Personalized, Browser-based Visual Phishing Detection Based on Deep Learning

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao

Department of Engineering and Architecture, University of Trieste, Trieste, Italy
{bartoli.alberto, andrea.delorenzo, emedvet, ftarlao}@units.it

Abstract. Phishing defense mechanisms that are close to browsers and that do not rely on any forms of website reputation may be a powerful tool for combating phishing campaigns that are increasingly more targeted and last for increasingly shorter life spans. Browser-based phishing detectors that are specialized for a *user-selected* set of targeted web sites and that are based *only* on the overall *visual appearance* of a target could be a very effective tool in this respect. Approaches of this kind have not been very successful for several reasons, including the difficulty of coping with the large set of genuine pages encountered in normal browser usage without flooding the user with false positives. In this work we intend to investigate whether the power of modern deep learning methodologies for image classification may enable solutions that are more practical and effective. Our experimental assessment of a convolutional neural network resulted in very high classification accuracy for targeted sets of 15 websites (the largest size that we analyzed) even when immersed in a set of login pages taken from 100 websites.

1 Introduction

Phishing campaigns are increasingly more targeted to specific and small population of users and last for increasingly shorter life spans [1, 4]. There is thus an urgent need for defense mechanisms that are close to browsers and that do not rely on any forms of blacklisting/URL-based reputation: there is simply no time for detecting novel phishing campaigns and notify all interested users quickly enough.

In this work we investigate the feasibility of browser-based phishing detectors that: (1) are specialized for a *user-selected* set of websites (i.e., we do not insist on detecting phishing attacks directed at any possible target); (2) are based *only* on the overall *visual appearance* of a website (i.e., without relying on any URL-related feature, blacklisting, peculiar features of a given screenshot); (3) allow incorporating a specific website in the set automatically (i.e., with a *systematic* and *website-independent* procedure based solely on screenshots); (4) are simple and fast enough to warn the user in real time.

There have been a number of proposals for attempting to detect phishing pages based solely on their visual features (e.g., [5, 2, 3]). The framework is based on an image classifier equipped with prior knowledge of the legitimate

⟨protocol, domainName⟩ pair(s) of each website of interest to the user. When the browser has loaded a web page p, the classifier determines whether the screenshot of p belongs to one of the visual classes corresponding to each website to be protected. In case of a match, the tool compares the actual ⟨protocol, domainName⟩ of p to those expected for that website and warns the user in case of a mismatch.

Key advantage of this framework is that it does not require any form of blacklisting or of URL-based reputation. The required tool could be implemented as a browser extension and could possibly be integrated within a password manager. The resulting defensive mechanism would implement the procedure that any technically-savvy and constantly vigilant user applies in practice, except that in this case the procedure would be automated and thus available to every user and continuously. The resulting scenario would thus raise the bar for attackers considerably.

It is fair to claim that approaches of this kind have not been very successful so far, though, the key reasons include the difficulty of actually implementing the above requirements while providing sufficiently high detection accuracy [6]. For example, we are not aware of any actual implementation of the image classifier devised in [5], the screenshot classifiers analyzed in [2] did not deliver adequate accuracy, the large scale classification experiment in [3] considered 16 targeted websites but without injecting pages from other websites (unlike what happens during normal browser usage). In this work we intend to revisit the framework and investigate whether modern deep learning methodologies for image classification may lead to solutions that are practical and effective.

2 Problem statement and proposed approach

The login page of a website may have several different appearances, depending on the user agent declared by the browser (e.g., desktop vs. mobile) and on the resolution of the page rendered by the browser. We say that a screenshot x *looks similar* to the screenshot $x(p)$ of a login page p, denoted $x \sim x(p)$, if the visual appearance of the two screenshots is sufficiently similar to let the user believe that x is indeed the rendering of a login page p. In this work we consider that two screenshots satisfy this definition only if they are screenshots taken from the same login page, possibly with different resolutions. This assumption corresponds to a scenario in which a phishing page is an exact visual replica of a genuine login page, hosted at a (fraudulent) website different from the genuine one. We will consider more general scenarios in future work.

We consider a statically defined list S of websites $s_1, s_2, \ldots, s_{n_S}$ that have a login page and that must be protected from phishing attempts (this set may be personalized on a user basis). The problem input consists of a screenshot x. The corresponding output y must be one of $n_S + 1$ categorical values (*classes*), as follows: if x looks similar to a login page of website $s_j \in S$, then $y = j$; otherwise, $y = n_S + 1$. In other words, the problem does not consist in discriminating between phishing pages and legitimate pages. The problem consists in associating a screenshot with a predefined set of visual classes, including a special

class meaning "none of the websites selected for protection" that is necessary in practice.

