Triplet Mining-based Phishing Webpage Detection

Kalana Abeywardena,*† Jiawei Zhao,* Lexi Brent,* Suranga Seneviratne,* Ralph Holz*‡

* University of Sydney, Australia, † University of Moratuwa, Sri Lanka, ‡ University of Twente, Netherlands
Email: {kalana.abeywardena, jzha4444, suranga.seneviratne}@sydney.edu.au, r.holz@utwente.nl}

***Abstract*—Phishing web pages impersonate legitimate websites to trick users into entering sensitive information such as their credentials. In many high profile data breaches, the initial entry points have been traced back to phishing attacks. Attackers are using increasingly sophisticated methods such as code obfuscation to bypass existing phishing detection systems. Since phishing websites show very high visual similarity to the respective target pages, recent advances in Convolutional Neural Networks (CNN) can be leveraged to build better phishing detection systems. In this work, we propose a novel CNN architecture consisting of two paths to capture the *content similarity* and *structural similarity* between web pages. Leveraging the fact that web pages of the same web site are visually similar, we use *triplet learning* to train our model without any labelled phishing examples.**

I. Introduction

Despite years of efforts, phishing remains a critical and often successful attack vector. Phishers lure victims onto crafted websites that give the impression of legitimacy, usually to obtain valuable information like credentials or financial data. Typically, phishing sites imitate one or more aspects of a legitimate site such as URL, content, layout, colors, or overall branding. Many well known recent data breaches such as Facebook and Google invoicing scams [8] and the DNC hack [4] were traced back to initial entries through phishing.

In a recent report [15], Trend Micro reports a rising number of phishing attempts: 2.4m attempts to obtain credentials in the first six months of 2019 alone. In their incident report of August 2019, the UK's National Cyber Security Centre calls phishing the 'most prevalent attack delivery method seen over the last few years' [16]. As such, it is highly important to come up with techniques to close this attack vector.

Methods to fight phishing include blacklisting or whitelisting as well as classification mechanisms that rely on text or image features, implemented with machine learning. Blacklisting and whitelisting benefits from the manual identification of known phishing sites. Nonetheless, both scale poorly and also require great diligence in creating and updating the respective lists. Automated text content-based and URL-based methods do not yet achieve the required accuracy and robustness.

We propose a novel automated phishing detection framework based on recent advances in Convolutional Neural Networks that focuses on the visual similarity between the phishing page and its target. The specific phishing detection scenario we consider is that, given a screenshot of a web page, we want to decide whether it is a possible phishing attempt to a well known website for which we have stored the screenshots. Many organizations indeed know what pages are usually being targeted (e.g., Office 365) and there are lists of common targets [14]. Thus, it is possible to maintain a database of target web pages and their screenshots and new web pages can be matched against them.

In contrast to existing work, the CNN architecture we propose can measure the *content similarity (e.g. colour and texture)* and the *structural similarity (e.g. arrangement of headers and footers)* between web page screenshots. Since our method does not rely on text and code features, it is resilient to *code obfuscation*. We make the following contributions.

- We propose a novel CNN architecture that is able to capture both content and structural similarity between images, in contrast to traditional image matching methods.
- We leverage the fact that pages of the same website show high visual similarity to each other and use triplet learning to train our model without any phishing samples.
- We show that triplet learning boosts phishing detection by 20%–30% and *structural similarity* results in further gains of 5%–8%.

II. Related work

There is a plethora of work on detection of phishing websites. The most commonly used method, *blacklisting*, has limitations with respect to the short lifetime of attacks and *zero hour phishing attacks* [13]. Some work looked into identifying phishing attacks based on a combination of textual features extracted from `HTML` and `CSS` files and network features such as IP addresses and DNS information [2], [11]. Such methods are generally susceptible to code obfuscation [5].

Fu et al. [5] propose measuring the visual similarity between web pages using Earth Mover's Distance (EMD). First, web page screenshots are converted to a low resolution and represented by a color vector and the centroid of its position in a degraded color space. Then, EMD is used to calculate the distance between the web pages. If the distance is less than a threshold, it is considered a phishing attempt. Medvet et al. [10] propose a similar approach. Wang et al. [17] and Zhou et al. [20] use SIFT and SURF features to extract logos from web pages and use that to decide whether they are impersonating popular pages.

