Scam Detection Assistant: Automated Protection from Scammers

Myeongsoo Kim
Kookmin University
myron89@kookmin.ac.kr

Changheon Song
Seoul National University
chsong@idb.snu.ac.kr

Hyeji Kim
Kookmin University
95kimhyeji@gmail.com

Deahyun Park
Kookmin University
eodhs0@kookmin.ac.kr

Yeeji Kwon
Seoul Women's University
yeeji77@naver.com

Eun Namkung
Kookmin University
nk8355@gmail.com

Ian G. Harris
University of California Irvine
harris@ics.uci.edu

Marcel Carlsson
Lootcore
mc@lootcore.com

***Abstract*—Scams, also known as social engineering attacks, are an extremely common and dangerous threat today. Scams typically result in financial loss by convincing a victim to perform an ill-advised action such as sending money, or convincing him to provide private information. In this paper we present an approach to detect scams, focusing on scams which are conveyed in-person, over the phone, or via text/chat message. We present a tool called Scam Detection Assistant (SDA) which analyzes attack content to detect inappropriate statements which are indicative of social engineering attacks. A great deal of previous research in the detection of scams focuses on the detection of email scams, phishing emails. Previous work relies heavily on the analysis of various metadata specific to the email attack vector, including header information and URL links. SDA is novel compared to previous work because it focuses on the natural language contained in the attack, performing semantic analysis of the content to detect malicious intent. Focusing on content analysis makes our approach applicable to detect scams using non-email attack vectors, including texting applications, chat applications, and phone/in-person attacks which have been converted to text using a speech-to-text application.**

I. Introduction

A critical threat to the security of individuals and organizations is the increasing rate of scams which are being perpetrated each year. In the year 2016, a total of 9.5 billion dollars was lost in the USA due to phone scams alone [16]. The more formal term used for scamming is *social engineering*, the psychological manipulation of people in order to gain access to a system for which the attacker is not authorized [10], [18]. Numerous experimental studies over the years have demonstrated the susceptibility of people to social engineering attacks [8], [13], [24], [19], [1]. The use of modern communication technologies, including cellular phones and the internet, have greatly increased the reach of an attacker, and the effectiveness of the attack.

Scams involve communication between the attacker and the victim in order to either elicit some information, or persuade the victim to perform a critical action. Information gathered might include explicitly secure information such as a credit card number, or seemingly innocuous information which can support a larger attack, such as the name of a coworker. An attacker might also convince the victim to perform tasks which would support an attack, such as going to a website. The effectiveness of social engineering has encouraged attackers to use it more frequently, relying on social engineering as a component of larger attacks. Scams are becoming larger and more ambitious each year, such as the IRS phone scam [11] which convince people to wire money in order to pay a debt to the Internal Revenue Service, the federal tax collection agency in the USA.

Scams are conveyed by many methods including in-person, phone, chat/text, and via email. Most previous work in the detection of scams focuses on email-based "phishing" scams, but non-email scams are potentially more dangerous. The nature of communication via these non-email attack vectors is substantially different from email-based communication, and offers many additional options to the attacker. Email communication is not real-time. When an email is received, a response is not necessarily expected immediately, if at all. Non-email communication involves two-way conversation in which each speaker is expected to respond immediately. From an attacker's point of view, conversations are a useful tool because it pressures the victim to respond without spending time considering the consequences. The feedback received by the attacker during the conversation enables the attacker to get a feel for the mood of the victim and adjust the attack accordingly. This allows non-email social engineering attacks to be more personalized and therefore more effective than phishing emails.

Existing approaches for automatic detection of social engineering attacks focus on the detection of phishing emails. These techniques rely heavily on the analysis of non-content metadata which is found in email, such as contained hyperlinks and SMTP headers. These tech-

niques are not effective for detecting non-email social engineering attacks which are not associated with accessible metadata. When there is no metadata to rely on, non-email social engineering attacks must be detected by analyzing the content of the communication. Content-based approaches do exist which analyze features of the content such as character frequency and word/n-gram frequency. However, these approaches do not attempt semantic analysis to extract the meaning the text, and the intent of the attacker. Without semantic analysis, the use of existing content-based metrics alone would result in low precision and accuracy.

