Seen the villains: Detecting Social Engineering Attacks using Case-based Reasoning and Deep Learning

Merton Lansley[1], Nikolaos Polatidis[1], Stelios Kapetanakis[1], Kareem Amin[2], George Samakovitis[3], Miltos Petridis[4]

[1]School of Computing, Engineering and Mathematics, University of Brighton, BN2 4GJ, Brighton, U.K
{M.Lansley, N.Polatidis, S.Kapetanakis}@brighton.ac.uk

[2]German Research Center for Artificial Intelligence, Smart Data and Knowledge Services, Trippstadter Strasse 122, 67663 Kaiserslautern, Germany
kareem.amin@dfki.uni-kl.de

[3]Department of Computing and Information Systems University of Greenwich, Greenwich, UK
g.samakovitis@gre.ac.uk

[4]Department of Computing, University of Middlesex, London, UK
m.petridis@mdx.ac.uk

Abstract. Social engineering attacks are frequent, well-known and easy-to-apply attacks in the cyber domain. Historical evidence of such attacks has shown that the vast majority of malicious attempts against both physical and virtual IT systems were based or been initiated using social engineering methods. By identifying the importance of tackling efficiently cybersecurity threats and using the recent developments in machine learning, case-based reasoning and cybersecurity we propose and demonstrate a two-stage approach that detects social engineering attacks and is based on natural language processing, case-based reasoning and deep learning. Our approach can be applied in offline texts or real time environments and can identify whether a human, chatbot or offline conversation is a potential social engineering attack or not. Initially, the conversation text is parsed and checked for grammatical errors using natural language processing techniques and case-based reasoning and then deep learning is used to identify and isolate possible attacks. Our proposed method is being evaluated using both real and semi-synthetic conversation points with high accuracy results. Comparison benchmarks are also presented for comparisons in both datasets.

Keywords: Social Engineering, Deep Learning, Case-based Reasoning, Natural Language Processing, Attack Detection, Cybersecurity.

1 Introduction

Social Engineering (SE) is an umbrella term for a series of methods and techniques where attackers are luring users into revealing sensitive, confidential and personal information to be used in committing fraud and other illegal activities. This challenge is well acknowledged, however, there is no SE detection mechanism or methodology that can dynamically detect and adapt to SE approaches and "patterns" being developed constantly. Several research projects have been proposing the need for an automated system to recognize SE attacks based most commonly on Natural Language Processing (NLP) and Machine Learning (ML) techniques. Whilst these exist there has been little of the rate of success [1-4]. SE is based on human psychology and the method of persuasion relating to the 6 principles of Authority, Scarcity, Liking/Similarity, Reciprocation, Social Proof and Commitment/Consistency. Work in the area of Computer Science and SE is usually converting these principles to the stages of: Data-Preprocessing, Feature Extraction, Feature Scoring and Aggregation. To determine whether a dialog is a SE attack, certain criteria should be evaluated against a number of features. We select these features prior the three stages and we base them on the principles of persuasion, common SE tactics like malicious links and the history of the attacker. Further feature classification generates a 'score' in term of how possible the selected dialog is to match the SE criteria.

This work is based on the concept that a dialogue is taking place in an online chat environment where there are fair chances of SE taking place. Each implementation is designed to detect features by using a variety of classification techniques, such as: Case-based Reasoning, Fuzzy Logic, Topic Blacklists, Decision Trees, Random Forest and Deep Neural Networks depending on what needs to be extracted. An automated system requires the output of a clear decision, albeit a probabilistic decision. This can be done by aggregating the outputs from the Feature Extraction process. The results of each feature are weighted by importance, and at basic level can be averaged to give a Fuzzy logic prediction. By using more advanced techniques such as decision trees or neural networks, the weight of each feature can be calculated programmatically to determine which features carry the most importance regarding whether or not a social engineering attack is taking place.