However, all of the above use methods that are either only suitable for identifying very similar images (e.g., perceptual hashing) or content-based image retrieval methods (e.g. SIFT and SURF) that are more suitable for scene comparison. In contrast, we propose to use triplet learning to measure both content and structural similarity of web pages.

(a) web-page (b) ground-truth mask

Fig. 1. An example of the annotated dataset

Recently, Abdelnabi et al. [1] proposed *WhiteNet*, a phishing detection method that uses a triplet CNN. The key idea was to use phishing websites and their targets to train the triplet network. In contrast, our method does not require phishing samples during training, which is a significant advantage when it comes to zero-hour phishing attack detection. Instead, we leverage the fact that pages of the *same* website are visually similar, which is useful for triplet training. We also propose a pre-training strategy to further improve the model training.

III. Data Collection

We build a dataset from the Top 10k domains of the Alexa list of 2019-01-25. The rationale is that phishers usually target popular sites. We use Selenium[1] to instrument a headless Chrome browser. We crawled each domain on HTTP, following redirects to HTTPS. We set a page load timeout of 60 seconds; after completion of a page load, we waited five seconds before taking a screenshot.

For each domain, we take screenshots of several pages as PNGs of dimension $1920 \times 1080 \times 3$. We first take a screenshot of the landing page. We then search the HTTP(S) links on the landing page for indications of login pages by applying a case-insensitive regular expression (looking for strings such as *login* and *log-in*) to both the *href* attribute of HTML anchor elements ($<a>$) as well as their value. We screenshot the first such page we can identify. With this method, our crawler identifies login pages for 38% of domains.

Finally, we add up to five more screenshots by following random links from the landing page, as long as they link to the same domain. In total, we collect *49,063 screenshots* belonging to *9,557 domains* and *3,619 login pages*. We resize the screenshots to $224 \times 224 \times 3$ so that they can be fed into a CNN. We next describe different subsets we use.

Training dataset: From the collected data, we randomly select *44,325 screenshots* of *7,494 domains* as our training set.

Validation dataset: We randomly select *118 screenshots* of *20 domains* (not in the training set) as our validation set.

Annotated pages: We select 370 screenshots from the training set and manually create the ground truth structural masks to pre-train our model. We show an example in Figure 1.

Phishing set: We use PhishTank[2] to obtain ground truth data on phishing sites. We crawl the landing page of each site reported on Phishtank on 2019-02-01 if it was also in our Alexa Top10K list. We then *manually* verified all sites and marked true positives. After removing duplicates, we obtain *113 phishing screenshots* belonging to *22 domains*. We split this dataset into: ***Query-threshold dataset*** of *47 phishing screenshots* of 22 domains and ***Query-web dataset*** of *66 phishing screenshots* of 22 domains.

Target scope: To evaluate our model as well as to compare with baselines in Section IV-E, we collect *232 new screenshots* of *36 domains* on the Alexa Top 10K list on 2019-10-07.

[1]www.seleniumhq.org
[2]www.phishtank.com

IV. Methodology

We consider the phishing website detection problem as an image matching problem and propose a novel CNN architecture based on three key ideas:

i) Image embeddings from a ResNet: Published work highlighted that embeddings generated by CNNs can be used for content based image matching [3], [19]. Thus, we simply use the embeddings of a ResNet as one similarity dimension.

ii) Feature Pyramid Network for structural similarity: The Feature Pyramid Network (FPN) is a CNN model proposed for object detection and image segmentation [9] that outputs bounding boxes for structural components and segmentation masks for a given input. The web pages of a given domain have a specific structure and therefore image segmentation helps to identify the structural components of a given web page screenshot that can be used to measure structural similarity.

iii) Triplet learning: Triplet learning that was successfully used in face recognition [12] uses the *triplet loss* to compare an *anchor image* with a *positive image* (which is similar to the anchor) and a *negative image* (which is different from the anchor) so that the CNN learns to compare images.