A. Our Contribution

We present an approach, Scam Detection Assistant (SDA), which analyzes the text of a conversation in order to determine if it a scam is being perpetrated. In order to detect a broad range of social engineering attacks using a range of communication vectors, we use semantic analysis to understand essential aspects of the meaning of the communication. Our approach identifies sentences with malicious intent in the communication from the potential attacker.

In order for the attacker to achieve his goal, the attacker must perform one of the following detectable actions.

- **Ask a question** whose answer is private.
- **Issue a command** to perform a forbidden operation.

Our approach identifies all statements which are either questions or commands posed to the victim, and checks the appropriateness of the statement. A question is considered to be inappropriate if it requests private information, and a command is considered to be inappropriate if it involves the performance of a secure operation.

Our approach to the detection of inappropriate questions leverages research in **question answering** systems to determine the privacy of the answers to questions posed by the attacker. We use question answering technology to provide the privacy status of the answer, rather than providing the answer itself, as is the goal of traditional question answering systems. The actual answer of each question is not needed, so our approach can tolerate imprecision inherent to current question answering approaches and still achieve high precision with respect to privacy status.

We evaluate commands by summarizing their meaning as a combination of the main verb and the object(s) of that verb in the sentence. For example, the meaning of the command, "Please reset the router" would be summarized by the **verb-object pair** (reset, router). This verb-object pair will be compared to a blacklist of verb-object pairs which are known to describe forbidden operations. Reducing the meaning of a command to the verb-object pair is beneficial as a method to normalize the description of sentences with different syntaxes but identical meaning. By considering synonyms and lemmatization as well, the set of all forbidden operations can be stored in a compact blacklist.

B. Authentication

Authentication is outside of the scope of this work, so our approach assumes that the party communicating with the victim is unknown to the victim. The type of trust relationship which a person has with other entities will impact what information is considered private and which operations are considered secure. For example, a close friend can ask for some private information while a stranger cannot. Considering the impact of identity on the privacy of data and operations requires an authentication approach. Since authentication is outside of the scope of this work, we determine the privacy of data and operations assuming that access is requested by an unknown and untrusted entity. Although our approach is limited in this way, it could easily be extended to consider a more graduated notion of privacy by using it together with an existing authentication approach.

C. Scam Data

Several parts of our approach require mining of actual scams as part of the training process. Scams are also required to evaluate the effectiveness of our approach. The databases of phishing emails summarized in Table I are publicly available, so those are what we have used to develop and evaluate our work. Since our approach is content-based, it can be applied to non-email social engineering attacks as well, but we were limited to the phishing email databases since non-email scam examples are not publicly available. All online databases were accessed October 11, 2017.

Database	URL	Size
Scamdex	http://www.scamdex.com	56555
Scamwarners	http://www.scamwarners.com	43241
Scamalot	http://scamalot.com	18149
Antifraudintl	http://antifraudintl.com	69209
Total		**187154**

TABLE I
SOCIAL ENGINEERING ATTACK DATA

D. Structure of the System

Figure 1 shows the structure of the SDA system. The input to the system is a block of text uttered by a potential attacker. The text may be extracted from any means of transmission including email, texting application, phone, or in-person. Text extracted from either phone or in-person communication would need to be transcribed using an existing speech-to-text engine. The output of the