The following contributions are delivered:

1. A method for detecting social engineering attack in online chat environments is proposed using CBR and Deep Learning
2. The proposed method is evaluated using a real and a semi-synthetic dataset with the results validating our approach.

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

2 Literature Survey

An early implementation of SE detection in real-time telephone systems is SEDA (Social Engineering Defense Architecture) [7]. The researchers mainly focused on identifying repeat callers using their voice signatures. Later a proof of concept model of SEDA [8] was produced being able to correctly identify all attacks in their dataset. A method of detection used by [1] is to identify the questions requesting private information and commands requesting that the user perform tasks they are not authorized to perform. This technique uses a manually derived topic blacklist of verb-noun pairs for which they state should be built around security policies associated with a system. This work is taken further in [2] by identifying 4 main attack; the urgency of the dialog, negative commands and questions, whether the message is likely automated identified by a generic greeting. Finally, they use a reputable cyber security service, Netcraft, to check the safety of a URL. Instead of manual blacklist creation, they use a large corpus of phishing emails to generate a topic blacklist using a Naive Bayes classier. Some approaches like SEADMv2 [3] and MPMPA [9] use complex state machines in order to map out the pathways that can be followed as a checklist-type system to mitigate an attack. This proposal is suited where there are multiple authorization layers that might prevent a request from being carried out. These two separate machines provide some overlap, but the explicit state definition required could be limiting to where these can be applied. The nature of SE attacks is unpredictable meaning that new methods of attacks are always being introduced. The SEADM versions state machines still rely on the user input for changing state, which means the chance for user error and naivety is still present. The works of [4] covers an extensive overview of the existing systems and provide a comprehensive recognition of subsystems for their detection architecture; in influence, deception, personality, speech act and past experiences. The work of [10] provides a semantic based approach of dialogues to detect social engineering attacks. In [11] the authors show that the human factor is the weakest link in social engineering attacks and based on a human study the prove that. In [12] and [13] the authors provide a theoretical foundation that potentially could be used in real systems to detect attacks., In the works of [14] it is explained how social engineering attacks can potentially be detected, whereas in [15] and [16] different examples of attacks with scenarios are presented. In addition to the works mentioned, there are numerous other articles from relevant domains that can be useful such as the one in [17] where the authors performed an influence analysis of the number of members on the quality of knowledge in a collective, the work in [18] where a neural network is used along with harmony for searching, the one in [19] where nature inspired optimization algorithms for fuzzy control servo systems are discussed and the one found in [20] where an Island-based cuckoo search algorithm with highly disruptive polynomial mutation is proposed

3 A CBR and Deep Learning approach for SE detection

A dialog in a social media engine, chat or web forum can reveal whether we do have a social engineering attempt or not. To evaluate our approach, we preprocess dialogs and convert them into a dataset for classification. A python pipeline has been created using SymSpellpy for spelling, a custom Case-based Reasoning system for link checking, Scikit learn for baseline classifiers and Keras for deep learning models. The score of each feature can be given by: Link score, S_L, Spelling score, S_{sp}, and

Intent score, S_I. Each score is then normalized and the value range resides 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 and (4) attack or no attack. The final steps use these as inputs for the classification mechanisms of an Convolutional Neural classifier.

The methodology is presented below in a concise way:

1. URLs are extracted from a dialog text using a URL regular expression pattern finder

2. If the text contains URLs, the URLs are send to a custom Case-based Reasoning system (CB-Ranker) to evaluate whether the web link is malicious or not

3. CB-Ranker classifies a link based in a set of features related both to its static structural attributes and attributes that pertain to its actual web content. Its features were inspired from Marchal et al. (2014) [21] and the concept of intra-URL relatedness that find "connections" between an investigated URL and text related to this URL. Table 1 shows the features that were considered:

Table 1. Intra-URL related features

Feature name	Description
URL - Rank	Relates to the traffic rank given from Alexa for the past three months
URL-Length	Fake links are usually hidden in long URL addresses
URL-AT	Checks whether anything after a @ sign is malicious
URL-DotCount	Excessive usage of dots indicates a non-secure website
URL-IPExistence	Usage of IP addresses can trick users, this feature investigates whether an IP is used
URL-LinksInCount	Identifies how many links a website has. This metric is obtained from Alexa
URL-Word-Number	Count the number of words in a URL

The final outcome of CB-Ranker is a "reputation score" for the website that scales between 0-10 followed by a Master Classification Tag (MCT). The MCT has 4 classes which can be Negative (the website is definitely malicious), Questionable (the website has elements of potentially malicious content e.g. ads/ popups), Neutral (various content but not necessarily malicious) and Positive (this pertains to reliable, non-malicious websites)

4. If the returned category is of group Negative or Questionable, 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. Check for spelling using the SymSpellpy library.

7. Using the best suggested spelling correction, determine the number of misspelled words, given by x. A score between 0 and 1 is then calculated based on Equation 2. The value of S_{SP} represents the spelling quality, where higher values represents

poor spelling. Equation 2 applies an exponential function to be able to penalize errors in more harsh way compared to a linear function

$$S_{SP} = 1 - e^{-ax} \quad (2)$$

8. Correct dialog spelling is then compared to a set of blacklisted words, derived from 48 security policy style words. This can be easily populated with company or environment related words such as: credentials, passwords, database etc. The number of blacklist matched words is given by M_B.

9. The algorithm at this step checks for intent verbs and adjectives such as need, must, urgent etc. 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 step has been added as an equation in case someone wants to change the weight of this step, if it is considered more important. The value x can hence be given by equation 3.

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

Then, the value of x is normalized in equation 4, using the same exponential function as 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.

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

11. At this step the original dialogue dataset is being checked to identify which dialogue was indeed an attack and assign the true (1) or false (0) value to the new dataset used for classification.

The dataset is populated and a Convolutional Neural Network (CNN) network is applied. If the output is high, then it is considered an attack otherwise it is not considered an attack. The CNN network has been trained over a corpus of dialog cases that have been tagged as SE attacks or not. For its training 3 datasets were used as its base including the targetedemailattacks from tumbler1[1] , ADFA[2] dataset, and a custom-cases one prepared for the purpose of this work. The network was trained using Keras python library.

4 Evaluation

For the evaluation purposes two datasets were used. The first was based on real cyber social engineering attacks whereas a second one was created to test the robustness of

[1] https://targetedemailattacks.tumblr.com/

[2] https://www.unsw.adfa.edu.au/

our proposed model classification as presented in section 3. Both datasets where relevant to specific SE patterns out of a population of 147 real entities. CBR was used to preprocess and classify our cases. The classification had four labels relating to
(a) intent (b) spelling (c) link (d) is attack. The real dataset cases were obtained from [12], whereas the synthetic dataset was based on [12] plus 1200 entries from human support-based cases from the Twitter. All the extra cases were not classified as attacks from a human expert. In both datasets the corpus has been converted to embedding representation and it was associated with its classification labe with four entries each. Three labels for the respective data and one with a yes or no (1 or 0) value of a conversation being a social engineering attack or not. To both datasets a small number of links to websites have been added, with some being malicious.

The accuracy metric has been used for the evaluating the classification models. The metric calculates the fraction of the prediction that each model got right. Equation 10 represents the accuracy where TP stands for true positives, TN for true negatives, FP for false positives and FN for false negatives.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

The results of the evaluation as shown in tables 1, 2, 3, 4, 5, 6, and 7 are based on the standard and compound datasets respectively and a 5-fold cross-validation approach has been used. In tables 1 to 6 the MAX and MIN values represent the maximum and minimum values of the 5-fold returned after each time the algorithm ran and the MEAN is the median value. Each algorithm was executed 10 times and at the end of each table the value is the mean value of 10 iterations for each of the three table labels. Table 7 however, presents the average mean results for the standard and compound datasets respectively.

