Towards a Content-based Defense against Text DDoS in 9-1-1 Emergency Systems

Bal Krishna Bal, Weidong Larry Shi, Shou-Hsuan Stephen Huang, Omprakash Gnawali
Department of Computer Science
University of Houston
{bbal, wshi3}@uh.edu, {gnawali, shuang}@cs.uh.edu

***Abstract*—Text messaging is getting increasingly popular among all generations because it is built into the cellphone carried by people all the time. People, including those with speech and hearing disabilities, are starting to use text messages in place of voice 9-1-1 calls to call for help during emergencies and many 9-1-1 centers are starting to support text messaging. Since text messages take less bandwidth, it is more likely to be available in a major disaster than voice calls. Text messages are also useful in emergency situation such as a child hiding in a closet during a home invasion. On the other hand, text message system is also subject to abuse by hackers. In a recent attack, a teenager was able to send voice calls from a large number of smartphones to 9-1-1 call centers. That same Distributed Denial of Services (DDoS) attack can happen to text messages. While it does not fall under the policy of the call centers to filter out the incoming messages, it would be useful to do some preliminary analyses of the text messages and based on the result of those analyses determine the priority order for processing thereby helping human operators to efficiently manage the messages. In this work, we design and implement several new Natural Language Processing (NLP) techniques to analyze the contents of the incoming text messages to an emergency call center to provide insights about potential spam or DDoS attacks to 9-1-1 centers. Our preliminary results show that the task of automatically analyzing the text to determine if a text is part of an attack can be done with reasonable accuracy.**

I. Introduction

Text Messaging, which has already become a prominent means of communication among people of different walks of life has the potential of serving as a life saver in emergency situations like disasters, fire breakouts, medical emergencies or situations where the help of law enforcement or emergency response bodies is required. It becomes particularly useful when the voice network is too busy or when people are not in a position to make a call either because they are physically unable to do so or the prevailing circumstances are too risky to make a call. According to a recent report [1], there are more than 37 million people in the United States having speech and hearing disabilities. 83% of the American adults (age 18-24) own cellphones and about 73% send and receive text messages [2]. Among these 73%, 31% prefer text to talking over the phone. The survey further revealed that cell owners of the aforementioned age group send and receive on an average of 41.5 messages daily. These facts alone justify the need and application of text messaging systems at crucial times of life and death.

Unfortunately, it is quite likely that text messages could be abused or even attacked by attackers with malicious intentions of disrupting the 9-1-1 emergency call centers by sending a large volume of text messages. In a typical attack scenario or at times of natural disaster, the volume of text messages could be significantly higher than the number of available human operators to handle them, thus resulting in the denial of service to genuine help seekers. While it does not fall under the policy of the emergency call centers to filter out the incoming messages, it certainly would be useful to do some preliminary analyses on the contents of the messages and based on the results of those analyses, set some priorities and provide additional contextual information about the messages so that the human operators could efficiently manage the messages.

In this work, we propose a general framework of a Text Analysis Engine, whose main objective is to do some pre-analysis of the incoming messages, such as determining whether it contains an address, whether the same or similar messages have been received earlier, whether the message(s)

is/are garbage or nonsensical, whether the texts have typos, classification of the texts into Emergency or Non-Emergency categories, and finally, the categorization of the text into one of the three emergency types − Medical, Police or Fire. We evaluate the system on a dataset of real text messages sent to 9-1-1 centers and find that the proposed system provides reasonable accuracy in automatically flagging malicious text activities.

II. Related Work

To the best of our knowledge, there is no prior work that performs early detection and mitigation of cybersecurity threat in the context of 9-1-1 emergency texts. However, there are prior work dealing with text analytics and the application of Natural Language Processing and Machine Learning tools and techniques for cybersecurity in general. Researchers have developed an initial corpus of text message data and applied simple bi-gram feature extraction methodology to classify each text message as drug-related or neutral [3]. There is also prior work on framework for information extraction to extract cybersecurity-relevant entities, terms and concepts from the National Vulnerability Database (NVD) and from unstructured text [4]. The extracted concepts are then mapped and linked to related resources on the web using an OWL ontology language and represented as RDF linked available data. The authors claim that their efforts can be useful in driving applications such as situation aware intrusion detection system. Researchers have also developed a precise method to automatically label text from several data sources by leveraging related, domain-specific, structured data and provide public access to a corpus annotated with cybersecurity entities [5]. Other approaches include following semi-supervised Natural Language Processing techniques to implement a bootstrapping algorithm to extract security entities and their relationships from text [6]. There is also existing work that proposes an early intrusion detection system based on structural modeling of cyber attack behavior [7]. The authors report F1 scores of 0.9 for early detection of network attacks in the KDD99 dataset within windows of certain sizes.

