SEADer++ v2: Detecting Social Engineering Attacks using Natural Language Processing and Machine Learning

Merton Lansley
Department of Informatics
University of Sussex
Brighton, United Kindom
M.Lansley@Sussex.ac.uk

Stelios Kapetanakis
School of Computing,
Engineering and Mathematics
University of Brighton
Brighton, United Kingdom
S.Kapetanakis@Brighton.ac.uk

Nikolaos Polatidis
School of Computing,
Engineering and Mathematics
University of Brighton
Brighton, United Kingdom
N.Polatidis@Brighton.ac.uk

***Abstract*—Social engineering attacks are well known attacks in the cyberspace and relatively easy to try and implement because no technical knowledge is required. In various online environments such as business domains where customers talk through a chat service with employees or in social networks potential hackers can try to manipulate other people by employing social attacks against them to gain information that will benefit them in future attacks. Thus, we have used a number of natural language processing steps and a machine learning algorithm to identify potential attacks. The proposed method has been tested on a semi-synthetic dataset and it is shown to be both practical and effective.**

Keywords—Social Engineering, Attack detection, Cyber security, Natural language processing, Machine Learning

I. Introduction

Social engineering attacks are based on human weaknesses or the lack of knowledge about them, thus specific technical knowledge is not required from the side of the attacker. That's the main reason why these types of attacks are highly employed and are attractive to applied to others. In the past there were various approaches arguing theoretically that social engineering detection is important [1-4]. Apart from those there are some practical works based on natural language processing (NLP) and machine learning, including neural networks [5-6]. It is also important to mention the important role that psychology plays in social engineering attacks and the part that it plays in NLP when software needs to be developed to detect attacks.

There are different online environments such as business domains where customers talk to employees through online chat and in social networking where people talk to each other which are considered very attractive to apply social engineering attacks. There are research approaches that consider attacks in such domains and apply NLP and machine learning to detect attack successfully to an extent [5-6]. To detect attacks written in human language such as English pre-processing of the dialogue is necessary using a parser such as the Stanford core NLP [7]. By using such a library then a classification dataset can be created, and a classifier applied. This paper builds on top of the algorithm found in [6]. The proposed method is based on the 12 steps of the SEADer++ algorithm which are also explained in section 3 and 2 more pre-processing steps are added to improve the quality of the output as shown in the evaluation section.

This paper delivers the following contributions:

1. A method for detecting social engineering attacks is proposed.
2. The proposed method has been evaluated using a semi-synthetic dataset and the results show that it is both practical and effective.

The rest of the paper is organized as follows: Section 2 is the related work, section 3 describes the proposed method, section 4 explains the experimental evaluation and section 5 contains the conclusions.

II. Related work

One of the first works in social engineering attack detection is the one in ref [8]. The research in this work is based on real time telephone systems and focused on the identification on voice signatures of repeated voice calls and a proof of concept was developed to identify the attacks in the dialogues dataset. One more recent work is the one in ref [4] where the authors identified questions requesting private information manually using a small database of verb-noun pairs. This work extended by [2] where the authors identified 4 attack vectors: the urgency of the dialog, negative commands and questions, whether the

978-1-7281-6799-2/20/$31.00 ©2020 IEEE

message is likely automated identified by a generic greeting and then check the URL and generated a topic blacklist using the Naïve Bayes classifier. Other proposed solutions such as [1] and [9] use complex state machines to map pathways that can be followed in order to mitigate an attack. The work in [3] is good in terms of a small survey on social engineering attacks by providing a relatively comprehensive background on them and on potential solutions. In [10] the authors provide an approach based on semantics of dialogues to detect social engineering attacks. The authors in [11] shown that humans are the weakest link in social engineering attacks and their study shows results that proves that. In [12] and [13] the authors provided theoretical work that can be potentially used to implement real systems. The authors of [14] give a good explanation of how social engineering attacks can be detected whereas in [15] and [16] different attack scenarios are delivered. More practical works include the ones in [5] and [6] where the authors use NLP and machine learning to detect attacks successfully.

In addition to that there are other works that can be useful such as the one in [17] where the authors performed influence analysis on the quality of knowledge in a collective. The one in [18] a neural network is used with harmony for searching and the one in [19] where nature inspired optimization is discussed. In [20] an Island-based cuckoo search algorithm with highly disruptive polynomial mutation has been proposed.