Two algorithms from the SciKit and one from Keras libraries have been used for comparison purposes: Decision Tree, Random Forest and Convolutional Neural Network.

Table 2. Decision Tree results for the standard dataset **Table 2.** Decision Tree results for the compound dataset

MEAN	MAX	MIN	MEAN	MAX	MIN
0.736231884	0.826086957	0.583333333	0.908050847	0.916666667	0.9
0.692885375	0.782608696	0.625	0.921481816	0.957983193	0.900826446
0.68442029	0.833333333	0.458333333	0.921386555	0.941176471	0.890756303
0.650362319	0.695652174	0.583333333	0.923023127	0.95	0.890756303
0.656347826	0.76	0.608695652	0.921470588	0.957983193	0.9
0.659914361	0.826086957	0.5	0.923120703	0.941176471	0.909090909
0.694762846	0.826086957	0.416666667	0.909774011	0.93220339	0.891666667
0.647463768	0.791666667	0.5	0.919747899	0.932773109	0.908333333
0.65	0.695652174	0.583333333	0.913009121	0.925619835	0.899159664
0.741798419	0.833333333	0.652173913	0.914705882	0.933333333	0.890756303
0.681	**0.787**	**0.551**	**0.918**	**0.939**	**0.898**

Table 3. Random Forest results for the standard dataset **Table 4.** Random Forest results for the compound dataset

MEAN	MAX	MIN	MEAN	MAX	MIN
0.683333333	0.791666667	0.608695652	0.902910619	0.95	0.857142857
0.631521739	0.708333333	0.565217391	0.919787138	0.941176471	0.900826446
0.691304348	0.791666667	0.608695652	0.909703336	0.924369748	0.899159664
0.657608696	0.75	0.565217391	0.92640056	0.941666667	0.915966387
0.651811594	0.782608696	0.333333333	0.921414566	0.95	0.883333333
0.700724638	0.75	0.666666667	0.919803922	0.941176471	0.891666667
0.647826087	0.75	0.5	0.916468927	0.940677966	0.883333333
0.736067194	0.782608696	0.708333333	0.911358543	0.933333333	0.883333333
0.699130435	0.8	0.608695652	0.931468273	0.941666667	0.917355372
0.72826087	0.826086957	0.583333333	0.911372549	0.916666667	0.907563025
0.683	**0.773**	**0.575**	**0.917**	**0.938**	**0.894**

Table 5. CNN results for the standard dataset

MEAN	MAX	MIN
0.69619565	0.77083333	0.63043478
0.68768116	0.76721015	0.60869565
0.68217391	0.79583333	0.58695652
0.64891304	0.76630435	0.52173913
0.70227273	0.80797101	0.47916667
0.67579051	0.74456522	0.60416667
0.69963768	0.78804348	0.5326087
0.69794664	0.76086957	0.63416667
0.69043478	0.81170962	0.56521739
0.68913044	0.76086957	0.58333333
0.687	**0.776**	**0.5745**

Table 6. CNN results for the compound dataset

MEAN	MAX	MIN
0.90693073	0.93474333	0.87998143
0.92208448	0.95098983	0.90223645
0.91699495	0.93418311	0.89636798
0.92616184	0.94724333	0.90477135
0.92289258	0.9554016	0.89307667
0.92291231	0.94258647	0.90178879
0.91457147	0.93785068	0.88891
0.91700322	0.93446322	0.89724333
0.9236887	0.93505325	0.90966752
0.91448922	0.92641	0.90056966
0.91895	**0.93991**	**0.89741**

Table. 7. Comparison of algorithms for the standard and the compound datasets

ALGORITHM	RESULT	ALGORITHM	RESULT
Decision Tree	0.681	Decision Tree	0.918
Random Forest	0.683	Random Forest	0.917
CNN	**0.687**	CNN	**0.917**

Tables 1-7 present our results from a CB-Ranker with Deep Neural networks as applied to SE dataset classifications. The results seem promising for an application that has been conducted for the first time and confident they the proposed approach is able to identify possible SE attacks.