III. System Architecture of the Text Analysis Engine

The Text Analysis Engine analyzes the incoming text messages to a 9−1−1 system and provides output with attributes extracted from the text and probability of the text being a part of an attack. The Text Analysis engine comprises the following sub-modules:

- Random Words Checking Module
- Emergency Non-Emergency Classification Module
- Address Checking Module
- Typo Checking Module
- Duplicate Checking Module
- Near-Duplicate Checking Module
- Address Checking Module

Architecture: The input to the engine is text messages arriving at the 9−1−1 center. The engine is integrated to the 9−1−1 as a passive listener so as not to disrupt the usual call and text handling workflow in place at 9−1−1 center. The goal is to provide additional context and decision support information to the 9−1−1 operator along with the text. Thus the output of the analysis done by the engine is attribute, context, and risk analysis score of the incoming text provided to the 9−1−1 operator dashboard or the supervisor dashboard. The analysis results are also stored in a local database for cross-referencing and forensics in the future. Each module operates independently and produces its own output, which is provided to the decision aggregation module. Different 9−1−1 centers may wish to utilize different modules, hence our modular design can accommodate different legal or technical needs of different centers. Figure 1 shows the overall architecture of the engine.

IV. Component Design

In this section, we describe the components of the text analysis engine.

A. Random Words Checking Module

Text-based attacks to the 9−1−1 centers may contain many random characters or words that do not make sense to a human. Such random words may occur in the attack text due to unsophisticated attackers not being able to create coherent words

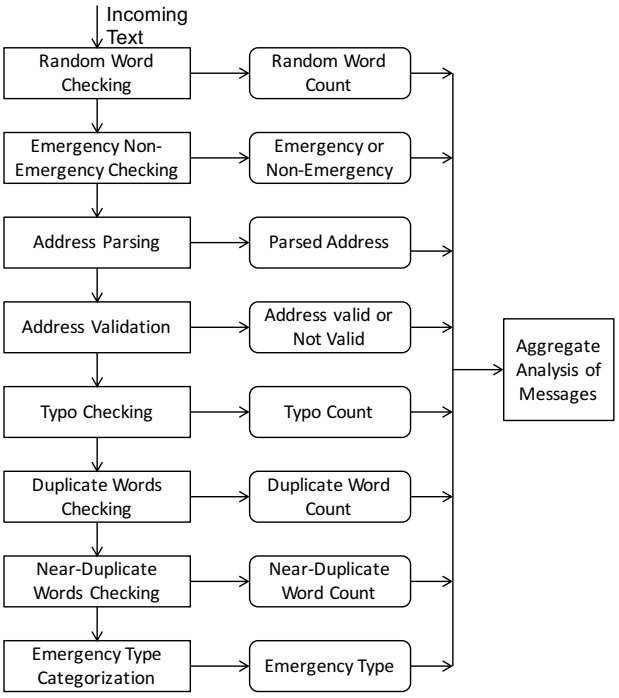

Fig. 1. Work Flow of the Text Analysis Engine.

or sentences automatically, adversarial text with an intention to thwart human cognition or algorithmic processing, or encrypted data for attacks. This module tries to detect such random words in text messages as a potential indicator of an attack. However, such random words or characters could also be due to typos on the text sent by people asking for help through the $9-1-1$ system. Thus, the existence of random words or characters alone is not sufficient to flag a text as an attack. The Random Words Checking Module, which is based on the PyEnchant package [8], determines if the text message contains some combination of random characters, and thus is non-sense or garbage. This module consults a dictionary of English words and uses the criteria that 20% of the words from the message must exist in the dictionary file, and 85% of all the characters in the message must be letters and spaces, rather than punctuation or numbers in order for the text to be considered as sensible.