III. Proposed method

There are several steps that need to be completed in the proposed method. Initially, the text is read using natural language processing and then is being processed into a classification dataset. At the last step a classifier such as Random Forest, Multi-Layer perceptron or others can be applied to see which entry is an attack or not. The python programming language has been used for all step along with the SymSpellpy library for spelling errors, the Web of Trust (WOT) to check if a link is malicious or not

The proposed method consists of several steps to preprocess the dialogs into a dataset for classification. The last steps are applying the classifier. All the steps explained below, have been written in the Python programming language. The SymSpellpy library (a Python port of SymSpell) was used for spelling, the Web of Trust (WOT) Application Programming Interface (API) was used to check any links. the SciKit library for the classifiers and the Gensim Latent Dirichlet Allocation (LDA) model was used for the estimation of topics for the dialogs.

Steps 1 to 5 are applied to check if links are malicious, steps 6 and 7 determine quality of the spelling, and steps 8 to 11 determine the intention and a pre-defined blacklist is used for this. Feature scoring can be given by: Link score, S_L, Spelling score, S_{sp}, and Intent score, S_I. Each score is scaled down to a value between 0 and 1. After the pre-processing of the dialogues (steps 1 – 11), the classification dataset has the following 4 labels: (1) Intent, (2) Spelling, (3) Link (4) XXX (5) XXX (6) XXX and (7) attack or no attack. The final steps use these as inputs for any classifier.

The steps of the proposed method are as follows:

1. URLs from a dialog are extracted text using a regex pattern matcher.
2. If the text contains URLs, the link(s) is sent to the WOT to check if it is malicious.
3. The WOT then returns a reputation value between 0 and 100 of the sites, the confidence of the given reputation between 0-100 is also returned and the 1 out of 17 identifying categories that identify the nature of the website is also returned. The broad categories and example subcategories are as follows:
    - 1XX Negative (101 Malware, 103 Phishing, 104 Scam, 105 Potentially illegal etc.)
    - 2XX Questionable (201 Misleading claims or unethical, 205 spam, 207 ads / popups etc.)
    - 3XX Neutral (301 Online tracking, 302 controversial, 303 political etc.)
    - 5XX Positive (501 A good site)
4. If the returned category is of group 1XX or 2XX, then $S_L = 1$.
5. Otherwise, divide the reputation by 100 and take it away from 1 as shown in equation 1.

$$S_L = 1 - \frac{reputation\ value}{100} \tag{1}$$

6. At this step the SymSpellpy library is used to check for spelling.

7. The number of misspelled words is determined and is given by x. This number is then converted between 0 and 1 by utilizing Equation 2. The value of S_{SP} is the spelling quality, where values that are high represent poorer spelling. An exponential function is used instead of a linear one to rate a higher number of spelling mistakes. To adjust the rate at which the score tends towards 1, the constant a can be varied to affect how you want to punish spelling errors. After a series of testing it was identified that 0.5 allowed the text to contain a small number of mistakes without creating a score of a high value. For example, if a is set to 0.5 and $x = 1$ then $S_{SP} = 0.39$, if $x = 5$ then $S_{SP} = 0.92$.

$$S_{SP} = 1 - e^{-ax} \tag{2}$$

8. At this step the corrected spelling of the dialog is used, and it is checked against a blacklist, derived from security policy dictionary of 48 words. This can be populated easily with environmental or company related words such as: passwords, credentials, database and others. The number of blacklist matched words is given by M_B.
9. At this step the algorithm checks for intent verbs and adjectives such as must, need, urgent and others. This value is given by M_I
10. To tune the results, values M_B and M_I are multiplied by the weights W_B and W_I, weighted at values of 2 and 1 retrospectively. This was added as an equation in case it is necessary to change the weight of this step, if it is considered important compared to others. The value x can be given by equation 3.

$$x = (M_B \times W_B) + (M_I \times W_I) \quad (3)$$

11. At this step, the value of x is normalized using equation 4, by applying the same exponential function as step 7 (equation 2). Where $a = 0.4$ to give the best output. A higher value of S_I indicates a higher concentration of blacklisted words in the text.

$$S_I = 1 - e^{-ax} \quad (4)$$

12. Then, the original dialogue dataset is checked to identify which dialogue was indeed an attack and assign the true (1) or false (0) value to the new classification column in the dataset.