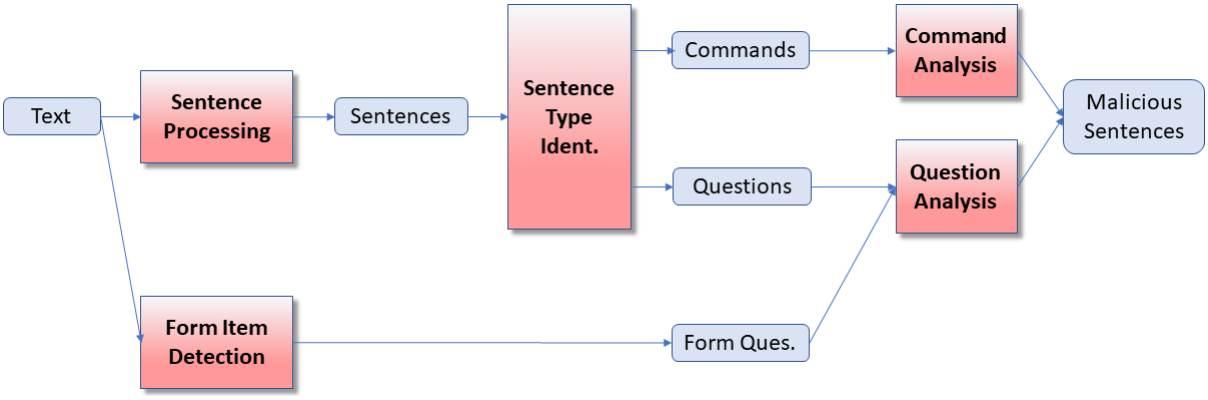

Fig. 1. Scam Detection Assistant (SDA) structure

system is a set of *malicious sentences* which are determined to either ask questions whose answers are private, or issue commands to perform forbidden operations. The original text is identified as a social engineering attack if the set of malicious sentences is non-empty.

The system as shown in Figure 1 is composed of five main components. The **Sentence Processing** step separates the text into individual sentences and parses each sentence in order to gather information about sentence components which will be used for analysis by other system components. Separating individual sentences is straightforward if proper punctuation is used in the original text, but we assume that proper punctuation cannot be assumed in the original text. This is particularly true if the text is taken from an acoustic source such as a phone or in-person conversation.

The **Sentence Type Identification** step determines whether each sentence is a question or a command, and places the sentence in the appropriate set for further analysis. Sentences which are neither questions nor commands are ignored. The **Question Analysis** step determines whether or not a question has a private answer, and **Command Analysis** determines whether or not a command refers to a forbidden operation.

Text may include utterances which do not express questions in isolation, but are understood to be questions based on the context within the text. Examples of such utterances are the items in a questionnaire where each individual item is not a question, but the context tells the listener that the item should be treated as a question and an appropriate answer is expected. For example, an attacker might make the following statements, “Please give me the following information. Name. Address. Phone”. This statement should be treated as three questions requesting the “Name”, “Address”, and “Phone”. We refer to these implied questions as *form questions* and the **Form Item Detection** step identifies utterances which represent implicit questions because they are items in a form. Each form item is used to generate an equivalent question which can be evaluated in the Question Answering step.

E. Organization of the Paper

The central components of this work are the **Question Analysis** and **Command Analysis** steps, so our description will focus on those steps and the **Sentence Type Identification** step which determines the type of analysis required for each sentence. However, our results are generated using all steps shown in Figure 1.

The remainder of this paper is organized as follows. Section II describes sentence type identification, Section III describes question analysis, and Section IV describes command analysis. Section V shows our experimental results. A discussion of the results is presented in Section VI Previous related research is summarized in Section VII and conclusions are discussed in Section VIII.

II. Sentence Type Identification

Questions and commands are the types of sentences which are of interest because they might refer to private data or private operations. The identification of questions and commands analyzes the words in the sentence and the syntactic and typed dependency parse trees of the sentence which were created in the Sentence Processing step.

Question detection is straightforward using the syntactic parse tree of the sentence. There are two types of questions and each type can be recognized by the presence of specific tags in their syntactic parse tree. *Closed questions* are those whose answers are ”yes” or ”no”. Closed questions are identified by the presence of the SQ tag in their parse tree. *Open questions* are those which ask open ended questions, typically containing a wh-word such as ”who” or ”where”. Open questions are

identified by the presence of the SBARQ tag in their parse tree.

We present four different types of commands, each of which is identified in a different way.

