity intelligence using open source data from Twitter.

The TwitterOSINT prototype system was implemented using the Twitter Streaming API to collect tweets that contain specified keywords related to cybersecurity. It used the Stanford CoreNLP library and the Stucco cyber-domain specific entity extraction library from Oak Ridge National Lab to analyze the unstructured tweet text and automatically classify and label cyber related entities from the texts. The system also used the Elastic Stack to store, search and visualize the annotated tweets.

One major challenge in using a data source like Twitter for cyber intelligence lies in the accurate classification and labeling of domain-specific entities from unstructured text, since most NLP tools are trained to handle the general English language. With the aid of ORNL's cyber-domain specific entity extraction library to address this challenge, the TwitterOSINT prototype system performed well.

Overall, the combination of NLP, machine learning and Elastic Stack provided a very powerful, extensible, and easy to use platform. The TwitterOSINT framework shows promise as a tool for helping cybersecurity analysts monitor the latest cyber threats being uncovered around the globe, empowering them to be proactive in protecting their organizations' assets from the threats, with a high degree of automation and accuracy.

In the future, we plan to incorporate additional open sources into the OSINT prototype system, such as security blogs, news websites, and other online venues where users discuss and debate cybersecurity threats, attacks and vulnerabilities. We also plan to make use of additional capabilities provided by the Elastic Stack, such as its built-in machine learning and alerting features, to facilitate further automation of data analytic and cybersecurity operation tasks. In addition, we like to trial our TwitterOSINT prototype system in actual cyber operation environments so we can get input and feedback from people who work on cyber operations to help guide our future planning, design, implementation, and experimentation efforts.

References

[1] "tweeter Developer," [Online]. Available: https://developer.twitter.com/en/docs. [Accessed 16 Jun 2018].

[2] S. Sola, "Playing with Twitter Streaming API," Medium, 23 Nov 2016. [Online]. Available: https://medium.com/@ssola/playing-with-twitter-streaming-api-b1f8912e50b0. [Accessed 16 Jun 2018].

[3] S. Bansal, "Ultimate Guide to Understand & Implement Natural Language Processing," Analytics Vidhya, 12 Jan 2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/01/ultimate-guide-to-understand-implement-natural-language-processing-codes-in-python/. [Accessed 15 Jun 2018].

[4] P. M. Nadkarni, L. Ohno-Machado and W. W. Chapman, "Natural language processing: an introduction," *Journal of the American Medical Informatics Association,* vol. 18, no. 5, pp. 544-551, 1 Sep 2011.

[5] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*, Helsinki, Finland, 2008.

[6] A. Arellano, E. Zontek-Carney and M. A. Austin, "Frameworks for Natural Language Processing of Textual Requirements," *International Journal on Advances in Systems and Measurements,* vol. 8, no. 3 & 4, pp. 230-240, 2015.

[7] "Stanford CoreNLP – Natural language software," [Online]. Available: https://stanfordnlp.github.io/CoreNLP/. [Accessed 15 Jun 2018].

[8] R. A. Bridges, C. L. Jones, M. D. Iannacone and J. R. Goodall, "Automatic labeling for entity extraction in cyber security," in *Third Annual ASE Cyber Security Conference*, Stanford University, 2014.

[9] C. L. Jones, R. A. Bridges, K. M. Huffer and J. R. Goodall, "Towards a Relation Extraction Framework for Cyber-Security Concepts," in *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, Oak Ridge, TN, 2015.

[10] "stucco / entity-extractor," [Online]. Available: https://github.com/stucco/entity-extractor. [Accessed 16 Jun 2018].

[11] "OpenNLP," [Online]. Available: https://opennlp.apache.org/. [Accessed 16 Jun 2018].

[12] "Elastic," [Online]. Available: https://www.elastic.co/. [Accessed 16 Jun 2018].

[13] "Significant Terms Aggregation," elastic, [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-significantterms-aggregation.html. [Accessed 17 Jun 2018].

[14] "Visualize," elastic, [Online]. Available: https://www.elastic.co/guide/en/kibana/current/visualize.html. [Accessed 17 Jun 2018].