We explored a solution based on deep learning, specifically, on a neural network in which: the input layer corresponds to a screenshot with a 640×360 pixel resolution with 3 channels (RGB); the output layer consists of $n_S + 1$ neurons, with one-hot encoding of the corresponding classes. We apply a screenshot x to the input layer after the following preprocessing steps. First, we create an image $x^{16:9}$ with 16:9 aspect ratio by either cropping or extending (by wrapping) the bottom part of x. Then, we resize $x^{16:9}$ to 640×360 resolution with the bilinear interpolation of the Pillow Python module. The chosen resolution is high enough to capture small graphical details such as, e.g., logo and text characters shape. We chose a 16:9 aspect ratio because this is the most common screen-ratio for desktop computers (we intend to explore a single classifier for both desktop and mobile platforms in future work).

We used a *convolutional neural network* (CNN) composed of a sequence of four pairs $\langle \mathrm{CN}_i, \mathrm{MP}_i \rangle$, i.e., $\langle$convolutional layer, maxpool layer$\rangle$, with $i = 1, \ldots, 4$, as follows. CN_1 applies 32 kernels of size $5 \times 5 \times 3$, CN_2 64 kernels of size $5 \times 5 \times 32$, CN_3 96 kernels of size $5 \times 5 \times 64$, CN_4 128 kernels of size $5 \times 5 \times 64$. MP_1 applies 4×4 kernel and 4×4 stride, while the other maxpool layers apply a 2×2 kernel and 2×2 stride. The output of MP_4 is fed to a fully connected layer FC_1, that is followed by another fully connected layer FC_2, that is followed by the output layer SM. Both FC_1 and FC_2 have 200 neurons. The activation function for all the CN and the FC layers is ReLU, while the output layer implements a softmax. We implemented this network architecture with Keras.

For our experimental assessment we collected 1500 screenshots of login pages from 100 websites, 15 different screenshots from each website. For each website, we identified the login page and captured 15 different screenshots of that page differing on the browser windows size resolution. We captured the resolutions that correspond to the 15 most common screen sizes[1], on the grounds that different resolutions may result in very different webpage layouts. We selected 30 websites of the companies most targeted by phishing attacks, according to reports by specialized IT security firms, and 70 websites from the Alexa ranking of the most visited websites. We skipped duplicate websites, sites with pornographic content, sites without a login page. We also skipped websites whose login page was identical to an already collected login page of another website, due to the usage of single sign on.

We trained the network after a *data augmentation* procedure applied to the collected screenshots. This procedure may be used for obtaining a virtually unlimited amount of artificial screenshots different from the real ones but that should be effective for training the multiclass classifier. The procedure consists of the following steps, executed whenever an artificial screenshot x^a is to be obtained from a real screenshot x (all random quantities have uniform distribution in a specified interval): (i) modify x and obtain a 16:9 aspect ratio (as in

[1] `http://gs.statcounter.com/screen-resolution-stats`

preprocessing); (ii) circular shift vertically and horizontally of a random quantity; (iii) with 50% chance, apply a centered zoom of a random zoom factor and keep size unchanged by cropping; (iv) with 50% chance, either lighten or darken the image; the 3 RGB channels are all lighted or all darkened, with a random multiplicative factor different in each channel such that the overall change never exceed 30% of the original pixel value.

We executed the actual training of the network as follows. Let S denote the set of websites $s_1, s_2, \ldots, s_{n_S}$ whose login page has to be protected from phishing attacks. Let X_S denote the set of login page screenshots of websites in S. Let T, V denote the learning data to be obtained from X_S, i.e., the training set and the validation set respectively. Both T and V are sets of pairs $\langle x, y \rangle$, where x is a screenshot and $y \in \{1, \ldots, n_S + 1\}$ is the corresponding *class* (encoded as one-hot in the output layer of the network): if x is a login page of website $s_j \in S$, then $y = j$; otherwise, $y = n_S + 1$.

At each training epoch, we randomly select a subset of T such that 50% of the pairs are of class $y = n_S + 1$ while the remaining pairs are equally distributed across the other classes. We loop across this subset for constructing a set of artificial screenshots T_a with the data augmentation procedure described above and use T_a for training in the current epoch (the class of an artificial screenshot will be the same as the corresponding real screenshot). We group pairs of T_a in batches of size $b_s = 32$ and execute each epoch for $n_b = 12(n_S + 1)\frac{c_T}{b_s}$ steps, c_T being the median cardinality of classes in T (we use each element of T_a once, hence $|T_a| = 12(n_S+1)c_T$). We use Stochastic Gradient Descent (SGD) with the following parameters: momentum set to 0.9; learning rate 0.02; gradient clipping with maximum norm value 1.0; dropout with probability 0.1 after each CN layer and with probability 0.3 after each FC layer.

At the end of each training epoch we evaluate the classification accuracy of the current network on a set of validation pairs V_a (the same set for all epochs). We construct this set by randomly selecting a subset of V so that all classes have the same cardinality. We then loop across this subset for constructing a set of artificial screenshots V_a with the data augmentation procedure described above until $|V_a| = 24(n_S + 1)c_V$, c_V being the median cardinality of classes in V. We trained the network for 400 epochs and used the network with higher classification accuracy on V_a ever seen on all the epochs.