- **Direct imperatives** - Commands are made using imperative sentences which generally start with the base form of the verb, as in "Open the door" or "Come in". The imperative form is a second-person form, so the subject is the person being spoken to. Direct imperative sentences often do not include an explicit subject because the subject is assumed to be the listener.
  Direct imperatives are identified if they match a regular expression of the form "¡verb¿ ...", indicating that the sentence starts with a verb.
- **Polite prefixes** - Simple imperative sentences can sound rude, so an attacker may desire to soften the request by prefixing the command with a polite greeting, such as, "Please go home".
  Polite prefix imperatives are identified if they match a regular expression of the form "Please ¡verb¿ ...", indicating that the sentence starts with the word "Please" followed by a verb.
- **Suggestion** - Rather than directly commanding a victim to perform an action, the command may be phrased as a suggestion which the victim should follow, such as, "You could open the door" or "You should go home".
  Suggestion imperatives are identified if they match a regular expression of the form "You ¡modal verb¿ ...", indicating that the sentence starts with the word "You" followed by a modal verb such as "should", "could", or "must".
- **Expression of desire** - Another way to soften a command is to prefix it with an expression of desire such as "I want you to come in" or "I urge you to come in".
  The detection of commands which express desire is performed based on two observations, 1) the sentence must include a *desire* verb such as "urge" or "encourage", and 2) the pronoun "you" is the direct object of the desire verb. The presence of a desire verb is determined by comparing each word in the sentence to a list of desire verbs.

III. Question Analysis

Our approach modifies an existing approach for question answering to determine whether or not the answer to a factoid question is private. We will first describe the existing question answering approach and we will then present our modifications to the approach to make it suitable for use in detecting social engineering attacks.

A. PARALEX Question Answering System

The existing PARALEX question answering system which we modified is fully described in previous work [5] and is outlined in Figure 2. Answers are found in an SQL database and the SQLite database engine [15] is used to search the database for answers. The database is a collection of triples of the form $r(e_1, e_2)$, where r is a relation and e_1, e_2 are entities. For example, the fact that the population of New York City is 8.5 million people might be stored as the triple $population(new - york, 8.5 * 10^6)$. A set of formal queries are generated from the original natural language question and the queries are ranked according to the likelihood of being a semantic match to the original query. The highest ranked query is used to search the database and the matching result is returned as the answer. Each query is in one of two forms: either $r(?, e)$ or $r(e, ?)$. Searching the database using a query returns the entity e which satisfies the given relationship.

The challenge is to generate a formal query which is semantically equivalent to the natural language question. Each formal query is composed of database concepts, specifically a relation r and an entity e. The approach uses a **lexicon** to associate natural language patterns with database concepts. Each lexicon entry has the form (p, d), where p is a string pattern found in a natural language question, and d is a database concept. There are three types of entries in the lexicon: *entity entries* pair strings with database entities ("NYC" and $new - york$), *relation entries* pair strings with database relations ("big" and $population$), and *question pattern entries* pair string patterns with query templates ("how r is e" and $r(?, e)$).

A *derivation* is a mapping from a natural language question to a formal query. A derivation is generated by matching lexicon entry patterns to strings in the natural language question. The database concepts of the matching lexicon entries are combined to make the query. For example, assume that we have a lexicon containing the following two entries: ("How big is e", $population(?, e)$) and ("NYC", $new - york$). Since the question "How big is NYC?" matches both lexicon entries, a derivation for this question would combine the database concepts of both entries to create the query $population(?, new - york)$. It is important to observe that there are often many possible derivations for a question with a given lexicon. This happens due to noise in the lexicon which may occur as a result of the learning process used to generate the lexicon [5]. A lexicon may contain entries which interpret the same natural language string pattern in different ways. The question How big is NYC? could be interpreted as asking about population or asking about land area. To describe these interpretations, the lexicon could contain both entries

