Detection of Social Engineering Attacks Through Natural Language Processing of Conversations

Yuki Sawa[1], Ram Bhakta[1], Ian G. Harris[1] and Christopher Hadnagy[2]

[1]Department of Computer Science, University of California Irvine, USA
ysawa@uci.edu, hiten.bhakta@gmail.com, harris@ics.uci.edu

[2]socialengineer.com, chris@social-engineer.com

Abstract

As computer security approaches improve, social engineering attacks have become more prevalent because they exploit human vulnerabilities which are hard to automatically protect. We present an approach to detecting a social engineering attack which applies natural language processing techniques to identify suspicious comments made by an attacker. Social engineering attacks involve either *questions* which request private information, or *commands* which request the listener to perform tasks which the speaker is not authorized to perform. Our approach uses natural language processing techniques to detect questions and commands, and extract their likely topics. Each extracted topic is compared against a topic blacklist to determine if the question or command is malicious. Our approach is generally applicable to many attack vectors since it relies only on the dialog text. We have applied our approach to analyze the transcripts of several attack dialogs and we have achieved high detection accuracy and low false positive rates in our experiments.

I. Introduction

A critical threat to information security is *social engineering*, the act of influencing a person to take an action that may or may not be in their best interest. For the purpose of this study, we focus on the acts that would not be in the targets best interests, such as using psychological manipulation of people in order to gain access to a system for which the attacker is not authorized [1], [2], [3]. Cyberattackers target the weakest part of a security system, and people are often more vulnerable than a hardened computer system. All manner of system defenses can often be circumvented if a user reveals a password or some other critical information. Social engineering is a modern form of the confidence scam which grifters have always performed. Phishing emails, which fraudulently request private information, are a common version of the attack, but social engineering comes in many forms designed to exploit psychological weaknesses of the target. A study by Verizon of security breaches in 2013 has shown that 29% of all security breaches involve social engineering to extract information for use primarily for phishing, bribery, and extortion [4]. These attacks were executed primarily via email but also in-person, via phone, SMS, websites, and other documents. The frequency and effectiveness of social engineering makes it a serious security issue which must be addressed.

We present an approach to the detection of social engineering attacks by examining all text transmitted from the attacker to the victim and checking the appropriateness of the statement topic. A statement is considered to be inappropriate if it either requests secure information or requests the performance of a secure operation. Our approach uses natural language processing (NLP) techniques to detect questions and commands, and to extract their likely topics. Each sentence is parsed and the resulting parse tree is searched for patterns which are indicative of questions and commands. Once a question or command is detected, its potential topics are identified by locating verb/noun pairs which relate verbs with their direct objects. Each topic is evaluated by comparing it to the contents of a *topic blacklist* which describes all forbidden topics of conversation with the victim. We assume that the topic blacklist will be derived manually either based on a basic understanding common security requirements, or an existing security policy document associated with a system.

A. Detection Example

To concretize our discussion on detecting social engineering attacks, we next motivate our approach with an example of attack detection. For this example we assume that social engineering is used to attack the information technology infrastructure of a corporate entity. We assume that there is a security policy document which has been manually processed to define an appropriate topic blacklist. Figure 1 shows a statement found in the policy document which is used to define a blacklist entry.

TBL 1: Networking equipment must not be manipulated.

Fig. 1. Security policy statement

The statement in Figure 1 is manually analyzed to generate the blacklist entry shown in Table I.

	Action	Resource
Entry 1	manipulate	networking equipment

TABLE I
Topic blacklist entry

For this example we assume that the social engineering attack is launched via texting, social media chat, or email, so that the dialog is already provided in text form. At some point during the dialog, the attacker makes the statement, “Reset the router”. The statement is recognized as a command by examining its parse tree and detecting a pattern which is common in direct commands. The topic of the command is extracted as *(reset, router)*, which are the verb and its direct object. The verb, “reset”, is a type of manipulation, so it matches the action “manipulate” in Entry 1. Also, the direct object of the predicate, “router”, is a type of networking equipment, so it matches the resource in Entry 1. Since the statement’s topic matches a topic blacklist entry, access will be denied and a warning message is transmitted to the victim.