B. Emergency Non-Emergency Classification Module

One of the first tasks a $9-1-1$ operator performs when handling an incoming text message is determine if it is a true emergency. Automatically performing this task or at least providing context to the operator to make this task easier to perform more accurately would improve the performance of the call centers. The Emergency Non-Emergency Classification Module analyzes the text messages and classifies them into either the Emergency or the Non-Emergency category. We used the scikit-learn platform [9], and the supervised learning classifiers Multinomial Naive Bayes and Support Vector Machine to implement this module. We used five-fold cross-validation in which four sub-parts of the data were used for training and the remaining part was used for testing. We have used the Term Frequency Inverse Document Frequency (TF-IDF) as the feature to train the classifier. The TF-IDF considers the frequency of the word in consideration in the text message as well as occurrence of the given term across a collection of multiple messages in this case.

C. Address Checking Module

When citizens ask for help, they often need to provide their physical address to which the center can dispatch police, fire, or medical personnel. Thus, the presence of address and the validity of address can provide additional context to the $9-1-1$ operator and also indicate the genuineness of the emergency and if it is a duplicate reporting of an emergency. The Address Checking Module has two components:

1) Address Parsing: We first determine if there is a physical address in the text. Address may be present in many formats and abbreviations, especially when the citizens are in an emergency. Thus we need a sophisticated address parsing so it is robust against those variations. We use the usaddress [10] python package to parse and extract the address from the text.

2) Address Validation: Once we know there is an address in the text, we need to determine if it is a *local* and *valid* address. The reason is a $9-1-1$ center can typically dispatch emergency to only local and valid addresses under its jurisdiction. If an address is detected, it is extracted from the text, and then the module searches for the address in the local database. If the address is found in the local database, then the validity of the address is

confirmed. Alternatively, if the address is not found in the local database, then a query about the address is sent to geopy [11] and if geopy determines that the address is valid, the located address is then saved in the local data base. Currently, if the address is not found even by geopy, we consider the address to be invalid. At the end of this process, the address is available in a structured format to the operator or for other IT systems to directly utilize. If the address is invalid or not local, the system or the operator can prompt the user for additional information about the location.

D. Typo Checking Module

The occurrence of typos in text could be an indication that the text was sent by a human, especially when it is an emergency. Although the attackers can design bots to also send text with typos, generally we assume that at least in the near future, the bots tend to try their best to send text with clean grammar as they try to imitate humans in a natural conversation in text. This module checks if there are any typos in the text message, and their respective counts. The module consults an English wordlist [12] and looks for words in the text message that do not match the words in the wordlist. Excluding punctuation symbols and whitespace characters, all words that do not match the English words are returned as typos. In order to handle contracted expressions like *don't*, we have developed a list of patterns, which when detected in the message, will be expanded first before conducting the matching search of the wordlist.

E. Duplicate Checking Module

A large-scale denial of service attack on a $9-1-1$ center may include duplicate, often hundreds or thousands of times, reporting of a single event. This module determines whether an incoming text message is identical to the one that already exists in the database of the system. For this purpose, each text message is treated as a record in the database, and the contents of the record are hashed. To detect if there is an identical text message, the hash value of the text message is matched with the previously computed and stored hash values of the existing text messages in the database.

F. Near-Duplicate Checking Module

Multiple reporting, malicious or otherwise, of the same event may not be an exact duplicate. This module determines whether the incoming text message is similar to the previously received text messages in the database. We compute Jaccard index or similarity coefficient for the messages in consideration. This measurement is defined as the size of an intersection divided by the size of the union of the sample records:

$$J(A,B) = \frac{|A \cap B|}{|A| \cup |B|} = \frac{|A| \cap |B|}{|A| + |B| - |A \cap B|} \quad (1)$$

Where, notation-wise, given set A, the cardinality of A, denoted by$|A|$, counts the number of elements in A. The intersection of the two sets A and B, denoted by $A \cap B$, contains all of the items in A and B. The union between the two sets A and B, denoted by $A \cup B$, contains all items which are in either set. Items refer to words in the given context. We employ the following steps to apply the above equation to find near-duplicate content:

1) Remove stop words from both messages
2) Tag the remaining words in the text with their respective parts of speech
3) Extract content words from the POS tagged words, namely the adjectives, verbs, nouns and adverbs, and extract their lemma or root forms
4) For each of the lemma forms in the text message pairs, compute the Jaccard coefficient given by the equation above

We consider text messages with the Jaccard coefficient value greater than 70% as near-duplicate messages.