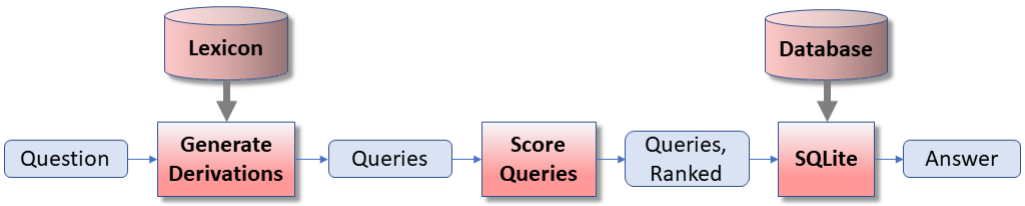

Fig. 2. PARALEX question answering approach

("How big is e", $population(?, e)$) and ("How big is e", $area(?, e)$), leading to two possible derivations. The **Generate Derivations** step shown in Figure 2 uses the lexicon to generate all possible derivations based on the entries in the lexicon. The queries produced by the derivations are scored in the **Score Queries** step and the top ranked query is used to search the database.

B. Modifications to Question Answering

We have modified the PARALEX question answering approach to determine whether or not the answer to a factoid question is private or not. We have made the following three modifications.

1) Creating a Private Database: We replace the database of facts with a database containing only facts which are considered to be private. This guarantees that if an answer to a query is found in the database, it must be a private answer. Since the number of private facts is very small relative to the set of answers to all possible factoid questions, the database used to determine privacy is much smaller than the database used in the original PARALEX approach which contained over 15 million facts. This system requires the creation of a database of private information. Such a database could be created manually, allowing a user to match the database contents to his/her enterprise. For example, if the system is being used to protect a medical office then the database would contain private medical information. Since we are experimenting with a set of phishing emails targeted at generic users, we have created our database based on samples of those emails.

Our procedure for creating the private database was to manually identify private questions and from 20,000 randomly selected phishing emails, and private form questions from 2,000 phishing emails within the larger set of 20,000. We manually determined that a question was private if we could not find the answer using Google. For each private question, we presented the question to the PARALEX tool and observed the formal queries that it created for the question using relations and entities already defined in its lexicon. We examined the top four queries generated by PARALEX and inserted answers to those queries into the private database, using the relations and entities contained in the queries. This process guarantees that if this question or a similar question is posed to the question answering tool, and answer will be found in the database and the question will be identified as being private.

2) k-best Queries: : Rather than using only the top ranked query, we search the database using the k-best queries. If matches are found for any of the k-best queries, the question is considered to be private. The chance that the correct answer will be found by one of the top k queries is considerably higher than the top query alone. The potential problem with this approach is that a question whose answer is not private may be incorrectly classified because one of the queries generated from it does have a private answer. However this is unlikely because private data is such a small subset of the set of all question answers.

3) Lexicon Modification: : The original lexicon developed as part of the PARALEX system provides a mapping from natural language strings to concepts in the original database. Since we are changing the database, the lexicon must be modified to map to the concepts in the new database. The original lexicon is quite expansive, having been generated from a corpus of over 15 million question-answer pairs, but occasionally a new entity must be added to the database, and appropriate entries must be added to the lexicon. During the process of creating the new database, we needed to define only two new entities which were not part of the original lexicon. We also modified only 341 lexicon entries out of a total of over 6 million entries.

IV. Command Analysis

Command analysis is performed by extracting the verb and the direct object of the command so that the verb-object pair can be used as a summary of the intent of the command. A command is considered to refer to a forbidden operation if its verb-object pair is found in a **verb-object blacklist** which describes forbidden operations. Extraction of the verb and direct object is performed by examining the typed dependency parse of the sentence which is generated during sentence processing. The *dobj* dependency relation indicates the

direct object for phrases written in the active voice, and the *nsubjpass* dependency relation indicates the direct object for phrases written in the passive voice. Both the *dobj* and *nsubjpass* dependencies relate the verb to the direct object, so the verb-object pair is determined using one of these two relations.

A. Generating the Verb-Object Blacklist

Command analysis depends on the existence of a blacklist of verb-object pairs which is correct and complete. The blacklist could be created manually, allowing a user to match blacklist contents to his/her enterprise. For example, if the system is being used to protect a bank then the blacklist would contain the pair $(give, combination)$. Since we are experimenting with a set of phishing emails targeted at generic users, we have created our blacklist based on samples of those emails.