A. The CNN Architecture

Figure 2 shows our CNN architecture, which has two main paths. The upper path contains a triplet network based on the ResNet that focuses on measuring *content similarity*. The bottom path contains a triplet network based on FPN that focuses on *structural similarity*. At training time, we feed the network with triplets obtained from our training set according to the triplet mining strategy we describe later in Section IV-C.

Content embeddings: We load the model with the weights of ResNet50 and use the content embeddings of the triplets to calculate the *positive and negative embedding distances* $d^{(i)}_{pos,c}$ and $d^{(i)}_{neg,c}$ of the i^{th} triplet:

$$d^{(i)}_{pos,c} = ||anc^{(i)}_c - pos^{(i)}_c||^2_2 \quad (1)$$

$$d^{(i)}_{neg,c} = ||anc^{(i)}_c - neg^{(i)}_c||^2_2 \quad (2)$$

Structural embedding: We use a ResNet50 FPN as the feature extractor and pre-train it using our annotated dataset.

The FPN network outputs 11 feature maps. That output is followed by a MaxPooling layer of $stride = 1$ and $pool\_size = (2, 2)$. To derive the final segmented map of size

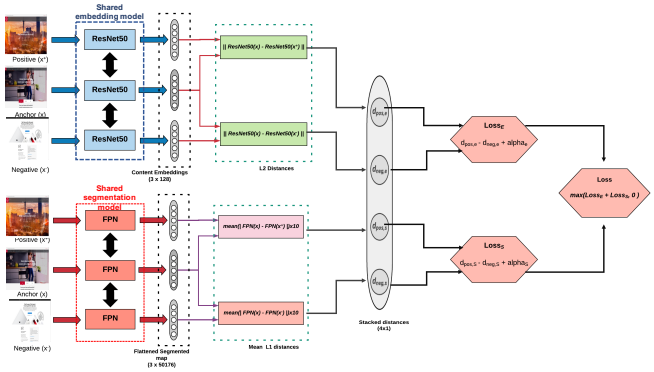

Fig. 2. CNN Architecture

224×224, we feed the output from the MaxPooling Layer to a Conv2D layer of $kernel\ size = (1, 1)$. The output is flattened to obtain the structural embedding $S \in \mathbb{R}^{50,176}$. We calculate the *positive and negative segmentation distances*, $d^{(i)}_{pos,s}$ and $d^{(i)}_{neg,s}$ of the i^{th} triplet as follows.

$$d^{(i)}_{pos,s} = \frac{\theta}{dim(anc^{(i)}_s)} \sum_{j=0}^{dim(anc^{(i)}_s)-1} (|anc^{(i,j)}_s - pos^{(i,j)}_s|) \quad (3)$$

$$d^{(i)}_{neg,s} = \frac{\theta}{dim(anc^{(i)}_s)} \sum_{j=0}^{dim(anc_c s^{(i)})-1} (|anc^{(i,j)}_s - neg^{(i,j)}_s|) \quad (4)$$

Since the output of this model branch is a flattened array of the segmented map, we use pixel-wise distance and take the mean of the summation of absolute pixel-wise difference between the structural embeddings. A scaling factor $\theta = 10$ is used as the mean value is too low to affect the final loss.

B. Overall Loss Function

Our model is similar to TripletNet [7] with two branches. Thus, we use a modified triplet loss as the final loss.

$$loss = \sum_i max(loss^{(i)}_c + loss^{(i)}_s, 0) \quad (5)$$

where

$$loss^{(i)}_c = d^{(i)}_{pos,c} + \alpha_c - d^{(i)}_{neg,c} \quad (6)$$

and

$$loss^{(i)}_s = d^{(i)}_{pos,s} + \alpha_s - d^{(i)}_{neg,s} \quad (7)$$

$loss^{(i)}_c$ is the content loss and $loss^{(i)}_s$ is the structural loss of the i^{th} triplet. For content margin, α_c, and structural margin, α_s, we use the value 0.4, similar as in the original TripletNet.

C. Triplet Learning

During training, we generate triplets at the beginning of each epoch to be fed into the training model from each batch. Initially, we use *easy triplet mining* [6], [18], with $anchor$ and $positive$ being two screenshots of the same domain directory, while $negatives$ are screenshots from other domain directories that are randomly chosen. In the second stage of the training, we change the triplet mining strategy to *semi-hard mining* [6], [18], where hard negatives are considered when generating the triplets. By changing the mining strategy, we select hard negative web-pages which are visually closer to the anchor web page. By doing so, the model learns to push the hardest negatives away from the positives so that it can differentiate web pages from the same domain from the rest of the screenshots.