G. Emergency Categorization Module

The $9-1-1$ operator needs to determine the type of emergency corresponding to the incoming text message. The Emergency Categorization Module analyzes the text messages and categorizes them into one of the emergency types: Medical, Police or Fire. We use two approaches for the implementation of this module: the WordNet-based approach and the machine learning approach. In the WordNet-based approach, we use the text relatedness measure called vector-pairs in WordNet, available via the

WordNet::Similarity package [13]. For the machine learning-based implementation, we use the scikit-learn platform and the supervised learning classifiers: Multinomial Naive Bayes and Support Vector Machine. We use a five-fold cross-validation for testing the performance of the classifiers. Similarly, we use the TF-IDF as the training feature for the classifiers.

V. Dataset for the Experiments

The 9-1-1 emergency text is not typically released to the public by the $9-1-1$ centers. Hence, for experimental and evaluation purposes, we built a dataset based on real 9-1-1 text messages and 9-1-1 call transcripts available in the Internet. We gathered a total of about 500 text messages out of which 321 are Emergency messages whereas 179 are Non-Emergency messages. Our dataset has 163 messages belonging to the Police category, 100 messages belong to the Medical category and 58 messages belong to the Fire category. We manually labeled these messages as Emergency and Non-Emergency as well as the three Emergency Types, respectively, Police, Medical and Fire. The labeled messages have been used for training and testing the classifiers.

VI. Performance Results

We evaluated each module of the text analysis engine separately and as an integrated system, focusing on the accuracy as the performance metric. The reported accuracy values are based on our current dataset. The Word-Net based Emergency Categorization module performs with an accuracy of 65.85%. We present the results of the Machine Learning based Emergency categorization module in Table I.

We present the results of the Emergency Non-Emergency Classification Module in Table II.

The results in Table 2 indicates that except for recall, the Support Vector Machine clearly outperforms the Multinomial Nave Bayes classifier in terms of performance. However, we should note that in $9-1-1$ emergency situations, it is permissible to have a few false positives (non-emergency being classified as emergency) but we cannot afford to have false negatives (emergency being considered as non-emergency). In this respect, it is desirable to pick Multinomial Nave Bayes classifier instead of the seemingly better performing Support Vector Machine. Alternatively, we can further attempt to enhance the performance of the Support Vector Machine by fine tuning the penalty parameter that helps in fitting the boundaries between classes more smoothly and appropriately. With a larger dataset, we can experiment on choosing appropriate kernel function for the Support Vector Machine. This will result in a better model fit and hence further contribute to a higher precision and recall. We present the summary of the performance of each module of the text analysis engine in Table III which shows that the different modules of the Text Analysis Engine achieve reasonable accuracy.

TABLE I
Performance of the Machine Learning based Categorization module. "P" is Police, "M" is Medical,and "F" is Fire.

Feature	Classifier	Precision	Recall	F1-Score
TF−IDF	Multinom. NB	P: 0.67	P: 0.99	P: 0.79
		M: 0.85	M: 0.67	M: 0.71
		F: 0.13	F: 0.08	F: 0.13
	SVM	P: 0.79	P: 0.95	P: 0.86
		M: 0.87	M: 0.73	M: 0.78
		F: 0.9	F: 0.74	F: 0.74

TABLE II
Performance of the Machine Learning based Categorization module

Feature	Classifier	Precision	Recall	F1-Score	Accuracy
TF−IDF	Multinom.NB	0.77	1	0.86	0.79
	SVM	0.90	0.98	0.94	0.92