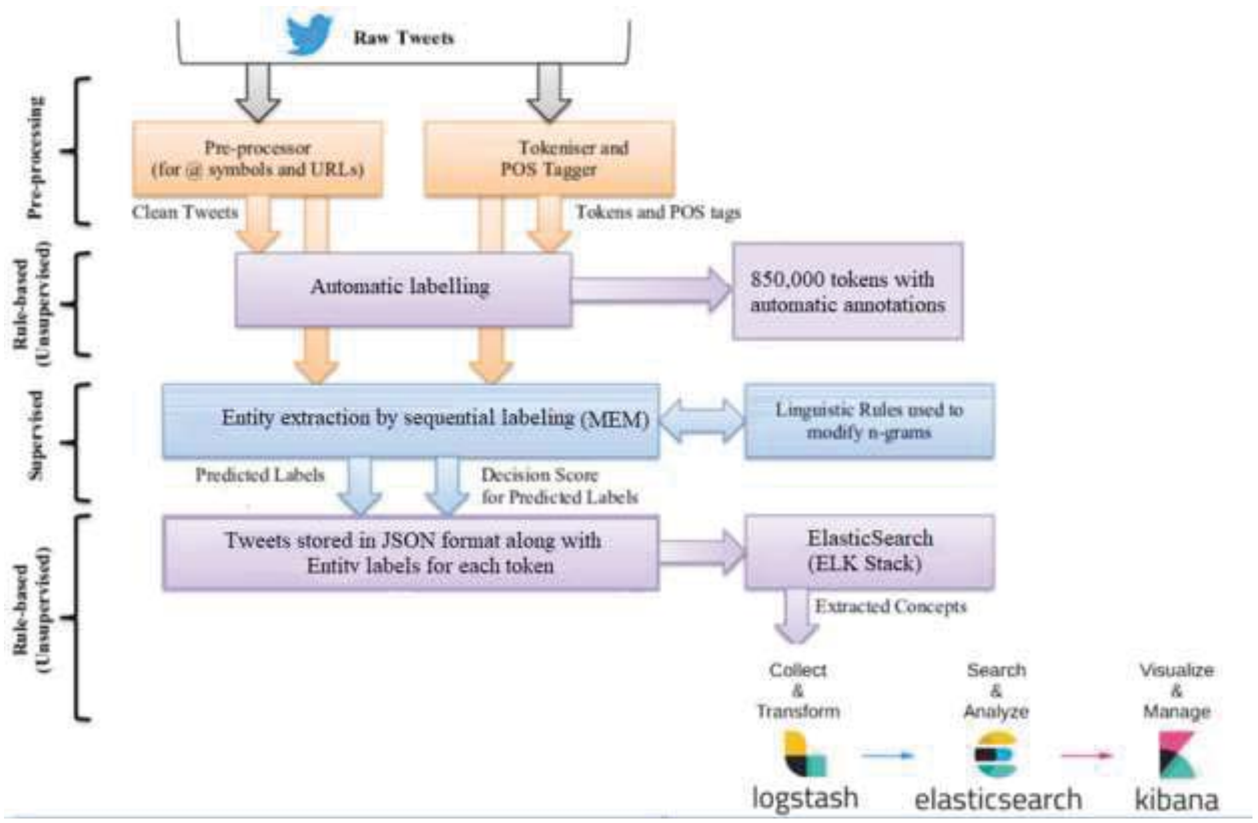

Raw Tweets
Pre-processing
Pre-processor (for @ symbols and URLs)
Tokeniser and POS Tagger
Clean Tweets
Tokens and POS tags
Rule-based (Unsupervised)
Automatic labelling
850,000 tokens with automatic annotations
Supervised
Entity extraction by sequential labeling (MEM)
Linguistic Rules used to modify n-grams
Predicted Labels
Decision Score for Predicted Labels
Rule-based (Unsupervised)
Tweets stored in JSON format along with Entity labels for each token
ElasticSearch (ELK Stack)
Extracted Concepts
Collect & Transform
Search & Analyze
Visualize & Manage
logstash
elasticsearch
kibana

Fig. 10. TwitterOSINT architecture.