II. Related Work

Existing techniques for detection and prevention of the broader class of social engineering attacks depend on training potential victims to be resistant to manipulation. Training regimens have been proposed in previous work [5], [6] which educate users on the techniques used in previous attacks, and the importance of various pieces of information. A multi-level training approach has been presented which matches the degree of training to the responsibility level of the user [6]. Training techniques depend on the users awareness of his/her mental state and thought processes, referred to as *metacognition* [6]. A social engineering detection approach presented in previous work also depends on the user’s metacognition in order to answer a sequence of security-related questions before providing data to an external agent [7]. In general, approaches which rely on the the cognitive abilities of the user will be unreliable since the mental state of users vary greatly, and can often be manipulated by a clever attacker.

Previous work has identified social engineering attacks in conversations by locating malicous keyword pairs on the same line of text [8]. This approach performed well in the detection of generic email and chat attacks, but the approach does not perform syntactic or semantic analysis, so it would not perform well with sophisticated targeted attacks.

III. System Outline

The first step of our approach is to parse the sentence using a context-free grammar of English, to generate a parse tree. The parse tree is analyzed during the **Question/Command Detection** step to identify patterns in the parse tree which are indicative of questions and commands. If a sentence is identified as a question or command then the subtree of its parse tree which contains the question or command is analyzed by **Topic Extraction** to find potential verb/noun *topic pairs* which describe the topic of the question or command. The topic pairs are compared to the topics in the topic blacklist to determine whether or not the topic of the question or command is considered to be malicious.

We use the Stanford Parser [9] to parse each sentence and generate a parse tree. The Stanford Parser is a probabilistic context-free parser which is a well used open source project in the natural language processing community. The symbols used in the parser’s grammar are from the Penn Treebank Tagset [10] which includes all accepted parts of speech. The following sections describe the Question/Command Detection and Topic Extraction steps in detail, as well as the construction of the topic blacklist.

IV. Question/Command Detection

The goal of this step is to examine the parse tree of a sentence to determine whether or not the sentence contains a question or a command. If the sentence is found to contain a question or a command, the subtree of the parse tree which expresses the question or command is identified and passed to the next step. Each sentence contains one or more clauses and each clause can express a number of different speech acts [11], [12]. Clauses can be categorized based on the speech act which they are used to perform, but we are interested in two types of clauses, 1) *imperative clauses* which are used to issue commands, and 2) *interrogative clauses* which are used to ask questions. Both imperative and interrogative clauses are associated with several syntactic forms which can be recognized as patterns in the parse tree of a sentence. We detect questions and commands by searching the parse tree of each sentence to find the patterns associated with imperative and interrogative clauses. We use the Tregex tool [13] to match patterns in parse trees. In the following sections we present the parse tree patterns for both imperative and interrogative clauses.

A. Command Detection

Clauses which express commands are called *imperative* clauses. A **direct imperative** clause does not contain a subject, as in the sentences, “Go home” and, “Stop right there”. Even without the subject it is clear that speaker is referring to the listener. A direct imperative clause is recognized as a clause with a verb but no subject. The subject of a sentence cannot be recognized directly by a traditional English parser since the term *subject* refers to a semantic role rather than a part of speech. For the purpose of command detection, we identify the subject of a clause as a noun phrase before the main verb in the clause. This is usually the case because the subject appears before the verb in most English sentences. Since noun phrases and verbs are parts of speech which are identified by the Stanford parser, we identify a direct imperative clause as one which does not contain a noun phrase before the verb. Only noun phrases contained in the main clause are considered and subordinate phrases are ignored. A subordinate clause contains a noun and a verb, but does not express a complete thought. The noun phrase in a subordinate clause is not considered to be the subject of the main clause. For example, the sentence, “when the car starts, go home”, contains the imperative clause “go home”. The noun phrase “the car” precedes the main verb “go” but the noun phrase is contained in the subordinate clause “when the car starts”, so it is ignored.

In many social settings, direct commands can seem rude and might alarm a victim to the fact that a social engineering attack is in process. For this reason, a speaker will often use a **soft imperative** which is a suggestion rather than a direct command. Soft imperatives include a second-person pronoun and a modal verb such as “you could” or “you should”. An example of a sentence expressing a soft imperative is, “you should shut down the router”. We detect a soft imperative clause by identifying a verb with a preceding modal verb and the pronoun “you” before that. In the case of the sentence above, the verb (VB) “shut” is preceded by the modal (MD) “should” and the pronoun (PRP) “you”.