5 Conclusions

This paper presents a novel approach for SE attack detection in a web environment based on a combination of deep learning and case-based reasoning working at complementary levels of understanding natural language processing. Our approach processes dialogues, comprising our core for further classification purposes. We have presented an evaluation using both real and a semi-synthetic human dialog conversation whilst getting very high accuracy in social engineering attack detection. We have presented several baseline classification methods for comparison to show the effectiveness of our method. This work presents results of high accuracy; however, we believe that there is still room for improvement. In future work we will investigate adding more features to the dataset as well as applying more sophisticated deep neural network architectures for the classification process. We do aim to improve further the software performance as well as investigate the opportunity to use it at real time as a cybersecurity classification toolkit.

References

1. 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. IEEE, 2016.
2. Peng, T., Harris, I., Sawa, Y. (2018) "Detecting phishing attacks using natural language processing and machine learning." 2018 IEEE 12th International Conference on Semantic Computing (ICSC). IEEE, 2018.
3. 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, no. 2 (2018): 133-148.
4. 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, p. 53. ACM, 2018.
5. Cialdini, R. B. "The science of persuasion." Scientific American 284, no. 2 (2001): 76-81.
6. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D. (2014) "The Stanford CoreNLP natural language processing toolkit." In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, pp. 55-60. 2014.
7. Hoeschele, M., Rogers, M. (2005) "Detecting social engineering." IFIP International Conference on Digital Forensics. Springer, Boston, MA, 2005.
8. Hoeschele, M. (2006) "CERIAS Tech Report 2006-15 DETECTING SOCIAL ENGINEERING." (2006).
9. Abid, J., Asif, K., Ghulam, Z., Nazir, M.K., Alam, S. M., 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. IEEE, 2018.
10. Bhakta, R, Harris, I.G. (2015) "Semantic analysis of dialogs to detect social engineering attacks." Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015). IEEE, 2015.
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 & Security 76 (2018): 101-127.
12. Bezuidenhout, M., Mouton, F., Venter, H.S. (2010) "Social engineering attack detection model: Seadm." 2010 Information Security for South Africa. IEEE, 2010.
13. Mouton, F., Leenen, L., Venter, H. S. (2015) "Social engineering attack detection model: Seadmv2." 2015 International Conference on Cyberworlds (CW). IEEE
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." Thirteenth Symposium on Usable Privacy and Security ({SOUPS} 2017)
15. Krombholz, K., Hobel, H., Huber, M., Weippl, E. (2015). "Advanced social engineering attacks." Journal of Information Security and applications 22 (2015): 113-122.
16. Mouton, F., Leenen, Λ., Venter, H. S. (2016) "Social engineering attack examples, templates and scenarios." Computers & Security 59 (2016): 186-209.
17. Thanh, N. N., Nguyen, V. D., Hwang, D. (2017) "An influence analysis of the number of members on the quality of knowledge in a collective." Journal of Intelligent & Fuzzy Systems 32.2 pp. 1217-1228.

18. Javad, S., Moallem, P., Koofigar, H. (2017) "Training echo state neural network using harmony search algorithm." Int. J. Artif. Intell 15.1 (2017): 163-179.
19. Precup, R. E., and Radu-Codrut D. (2019) Nature-Inspired Optimization Algorithms for Fuzzy Controlled Servo Systems. Butterworth-Heinemann
20. Bilal H., Abedalguni. (2019) “Island-based Cuckoo Search with Highly Disruptive Polynomial Mutation”. Int. J. Artif. Intell 17.1 (2019): 57-82
21. Marchal, S., Franc Ãßois, J. State, R. , Engel, T. (2014), "Phishstorm: Detecting phishing with streaming analytics," IEEE Transactions on Network and Service Management, vol. 11, no. 4, pp. 458-471