HaploPhysh: Phishing URL Detection using Deep Learning

Mohammad Jafar Mashhadi
University of Calgary

Phishing is a type of online scam aiming to steal private sensitive information from the victims. Adversaries create fake copies of their target web page (e.g. online banking log-in page) and trick their victims into trusting the page. An important property of the page that can be used to assess its trustworthiness is its URL. However, there are many ways to make small alterations to a URL to come up with a new one, so creating black lists or a rule-based method may not be the most effective ways to defend against it. The limitations include: The URL should be an exact match, the list needs to be kept up to date, the clients must keep up to date with it, also there will be an inevitable delay between the first time someone falls into a phishing trap and the URL being added to the lists and get to the users. In this project we proposed a machine learning based solution called HaploPhysh[1] that can detect a completely new phishing web page from its URL. We trained several models on a data set made by combining several existing blacklists. The results show that this is a promising method for automatically detecting malicious web pages. There were three research questions we sought answers to:

RQ 1) Is it feasible to detect phishing scams from their URLs alone? The data set properties affect whether a model is trainable. Ours was different from the ones used in the literature, so answering to this question for *our data* is crucial before moving forward. 30% of the data was set aside for validation and the remaining 70% were used for training. All models were optimized using Adam optimizer [2] on a MSE loss function. **Answer:** It works. The data we collected was large enough to train a model that performs greatly in this classification task. The adversaries leave footprints in the URL making it detectable by machine learning algorithms.

RQ 2) Can a better performing deep learning architecture be found and how will this model compare to the previous work? There are great architectures proposed in prior work such as URLNet that adapted different ideas in deep learning literature. There are more ideas that they have not tried and can improve the model even more. Recurrent layers are one of them. To answer this research question we tried a few new model architectures to see how much improvement they can bring into the game. **Answer**: Yes, we could do better. We created four models: 1. a convolutional model that uses character level embeddings, 2. one with word level embeddings, 3. a combination of both and also 4. one that uses a recurrent layer on top of number 3. We compared them based on TPR (Precision), recall, F1 score, FPR, accuracy, and AUC-ROC. The results can be seen in table 1. In figure 1 the ROC curves of our proposed architectures can be seen which indicate a pretty good discriminator performance. The performance of the word embedding model is slightly worse and that is not a surprise. There are far more words than there are characters, typos will make new words. The word boundaries also cannot be found with RegExs so a string like 'BankOfNovaScotia' will be considered one single word. This variety of words along with the limit on vocabulary size (that will cause a lot of OOV-Out of Vocabulary- tokens) makes

[1]https://github.com/MJafarMashhadi/Haplophysh

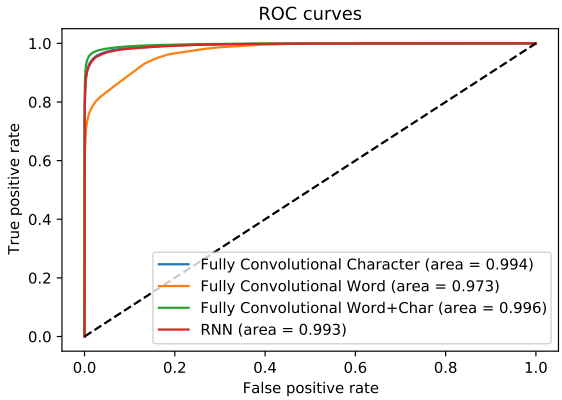

Figure 1: ROC curves of our proposed models

it harder to draw conclusions about the URL. Actually the fact that the performance is only *slightly* worse is more surprising. The combi model (3) outperforms both individual methods which lives up to the expectations. Although a larger model has more learning capacity and thus more prone to memorization and over-fitting, if that is managed the performance is expected to get better. Imagine an extreme case when one branch of the model (say, the word embedding branch) couldn't provide any more information than the other one; in that case right after the concatenation layer where the branches merge the model learns to assign zero or very small weights to that branch, effectively eliminating it. So it is unlikely to have a combi model that has worse performance than both its parts. Here, the performance is quite close to the better performing one (char level), not only close but a tad bit better. In URLNet paper the TPRs are reported for fixed FPRs. The best TPR numbers were related to the highest reported FPR in their paper, 0.1. So even though the TPR by itself is impressive, it costs a higher false positive rate to achieve. RQ 3) Does training on an extended data set containing more malicious URLs (not limited to phishing) result in a better performing model? There are data sets available that contain other malicious URLs such as spams and malware links. Since they try to deceive user they might share features with phishing URLs. We wanted to see if this holds for them too and also if the model can be trained as (or more) effectively. **Answer:** Yes, it gets better! We retrained the best architecture with the new data to see if the results get better. (see table 1) We can conclude that other malicious URLs too have latent features a deep neural network can discover and use to detect them.

REFERENCES

[1] Birhanu Eshete, Adolfo Villafiorita, and Komminist Weldemariam. 2013. BINSPECT: Holistic analysis and detection of malicious web pages. In *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, Vol. 106 LNICS. 149–166. https://doi.org/10.1007/978-3-642-36883-7_10

Table 1: Metrics for all of our proposed models trained on the Phishing data set versus the best model being trained on a extended data set containing malware and spam URLs as well accompanied with the metrics for the previous work

Model	TPR	FPR	Accuracy	AUC-ROC	Recall	F1 Score
Character Embedding + Conv.	95.8444%	2.7622%	96.8747%	0.994	92.4401%	94.1115%
Word Embedding + Conv.	92.5722%	12.7404%	88.6439%	0.973	71.9143%	80.9461%
Word+Char Embedding + Conv.	96.7278%	1.7222%	97.8739%	0.996	95.1905%	95.9530%
Word+Char Embedding + RNN	95.3940%	2.6263%	96.8579%	0.993	92.7536%	94.0553%
RQ2, Related Work						
URLNet - Char [3]	95.46%	10%	N/A	0.9892	N/A	N/A
URLNet - Word	92.97%	10%	N/A	0.9842	N/A	N/A
URLNet - Full	97.22%	10%	N/A	0.9929	N/A	N/A
BISNPECT[1]	N/A	18.9%	97.81%	N/A	N/A	N/A
eXpose[4]	92%	1%	N/A	0.993	N/A	N/A
RQ3, retraining with the extended data						
Word+Char Embedding + Conv	97.1775%	1.7503%	97.9484%	0.997	95.5946%	96.3795%

[2] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[3] Hung Le, Quang Pham, Doyen Sahoo, and Steven C H Hoi. 2018. *URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection.* Technical Report. arXiv:1802.03162v2 https://doi.org/10.475/123{_}4

[4] Joshua Saxe and Konstantin Berlin. 2017. eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys. (2017). arXiv:1702.08568 http://arxiv.org/abs/1702.08568