Our goal is to find the verb-object pairs which are most closely associated with phishing emails, but are also most weakly associated with non-phishing emails. For this purpose, we compute the *term frequency-inverse document frequency* (TF-IDF) statistic [20] for each verb-object pair. TF-IDF is a well accepted statistic used in data mining and information retrieval to determine how important a term is in a corpus. The statistic takes the product of the *term frequency* which is a measure of how often a term appears in a corpus of documents, and the *inverse document frequency* which is a measure of how often the term appears in documents outside of the corpus. A term receives a high TF-IDF score if it is common within the corpus, but rare outside of the corpus.

For our problem, a term is a verb-object pair in a command, and the corpus in question is a random selection of 100,000 phishing emails from the set of phishing emails which we are using. For a set of non-corpus documents, we use 100,000 emails in the Enron email corpus [14] which is assumed not to contain phishing emails. All commands are identified in both the phishing and non-phishing emails, and all verb-object pairs are identified in each command. The TF-IDF statistic is computed for each verb-object pair and all verb-object pairs whose TF-IDF is above a threshold (0.45) is included in the blacklist. This produced a blacklist containing 508 verb-object pairs.

In order to improve the generality of the blacklist, we performed two additional processing steps on the blacklist. First, we considered synonyms of each word in the blacklist so that the use of synonyms in the social engineering attacks will not cause a command to be classified incorrectly. Additionally, we perform *lemmatization* which reduces different forms of the same word to their common stem. Lemmatization allows words to be recognized regardless of the form in which they are used in a sentence. We use the lemmatizer which is part of the Stanford CoreNLP Toolkit [17]. After this processing, the final blacklist contains 1709 verb-object pairs.

V. Experimental Results

We have evaluated SDA by implementing it and using it to identify phishing emails. Our implementation is primarily a Python script, using Java when necessary to interface with a particular tool. Specifically, Java is used to interact with the Stanford Parser and CoreNLP [17] which have a Java API. The phishing datasets which we use are presented in Table I and the non-phishing emails which we use are those contained in the Enron email corpus [14]. Of the total set of 187,048 phishing emails, we subtract the 100,000 which we used for creating the verb-object blacklist and creating the private database. For evaluating SDA, we use the remaining 87,048 phishing emails and an equal number of non-phishing emails taken from the Enron corpus.

	Phishing	Enron
Detected	56616 (TP)	14168 (FP)
Not Detected	30432 (FN)	72880 (TN)

TABLE II
SDA DETECTION RESULTS

Table II contains the detection results using the SDA system. Each cell in the table contains the count of how many emails fell into each category. The two rows indicate the labeling applied by our tool, either Detected or Not Detected. The two columns indicate the true label of the email, either Phishing (social engineering) or Enron (not social engineering). Each cell also contains initials indicating the correctness of the labeling: TP (true positive), TN (true negative), FP (false positive), or (FN) false negative. Both FP and FN indicate that the email was labeled incorrectly by our system. We also present the Precision $(TP/(TP+FP))$ and Recall $(TP/(TP+FN))$ summary statistics, Precision = 0.800 and Recall = 0.650.

VI. Discussion of Results

It is important to understand the reasons for the sub-optimal results shown in Table II in order to determine if the approach is inherently flawed, or improvements can be made in the future to make the approach more effective. We will examine the reasons for both false negative and false positive examples.

A. False Negatives

False negative examples are those emails which are phishing emails but were not detected as phishing emails by our approach. In order to consider the reason for false negatives, we first need to classify different stages of a

social engineering attack. A social engineering attack can be subdivided into at least 3 parts.

1) **Pretext** The act of *pretexting* is the creation of a scenario to persuade the target to either provide the desired information, or perform the desired action. The pretext will typically define a false identity for the attacker which is trusted by the target to some degree.
2) **Elicitation** Elicitation is the process of building a rapport with the target in order to make the target comfortable enough to provide the desired information or perform the desired action. The target needs to trust the attacker and elicitation is the act of building that trust through communication. Common techniques include flattery, expressing a mutual interest, and volunteering private information [9].
3) **Information/Command Goal** The culmination of the social engineering attack is to either request private information or ask the target to perform an inappropriate operation. The goal will vary based on the information desired ("Please confirm your social security number") or the operation desired ("Please click on the link").