B. Question Detection

Interrogative clauses are broadly classified as either *closed* (yes/no) questions which request a yes or no answer, or *open* questions which request a more elaborate answer. Yes/no questions are characterized by *subject/auxiliary inversion*, placing the auxiliary verb before the subject in the sentence. Auxiliary verbs are helper verbs such as “will”, “may”, or “can”, which add meaning (tense, aspect, etc.) to the clause. In a statement, the auxiliary verb is placed after the subject as in, “I can eat”. However, the sentence becomes a question when the subject and auxiliary positions are reversed as in, “Can I eat”. The Stanford Parser automatically detects subject/auxiliary inversion, and the Penn Treebank tagset includes several tags which are used to demark subject/auxiliary inversion. The **SQ** tag indicates an inverted yes/no question, and the **SINV** tag indicates inversion of the subject and auxiliary which is associated with closed questions.

Most open questions are *wh-questions* which use one of the wh-pronouns: who, whom, what, where, when, why, which, whose and how. The Stanford Parser automatically detects the use of wh-pronouns as part of different types of phrases, and the Penn Treebank tagset includes the following tags which are used to demark wh-phrases: **WHADJP** - wh adjective phrase, **WHAVP** - wh adverb phrase, **WHNP** - wh noun phrase, and **WHPP** - wh prepositional phrase. We detect a wh-question clause as one with a wh-pronoun at the beginning of the clause.

V. Topic Extraction

We capture the topic of a clause as a pair, the main verb of the clause and its direct object. Given a parse tree which represents the clause, the goal of this step is to identify all likely topic pairs. Identification of the main verb is straightforward based on the tags generated by the parser. To find the direct object we assume that the direct object is a noun or noun phrase located after the verb in the clause. This assumption is usually correct because the *Subject-Verb-Direct Object* sentence form is very common in practice. There may be several nouns after the verb, so our approach produces one potential topic pair involving the verb and each noun after the verb. In order to fully specify sentence topics in the presence of pronouns, it is necessary to perform *pronomial anaphora resolution*, determining the antecedent of a pronoun. Each pronoun must be replaced with the noun or noun phrase which it corresponds to in the context of the dialog. This is a well studied problem and acceptable algorithms have already been presented in previous work, such as the Hobbs Algorithm [14]. We have found that in common dialogs, a heuristic can be applied which replaces each pronoun with the noun phrase in the text which most recently precedes the pronoun. In the case of the example, “The pizza is good. Eat it”, the most recent noun preceding the pronoun is “pizza”, so we assume that “pizza” is the antecedent. This heuristic is simple, but it is sufficiently accurate for this purpose.

VI. Topic Blacklists

The topic blacklist is the list of statement topics whose discussion would indicate a violation of security protocol. Each topic either describes sensitive data or a sensitive operation. Each topic is composed of two elements, an *action* and a *resource*. The *action* describes an operation which the subject may wish to perform. The *resource* is the resource inside the system to which access is restricted.

We assume that the topic blacklist will be derived manually either based on a basic understanding common security requirements, or from an existing security policy document associated with a system. The blacklist would be created to match the security requirements of the system which it is being used to protect. A blacklist used to protect a regular individual would contain generic topics such as {“send”, “money”} or {“tell”, “social security number”}. A topic blacklist used by a corporate entity would contain topics related to the business of that entity.

VII. Results

We have evaluated our system by applying it to two different corpora, The first corpus is a set of threee actual social engineering attacks which are known to contain malicious sentences. The second corpus is the Supreme Court Dialogs Corpus [15] which is known not to contain any social engineering attacks.

The first corpus, containing three social engineering attacks, is used to demonstrate that our approach successfully detects malicious sentences in social engineering attacks. The corpus contains a set of three social engineering dialogs which we refer to as Dialogs A, B, and C. Each of these dialogs is the transcript of a phone conversation between a professional social engineer acting as an attacker, and a victim working at a company with an internal computer system. The social engineers goal in each dialog is to extract information about the local computer system and to convince the victim to download and execute code provided by the attacker. In order to achieve these goals, the pentester used social engineering techniques described in previous work [3]. During the audits, the victims are unaware that the calls are part of an audit being performed, so their responses and other reactions are genuine, reflecting a true attack scenario. We have received all necessary approvals to evaluate anonymized versions of the dialog transcripts.

Table II shows information about each dialog in the first corpus. The second column is the number of sentences spoken by the attacker in the dialog. We do not include the number of sentences spoken by the victim because our approach does not evaluate those sentences. The third column is the number of sentences in each dialog which contain malicious questions or commands which we have identified manually.