13. The LDA probabilistic generative model is used to infer the topic structure, β. The number of topics are represented by $\beta_{1:K}$, where each β_k is the distribution over the vocabulary. The topic proportions for any given document is given by θ_d. The assigned topics are z_d where $z_{d,n}$ is the topic assignment for the nth word in the document, *d*. The observed words for document *d* are given by w_d . The generative model showing the hidden and observed variables in the LDA can be seen in Equation 5 [21].

$$p(D|\alpha,\beta) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\Theta_d) \prod_{n=1}^{N} p(z_{d,n}|\Theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n}) \quad (5)$$

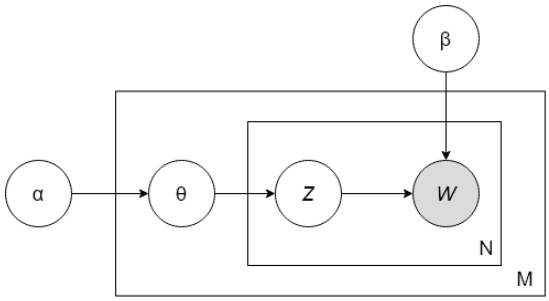

Fig. 1. LDA generative process

14. The inference method to determine the distribution of the topic structure given the observed documents (posterior), is given in Equation 6.

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})(p(w_{1:D}) \quad (6)$$

IV. Experimental evaluation

For the experimental evaluation we have used a real dataset and the accuracy, precision and area under the curve metrics.
The dataset consists of 1,051 entries and was originally derived from social engineering dialogues as described in [8] and was later extended and used in [6]. The extended dataset from [6] was used which has 4 columns:
1. Intent
2. Spelling
3. Link
4. Attack or not

This dataset was based on the dialogues from [8] and then the preprocessing steps 1 to 12 were applied to create it. Then, steps 13 and 14 were applied and 3 more columns were created in the classification dataset which now has the following 7 columns:

1. Intent
2. Spelling
3. Link
4. Topic modeling
5. Topic modeling
6. Topic modeling
7. Attack or not

4, 5 and 6 represent 3 topics that the dialogue(s) is based on and how close is to each topic (value between 0 and 1). The dataset is available online[1].

To evaluate the quality of the algorithm the accuracy, precision and area under the roc curve (AUC) metrics have been used. Accuracy is defined in equation 7 and Precision is defined in equation 8. TP stands for true positives, TN for true negatives, FP for false positives and FN for false negatives. All executions are based on a 5-fold cross validation approach using the Python programming language and the Scikit library.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (7)$$

$$Precision = \frac{TP}{TP + FP} \quad (8)$$

The accuracy results of the proposed method are presented in table 1 and it is shown the random forest classifier is better when compared to the MLP and K nearest neighbor (KNN). In table 2 the Precision results are presented which show again that the random forest classifier is better. Each algorithm was executed 5 times and at the end the average value of these 5 times is presented.

TABLE I. ACCURACY RESULTS

EXECUTION	RANDOM FOREST	MLP	KNN
1	0.796	0.777	0.791
2	0.787	0.779	0.763
3	0.784	0.782	0.781
4	0.794	0.773	0.779
5	0.795	0.764	0.759
AVERAGE	0.791	0.775	0.774

TABLE II. PRECISION RESULTS

EXECUTION	RANDOM FOREST	MLP	KNN
1	0.783	0.781	0.775
2	0.807	0.764	0.773
3	0.781	0.772	0.774
4	0.796	0.778	0.751
5	0.783	0.777	0.805
AVERAGE	0.790	0.774	0.775

Having seen now that the random forest classifier behaves better for this dataset, figure 2 presents an accuracy comparison between 10 and 100 estimators for the random forest classifier and figure 3 similarly presents a precision comparison. The results show that having 100 estimators is slightly better compared to having 10. The average accuracy for 10 is 79.2% and for 100 is 80.1 while the average precision for 10 if 79.4% and for 100 is 79.7%.