D. Evaluation Metrics

During the evaluation, we use the validation set. We obtain the $C \in \mathbb{R}^{128}$ and $S \in \mathbb{R}^{50,176}$ for each image in the validation set by forward passing. Then, for an image in each domain of the 20 domains of the validation set, we retrieve k nearest neighbours and decide whether there is a match based on an empirically obtained threshold.

We evaluate the performance of our model using the precision@k and recall@k for $k = 1$ and 6. We did not try higher k values as, on average, we have only six images per directory in the validation set. We retrieve the similarity scores that are below the threshold for different similarity comparison methods and obtain the closest k similarity scores.

E. Baselines

We compare the performance of our method with a number of other image matching methods such as raw pixel-wise distance, hashing methods, feature based methods (SIFT and SURF), and traditional segmentation.

We also consider the components of our model separately to generate the embeddings. For example, we evaluate embeddings from pre-trained ResNet50 and pre-trained ResNet50 fine-tuned with triplet learning using our training dataset (TripletNet), and pre-trained FPN fine-tuned using the annotated web-pages dataset. For each scenario, we identify the decision threshold using the **Query-Threshold** dataset.

F. Training the Model

We train our model using a two GPU setup for 500 epochs with Adam optimizer and a static learning rate of 0.001. We create five batches of screenshots of size 8,192 from the training set. For the first stage of training, for each batch of screenshots, we generate 8,192 easy-triplets totalling 40,960 triplets. For the second stage of training, using the model trained on easy triplets, we generate 4,000 triplets per batch, resulting in a total of 20,000 triplets. We change the triplet mining method to semi-hard mining at the 400^{th} epoch and continue to train the model to 500 epochs.

V. Results

We evaluate the performance of our method in comparison to other baselines using the **Target Scope** and **Query web** datasets. For each method, we derive the similarity scores that are below the empirically obtained threshold for each query image. If no such similarity scores are detected, we conclude that the query page is benign. We show the results in Table I.

TABLE I
PERFORMANCE ON THE TARGET SCOPE DATASET

	k	Pixel L2	Avg. hash	Diff. hash	Perc. hash	Wavelet hash	SIFT	SURF	Segment.	FPN	ResNet	TripletNet	Our method
Precision	1	0.3056	0.4500	0.6341	0.4000	0.3056	0.3478	0.4419	0.1220	0.5349	0.5000	0.7111	**0.7955**
	6	0.1698	0.3121	0.6610	0.1560	0.1185	0.1813	0.1552	0.0830	0.2806	0.3153	**0.6720**	0.6477
Recall	1	0.1667	0.2727	0.3939	0.1667	0.2424	0.2424	0.2879	0.0758	0.3485	0.2879	0.4848	**0.5303**
	6	0.0758	0.1247	0.1801	0.0577	0.0508	0.0808	0.0624	0.0439	0.1270	0.0152	0.1940	**0.3756**

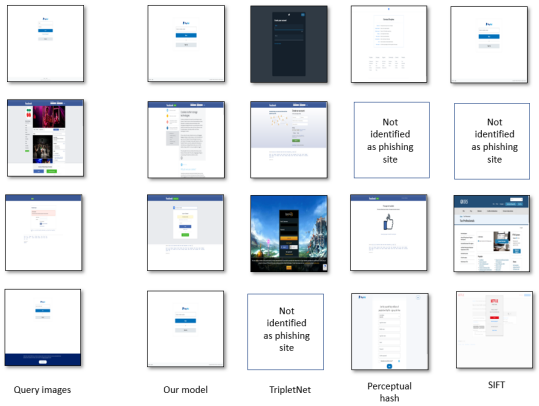

Fig. 3. Target detection comparison

Our results show that the content embeddings derived from the TripletNet can predict 71% of the targets that are identified as possible phishing sites. We identify the target correctly for 48.5% of the the query images, which is the highest value among all the methods. Use of both content and structural embeddings increases the precision and recall values by 8.44% and 4.55% respectively, indicating the importance of combined embeddings.