Our detection approach can only detect the final stage of the attack, the **Information/Command Goal**. We observe that many of the undetected phishing emails are only the first stage in a potential sequence of emails, and as a result, involve pretexting and elicitation. These emails may tell a story to engage the victim, and then request that the victim respond in order to start a conversation. The following email is an example of this type of phishing email

```
I KNOW THIS MAIL WILL COME AS SURPRISE
TO YOU BUT IN A BRIEF INTRODUCTION . MY
NAME IS MR TERRY ARUMAH FROM GHANA WEST
AFRICA . I AM A MARKETING MANGER OF
TARKWAH COMMUNITY GOLD MINING CON-PAY IN
TARKWAH COMMUNITY HERE IN THE REPUBLIC
OF GHANA, WE HAVE GOLD DUST AND ALSO
GOLD BAR OUR PRODUCT IS GOOD ,FOR YOU
TO BE SURE OF THE TYPE OF GOLD YOU ARE
BUYING YOU WILL BE ALLOWED TO TEST IT
IN ANY PLACE OF YOUR CHOOSE SO IF YOU
ARE INTERESTED PLEASE YOU CAN CALL US
HERE +2335403977 OR REPLY US HERE OKAY.
GOOD BYE TAKE GOOD CARE OF YOUR SELF .
```

Notice that the email requests that the victim responds with the statement, "PLEASE YOU CAN CALL US", but no overt attempt is made to gain information or suggest a forbidden operation. We manually examined 100 of the false negative emails in order to determine the specific reason why these were not detected. We found that the majority of these emails, 79%, were this type of email, only the beginning of a longer sequence of attack emails.

Although our approach does not detect pretexting or elicitation, we claim that our approach is still useful in practice because the Information/Command Goal stage must eventually occur, and our technique would detect the attack at that point.

B. False Positives

False positives are the emails which are not phishing (Enron emails) but are detected as phishing emails by our approach. We manually examined 100 of the false positive emails in order to determine the specific reason why these were falsely detected. We found that the large majority of these emails, 97%, were detected because they were found to contain suspicious commands whose verb-object pairs were found to be in the blacklist. This occurred with the verb-object pairs such as $(call, me)$, and $(pay, < number >)$. These verb-object pairs are part of the blacklist, so they must have had relatively high TF-IDF scores, but these pairs can also be used in an innocuous way.

This indicates to us that we will need to improve the blacklist in the future. We can experiment with changing the TF-IDF threshold, but this may increase the number of false negatives. This may indicate that the pair of the verb and direct object are not sufficient to summarize the meaning of a command. We will explore using additional semantic information to add context, such as indirect objects also contained in the sentence.

VII. Related Work

A commonly used approach to social engineering detection and prevention is to provide training for employees to make them aware of the risks [21], [7]. Training-based approaches rely on the human to detect and prevent attacks manually. The problem with these techniques is that the vulnerability of a person to a social engineering attack depends on the person's emotional state at the time of the attack, regardless of training received. Manual attack prevention demands more discipline than can be expected from most people. For example, one training-based approach expects a person to internally answer a set of security-related questions before providing information to an external agent [2]. It is hard to believe that a person will consistently maintain this procedure, especially when under the influence of an attacker who is expert at manipulating the emotions of a victim.

A. Automatic Detection

Many previous contributions in phishing detection rely on non-content-based information associated with the email, data contained in headers or log file entries. Examples of non-content-based information used include SMTP headers, NIDS logs, LDAP logs, and cc lists. Several approaches use this information to evaluate emails [12], [4], [23]. A notable approach in this category applied their technique to over 370 million emails and detected spearphishing campaigns with a false positive rate of only 0.004% [12]. Several approaches examine the URLs contained inside the message [6], [3]. Existing non-content-based information are effective for detecting email-based social engineering attacks, but would not be useful for non-email attacks, since the same information is not available.