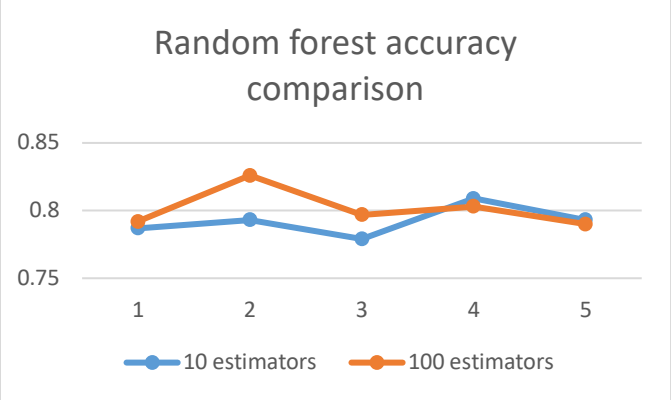

Fig. 2. Accuracy comparison results for random forest

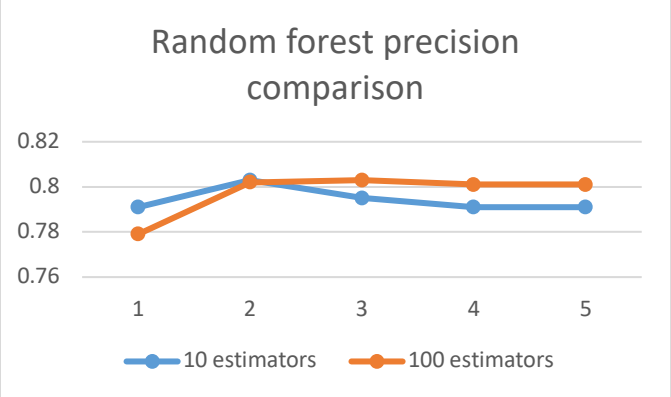

Fig. 3. Precision comparison results for random forest

[1] https://github.com/npolatidis/SEADer-v2

Figure 4 presents an accuracy comparison between the proposed method and SEADer++ [6] both for the same dataset and using a random forest classifier, while a precision comparison is shown in figure 5. The proposed method achieves 80.1% accuracy while SEADer++ achieves 78.4% and in terms of precision the proposed method achieves 79.7% compared to 80.1%.

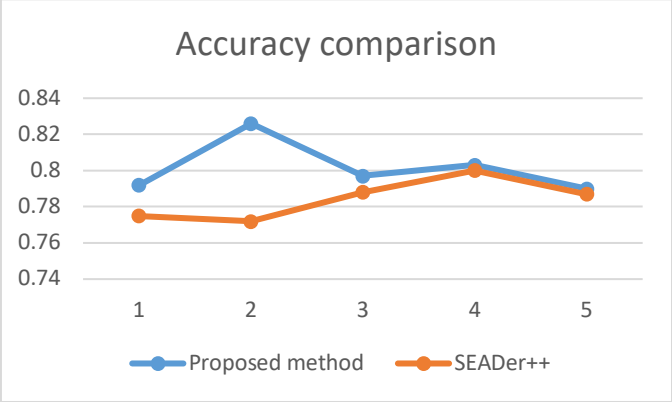

Fig. 4. Accuracy comparison

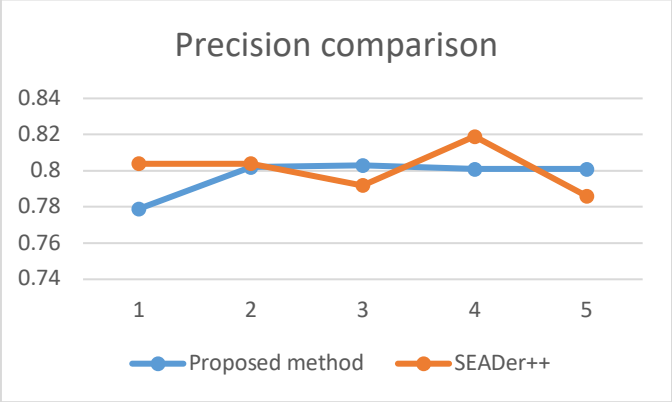

Fig. 5. Percision comparison

Figure 6 shows the area under the curve results for which the average of the proposed method is 89.2% and for SEADer++ is 81.6%.

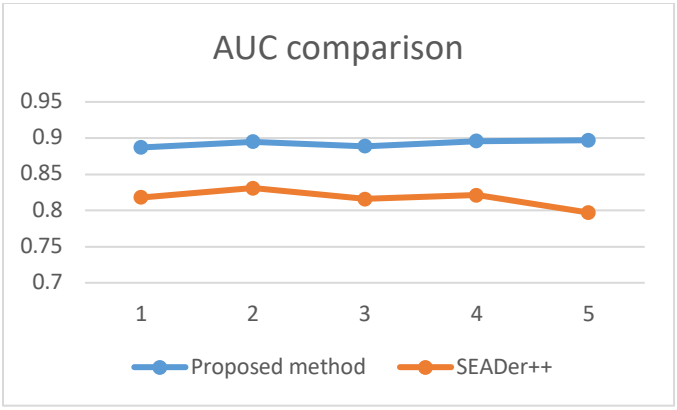

Fig. 6. AUC comparison

V. Conclusions

In this paper we presented a social engineering attack detection method that is based on NLP and machine learning. The proposed method has been evaluated using a semi-synthetic dataset and well-known metrics along with a comparison to a state-of-the-art alternative method. It has been shown that it performs well and outperforms the alternative methodology.