VII. Conclusion and Future Work

We presented a Text Analysis Engine which consists of a set of modules working together to analyze the incoming $9-1-1$ texts. Our preliminary results show that the text message analysis for prioritization and classification can be done with reasonable accuracy. The analyses returned by the engine can be a first step towards early detection and mitigation of cyber-attacks via $9-1-1$ texts. To further improve the accuracy, we plan to include more advanced

TABLE III
ACCURACY ACHIEVED BY EACH MODULE OF THE TEXT ANALYSIS ENGINE

No.	Module Name	Accuracy
1	Random Words Checking Module	93%
2	Emergency Non-Emergency Classification Module	92%
3	Address Checking Module (Parsing)	94%
4	Address Checking Module (Validation - Local Database)	100%
5	Address Checking Module (Validation - Global DB)	100%
6	Typo Checking Module	91%
7	Duplicate Checking Module	100%
8	Near−Duplicate Checking Module	91%
9	Emergency Type Categorization Module	84%
10	Integrated Text Analysis Engine	90%

features or attributes associated with the texts, for example, timestamps of submission of text(s), the phone number of the incoming texts, submission pattern of the texts (whether it is regular or not, whether the texts are sent on week days/weekends, normal or late hours) etc. The inclusion of these features along with the current textual features will likely improve the accuracy of the system. Furthermore, we intend to test our engine with improved training over larger datasets in the future. With machine learning algorithms trained on those larger datasets, we expect the accuracy to increase for the categorization and the classification tasks.

ACKNOWLEDGMENT

This Project is the result of funding provided by the Science and Technology Directorate of the United States Department of Homeland Security under contract number D15PC00185. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied of the Department of Homeland Security or the U.S. Government.

REFERENCES

[1] "National organization on disability," [Online]. Available: https://www.nod.org/. [Accessed: October 29, 2017].

[2] "Pew research center, internet, science & tech." [Online]. Available: http://pewinternet.org/.[Accessed: October 29, 2017].

[3] D. R. O'Day and R. A. Calix, "Text message corpus: Applying natural language processing to mobile device forensics," in *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. "IEEE", July 2013.

[4] A. Joshi, R. Lal, T. Finin, and A. Joshi, "Extracting cybersecurity related linked data from text," in *Proceedings of the 7th IEEE International Conference on Semantic Computing*. IEEE Computer Society Press, September 2013.

[5] R. A. Bridges, C. L. Jones, M. D. Iannacone, and J. R. Goodall, "Automatic labeling for entity extraction in cyber security," *CoRR*, vol. abs/1308.4941, 2013. [Online]. Available: http://arxiv.org/abs/1308.4941

[6] C. L. Jones, R. A. Bridges, K. M. T. Huffer, and J. R. Goodall, "Towards a relation extraction framework for cyber-security concepts," *CoRR*, vol. abs/1504.04317, 2015. [Online]. Available: http://arxiv.org/abs/1504.04317

[7] X. Yan and J. Y. Zhang, "A early detection of cyber security threats using structured behavior modeling," 2013.

[8] "Pyenchant - a spellchecking library for python," [Online]. Available: http://pythonhosted.org/pyenchant/.[Accessed: October 29, 2017].

[9] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," *CoRR*, vol. abs/1309.0238, 2013. [Online]. Available: http://arxiv.org/abs/1309.0238

[10] "usaddress 0.5.7," [Online]. Available: https://pypi.python.org/pypi/usaddress/.[Accessed: October 29, 2017].

[11] "geopy 1.11.0," [Online]. Available: https://pypi.python.org/pypi/geopy/.[Accessed: October 29, 2017].

[12] "Infochimps, compiler," [Online]. Available: https://github.com/dwyl/english-words/.[Accessed: October 29, 2017].

[13] T. Pedersen, S. Patwardhan, and J. Michelizzi, "Wordnet::similarity: Measuring the relatedness of concepts," in *Demonstration Papers at HLT-NAACL 2004*, ser. HLT-NAACL–Demonstrations '04. Stroudsburg, PA, USA: Association for Computational Linguistics, 2004, pp. 38–41. [Online]. Available: http://dl.acm.org/citation.cfm?id=1614025.1614037