Some content-based approaches use natural language processing (NLP) techniques to extract information from the email content. For example, EmailProfiler [4] uses the number of each part-of-speech as a feature, using a Stanford Parser to perform part-of-speech tagging. Other techniques attempt to infer some aspect of sentence meaning based on the presence of particular words. An example of this type of text analysis is seen in [22] which defines a set of rules which are regular expressions that match each expected category of phishing email.

VIII. Conclusion

We present the SDA approach to automatically detect scams. Our approach relies on semantic analysis of the content, rather than metadata which might be associated with emails. As a result, our approach can be applied to detecting scam dialogs which are composed of pure text. Our results on phishing emails demonstrate that the use of question answering and verb-object semantic information is useful in detecting scams. The applicability of our approach to the detection of any text-based social engineering attacks is also unique.

References

[1] T. Bakhshi, M. Papadaki, and S. Furnell. A practical assessment of social engineering vulnerabilities. In *Human Aspects of Information Security and Assurance (HAISA)*, 2008.

[2] M. Bezuidenhout, F. Mouton, and H. S. Venter. Social engineering attack detection model: Seadm. In *Information Security for South Africa (ISSA)*, 2010.

[3] J. Chen and C. Guo. Online detection and prevention of phishing attacks. In *First International Conference on Communications and Networking in China*, Oct 2006.

[4] S. Duman, K. Kalkan-Cakmakci, M. Egele, W. Robertson, and E. Kirda. Emailprofiler: Spearphishing filtering with header and stylometric features of emails. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, June 2016.

[5] A. Fader, L. S. Zettlemoyer, and O. Etzioni. Paraphrase-driven learning for open question answering. In *ACL*, 2013.

[6] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM Workshop on Recurring Malcode*, 2007.

[7] D. Gragg. A multi-level defense against social engineering. Technical report, SANS Institute, December 2002.

[8] T. Greening. Ask and ye shall receive: A study in social engineering. *SIGSAC Rev.*, 14(2), Apr. 1996.

[9] C. Hadnagy. *Social Engineering The Art of Human Hacking*. Wiley Publishing Inc., 2011.

[10] C. Hadnagy and P. Wilson. *Social Engineering: The Art of Human Hacking*. Wiley, 2010.

[11] C. Hauser. U.s. breaks up vast i.r.s. phone scam. New York Times, July 23 2018.

[12] G. Ho, A. Sharma, M. Javed, V. Paxson, and D. Wagner. Detecting credential spearphishing in enterprise settings. In *26th USENIX Security Symposium (USENIX Security 17)*, 2017.

[13] A. Karakasilitiosis, S. M. Furnell, and M. Papadaki. Assessing end-user awareness of social engineering and phishing. In *Australian Information Warfare and Security Conference*, 2006.

[14] B. Klimt and Y. Yang. *The Enron Corpus: A New Dataset for Email Classification Research*. 2004.

[15] J. A. Kreibich. *Using SQLite*. O'Reilly Media, Inc., 1st edition, 2010.

[16] M. Lamagna. Heres how much phone scams cost americans last year... marketwatch.com, April 21 2017.

[17] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[18] K. Mitnick and W. Simon. *The Art of Intrusion: The Real Stories Behind the Exploits of Hackers, Intruders and Deceivers*. Wiley, 2009.

[19] G. L. Orgill, G. W. Romney, M. G. Bailey, and P. M. Orgill. The urgency for effective user privacy-education to counter social engineering attacks on secure computer systems. In *Proceedings of the 5th Conference on Information Technology Education*, 2004.

[20] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[21] J. Scheeres. *Establishing the Human Firewall: Reducing an Individual's Vulnerability to Social Engineering Attacks*. Bibliioscholar, 2012.

[22] A. Stone. Natural-language processing for intrusion detection. *Computer*, 40(12), Dec 2007.

[23] G. Stringhini and O. Thonnard. That ain't you: Blocking spearphishing through behavioral modelling. In *Proceedings of the 12th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment - Volume 9148*, DIMVA 2015, 2015.

[24] M. Workman. A test of interventions for security threats from social engineering. *Inf. Manag. Comput. Security*, 16(5), 2008.