It is notable that methods such as SIFT and perceptual hashing do not perform well in our comparison. Traditional feature extractors like SIFT and SURF tend to identify blobs and corners in letters and symbols in the web pages as key features. Thus, when there is less content on a page, they are successful. But with more content on a web page, they tend to identify the wrong target. In Figure 3, we show some sample queries and the results we obtain for several methods.

VI. CONCLUSION

In this paper, we proposed a novel CNN architecture for phishing web page detection. In one branch of our CNN, we measure the *content similarity* between web pages using a ResNet. In the other branch, we measure the *structural similarity* between web pages using an FPN. By leveraging the fact that web pages of the same web site are usually visually similar and combining that with triplet learning, we can train our model without the need for any phishing samples. We compared the performance of our method with baseline methods and showed that use of triplet learning helps to improve the precision and recall in phishing detection by approximately 20%–30%. Our results also show that adding *structural similarity* as another similarity dimension gives further improvements in the range of 5%–8%.

ACKNOWLEDGEMENT

This project is partially funded by the NSW Cyber Security Network's Pilot Grant Program 2018.

REFERENCES

[1] S. Abdelnabi, K. Krombholz, and M. Fritz, "Whitenet: Phishing website detection by visual whitelists," 2019.

[2] S. Afroz and R. Greenstadt, "Phishzoo: Detecting phishing websites by looking at them," in *IEEE 5th International Conference on Semantic Computing*, 2011.

[3] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *ECCV*. Springer, 2014.

[4] P. Bump, "Timeline: How Russian agents allegedly hacked the DNC and Clinton's campaign," https://www.washingtonpost.com /news/politics/wp/2018/07/13/timeline-how-russian-agents-allegedly-hacked-the-dnc-and-clintons-campaign/.

[5] A. Y. Fu, L. Wenyin, and X. Deng, "Detecting phishing web pages with visual similarity assessment based on earth mover's distance," *IEEE Transactions on Dependable and Secure Computing*, 2006.

[6] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[7] E. Hoffer and N. Ailon, "Deep metric learning using triplet network." in *ICLR (Workshop)*, Y. Bengio and Y. LeCun, Eds., 2015.

[8] T. Huddleston, "How this scammer used phishing emails to steal over $100 million from Google and Facebook," https://www.cnbc.com /2019/03/27/phishing-email-scam-stole-100-million-from-facebook-and-google.html, 2019.

[9] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[10] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proceedings of the 4th international conference on Security and privacy in communication networks*. ACM, 2008, p. 22.

[11] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "Phishnet: Predictive blacklisting to detect phishing attacks," in *IEEE INFOCOM*, 2010.

[12] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *IEEE CVPR*, 2015.

[13] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in *Sixth Conference on Email and Anti-Spam*, 2009.

[14] SonicWall, "Inside the modern phishing campaigns," 2019. [Online]. Available: https://blog.sonicwall.com/en-us/2019/05/inside-the-modern-phishing-campaigns-of-2019/9

[15] Trend Micro, "The rising tide of credential phishing," https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/the-rising-tide-of-credential-phishing, 2019.

[16] UK Cyber Security Centre, "Incident trends report," https://www.ncsc.gov.uk/report/incident-trends-report, 2019.

[17] G. Wang, H. Liu, S. Becerra, K. Wang, S. J. Belongie, H. Shacham, and S. Savage, *Verilogo: Proactive phishing detection via logo recognition*. University of California, 2011.

[18] H. Xuan, A. Stylianou, and R. Pless, "Improved embeddings with easy positive triplet mining," *CoRR*, vol. abs/1904.04370, 2019. [Online]. Available: http://arxiv.org/abs/1904.04370

[19] J. Yue-Hei Ng, F. Yang, and L. S. Davis, "Exploiting local features from deep networks for image retrieval," in *CVPR workshops*, 2015.

[20] Y. Zhou, Y. Zhang, J. Xiao, Y. Wang, and W. Lin, "Visual similarity based anti-phishing with the combination of local and global features," in *13th International Conference on Trust, Security and Privacy in Computing and Communications*, 2014.