Dialog	Sentences	Violations
A	24	3
B	30	6
C	20	1

TABLE II
SOCIAL ENGINEERING ATTACK DIALOGS, FIRST CORPUS

The second corpus, containing conversations from the U.S. Supreme Court Oral Arguments [16], is used to evaluate the false positive rate of our approach. Since this corpus contains no social engineering attacks, our approach should detect no malicious sentences. The entire courpus contains 51,498 utterances and we evaluated the first 10,000 sentences in the corpus.

Table III shows the topic blacklist which we use. The blacklist was manually generated based on our understanding of common generic social engineering attack goals. Synonyms for the terms in the blacklist are not shown.

Action	**Resource**
download	file
open	Internet Explorer
update	records
give	social
give	address
give	cert

TABLE III
TOPIC BLACKLIST

A. Corpus 1 Results

The results of applying our tool to the first corpus is shown in Table II. There are a total of 10 malicious sentences out of a total of 74 sentences. Our tool detected 6 of the 11 malicious sentences, with no false positives. We compute $precision = \frac{tp}{tp+fp}$ and $recall = \frac{tp}{tp+fn}$ metrics according to their standard definitions. In these equations, tp is the number of true positives, fp is the number of false positives, and fn is the number of false negatives. Our results for the first corpus contain 6 true positives, 0 false positives, and 4 false negatives. The precision of our approach is 100% and the recall is 60%.

Corpus 1 experiments were run on an Intel i5 processor, 3.4 GHz. The performance results of each step are shown in the following list.

- **Sentence Parsing** 5.357 seconds
- **Question/Command Detection** 0.101 seconds
- **Topic Extraction** 0.004 seconds
- **Topic Comparison** $<$0.001 seconds

These measurements are the CPU time required to process all 74 sentences. Sentence Parsing and Question/Command Detection are performed 10 times for each sentence and the CPU time required is reflected in these performance results.

B. Corpus 2 Results

The results of applying our tool to the second corpus are as follows. Of the 10,000 sentences in the corpus, no malicious sentences were detected. Since the corpus is known to contain no social engineering attacks, it is clear that our tool achieved 0 false positives for this corpus as well.

Corpus 2 experiments were performed on an Intel Xeon, 2.4 GHz, with 12 cores. Total CPU time for execution of all stages was 2421 seconds.

VIII. CONCLUSIONS

We present an approach to detect social engineering attacks by verifying discussion topics against a topic blacklist. The approach is robust enough to effectively analyze the language of real attacks. The use of natural language processing techniques to detect sentence type and extract topics is novel. The definition of the topic blacklist is manual but we do not believe that it is onerous for a person with an understanding of the security requirements of the system being protected. The performance of our tool is good enough to provide attack warnings in real time during a conversation to prevent the victim from violating security protocol.

REFERENCES

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

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

[3] C. Hadnagy, P. F. Kelly, and E. P., *Unmasking the Social Engineer: The Human Element of Security*. Wiley, 2014.

[4] *2013 Data Breach Investigations Report*. Verizon, 2013. [Online]. Available: http://books.google.com/books?id=YXi0nQEACAAJ

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

[6] D. Gragg, "A multi-level defense against social engineering," SANS Institute, Tech. Rep., December 2002.

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

[8] R. Bhakta and G. Harris, I., "Semantic analysis of dialogs to detect social engineering attacks," in *IEEE International Conference on Semantic Computing (ICSC)*, Feb 2015.

[9] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, 2003.

[10] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn treebank," *Comput. Linguist.*, vol. 19, no. 2, Jun. 1993.

[11] L. Austin, J., *How to do things with words*. Oxford University Press, 1962.

[12] J. Searle, *Speech acts: an essay in the phiolosophy of language*. Cambridge University Press, 1969.

[13] R. Levy and G. Andrew, "Tregex and tsurgeon: tools for querying and manipulating tree data structures," in *International Conference on Language Resources and Evaluation*, 2006.

[14] J. Hobbs, "Resolving pronoun references," in *Readings in Natural Language Processing*, B. J. Grosz, K. Sparck-Jones, and B. L. Webber, Eds., 1986.

[15] C. Danescu-Niculescu-Mizil, B. Pang, L. Lee, and J. Kleinberg, "Echoes of power: Language effects and power differences in social interaction," in *Proceedings of WWW*, 2012.

[16] [Online]. Available: http://www.supremecourt.gov/