In the future we aim to explore how the use of fuzzy logic techniques with machine learning algorithms can improve detection quality.

References

[1] Mouton, F., Nottingham, A., Leenen, L., & Venter, H. S. (2018). Finite state machine for the social engineering attack detection model: Seadm. *SAIEE Africa Research Journal*, *109*(2), 133–148. https://doi.org/10.23919/SAIEE.2018.8531953

[2] Peng, T., Harris, I., & Sawa, Y. (2018). Detecting phishing attacks using natural language processing and machine learning. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)* (pp 300–301).

[3] Tsinganos, N., Sakellariou, G., Fouliras, P., & Mavridis, I. (2018). Towards an automated recognition system for chat-based social engineering attacks in enterprise environments. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ARES 2018, (pp. 53:1–53:10). ACM.

[4] Sawa, Y., Bhakta, R., Harris, I. G., & Hadnagy, C. (2016). Detection of social engineering attacks through natural language processing of conversations. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)* (pp. 262–265).

[5] Lansley, M., Polatidis, N., & Kapetanakis, S. (2019). Seader: A social engineering attack detection method based on natural language processing and artificial neural networks. In N. T. Nguyen, R. Chbeir, E. Exposito, P. Aniorté, and B. Trawiński (Eds.), *Computational collective intelligence* (pp. 686–696). Springer International Publishing.

[6] Merton Lansley, Francois Mouton, Stelios Kapetanakis & Nikolaos Polatidis (2020) SEADer++: social engineering attack detection in online environments using machine learning, Journal of Information and Telecommunication,DOI: 10.1080/24751839.2020.1747001

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

[8] Hoeschele, M., & Rogers, M. (2005). Detecting social engineering. In Pollitt, M. and Shenoi, S. (Eds.), *Advances in digital forensics* (pp. 67–77). Springer US.

[9] Jamil, A., Asif, K., Ghulam, Z., Nazir, M. K., Mudassar Alam, S., & Ashraf, R. (2018). Mpmpa: A mitigation and prevention model for social engineering based phishing attacks on facebook. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 5040–5048).

[10] Bhakta, R., & Harris, I. G. (2015). Semantic analysis of dialogs to detect social engineering attacks. In *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*(pp. 424–427).

[11] Heartfield, R., & Loukas, G. (2018). Detecting semantic social engineering attacks with the weakest link: Implementation and empirical evaluation of a human-as-a-security-sensor framework. *Computers and Security*, *76*, 101–127. https://doi.org/10.1016/j.cose.2018.02.020

[12] Bezuidenhout, M., Mouton, F., & Venter, H. S. (2010). Social engineering attack detection model: Seadm. In *2010 Information Security for South Africa* (pp. 1–8).

[13] Mouton, F., Leenen, L., & Venter, H. S. (2015). Social engineering attack detection model: Seadmv2. In *2015 International Conference on Cyberworlds (CW)* (pp. 216–223).

[14] Nicholson, J., Coventry, L., & Briggs, P. (2017). Can we fight social engineering attacks by social means? Assessing social salience as a means to improve phish detection. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)* (pp. 285–298). USENIX Association.

[15] Krombholz, K., Hobel, H., Huber, M.,& Weippl, E. (2015). Advanced social engineering attacks. *Journal of Information Security and Applications*, *22*, 113–122. Special Issue on Security of Information and Networks. https://doi.org/10.1016/j.jisa.2014.09.005

[16] Mouton, F., Leenen, L., & Venter, H. S. (2016). Social engineering attack examples, templates and scenarios. *Computers and Security*, *59*, 186–209. https://doi.org/10.1016/j.cose.2016.03.004

[17] Nguyen, N. T. (2016). An influence analysis of the inconsistency degree on the quality of collective knowledge for objective case. In *Asian conference on intelligent information and database systems*(pp. 23–32). Berlin: Springer.https://doi.org/10.1007/978-3-662-.

[18] Saadat Javad, P. M., & Koofigar, H. (2017). Training echo state neural network using harmony search algorithm *International Journal of Artificial Intelligence*, *15*(1), 163–179.

[19] Precup, R.-E., & David, R. C. (2019). Nature-inspired optimization algorithms for fuzzy controlled servo systems. Butterworth-Heinemann.

[20] Abed-alguni, B. H. (2019). Island-based cuckoo search with highly disruptive polynomial mutation. *International Journal of Artificial Intelligence*, *17*(1), 57–82.

[21] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993-1022.