3 Experimental assessment

We assessed three different values for the number of websites to be protected $n_S = 5, 10, 15$, corresponding to $6, 11, 16$ classes respectively (we remark that the large scale classification experiment in [3] considered 16 targeted websites). We constructed the training set T, validation set V and testing set E so as to ensure that: (a) each of the $n_S + 1$ classes has the same cardinality in T and in V; (b) all the remaining data are used in E.

In detail, let S_D be the set of 100 websites of our dataset and let X_D be the corresponding set of 1500 screenshots. We denote each element of X_D by $\langle x, i \rangle$

where x is a screenshot taken from website $s_i \in S_D$. Let X_i denote the subset of X_D containing screenshots taken from $s_i \in S_D$. Initially, we set $T = V = E = X_o = \emptyset$; then:

1. We randomly selected a subset $S' \subset S_D$ such that $|S'| = n_S$.
2. For each website $s_i \in S'$, we randomly partitioned X_i in three subsets X_i^T, X_i^V, X_i^E with the same cardinality; then, we added X_i^T to T, X_i^V to V and X_i^E to E.
3. For each website $s_j \notin S'$, we added X_j to X_o.
4. We randomly partitioned X_o in three subsets X_o^T, X_o^V and X_o^E such that $|X_o^T| = |X_o^V| = |X_i^T|$ (thus, $|X_o^E| = |X_o \setminus (X_o^T \cup X_o^V)|$); then, we added X_o^T to T, X_o^V to V and X_o^E to E.
5. We adjusted all class labels so that elements from S' were of classes $1, 2, \ldots, n_S$ and elements from $S_D \setminus S'$ were of class $n_S + 1$.

We repeated the above procedure 3 times for each value of n_S, each time selecting a different subset S' of websites at step 1 and ensuring that the three subsets have empty intersection. Furthermore, for each selected subset S', we executed a 3-fold cross validation by rotating the roles of sets T, V, E. Thus, we executed 9 different experiments for each value of n_S. For each trained network we computed the performance indexes described below on the testing set E. The *Categorical Accuracy* (CA) is the ratio of correctly classified screenshots while the *Balanced Categorical Accuracy* (BCA) is the arithmetic mean of the accuracy in each class. The *Missed Alarm Ratio (MAR)* is the ratio of screenshots from websites in S' classified as belonging to class $n_S + 1$ (screenshots from websites that should be protected from phishing attacks, but are not recognized as belonging to those sites). The *False Alarm Ratio (FAR_I)* is the ratio of screenshots from websites in S' classified as belonging to a class different from the correct class and different from $n_S + 1$ (screenshots from websites that should be protected and that are recognized as a login page, but are attributed to a website different from the real one). The *False Alarm Ratio on other web sites (FAR_U)* is ratio of screenshots from websites not in S' classified as belonging to a class different from $n_S + 1$ (screenshots from websites for which a protection from phishing attacks has not been required, but are attributed to a website that should be protected).

Table 1 shows the indexes values, averaged across the 9 experiments. Column ET reports the average execution time for each experiment (on a machine with 18 cores, 128 GB RAM, Xeon(R) E5-2697 v4 @ 2.30 GHz).

4 Discussion and concluding remarks

We believe the results are highly encouraging: the multiclass classifier delivers very good performance in each considered index. While such a performance level may not be enough for a full phishing defense, an effective phishing defense cannot rely on a single tool: a defense in depth strategy working at different levels is necessary. In this respect, we believe that our proposed approach may indeed

Table 1. Performance indexes, averaged across 9 experiments, for different values of n_S (number of websites to be protected). All values are in percentage, except for ET that is in hours:minutes format.

n_S	CA	BCA	FAR_I	FAR_U	MAR	ET
5	99.0	99.2	0.0	1.0	0.7	4:49
10	98.0	98.2	0.1	2.1	1.7	9:22
15	98.4	98.6	0.3	1.6	1.1	14:17

be practically viable and may provide complementary capabilities to existing tools. The ability to warn the user of a phishing site without any assumption on the reputation of the IP address, hosting provider, and website may be extremely useful for combating phishing attack strategies that are increasingly shorter and more targeted. While in principle one would like to be protected everywhere, we believe that even a protection on a user-selected set of 10–15 sites may be very useful [3].

Further investigation is obviously needed from several points of view, including in particular the ability to classify correctly screenshots that are not exact replicas of the original login page but that are similar enough to fool a user. To this end, we intend to explore more sophisticated data augmentation strategies and use suitably crafted artificial screenshots in testing as well. Adversarial attacks, i.e., login pages systematically crafted by an attacker to induce the classifier to output an attacker-chosen wrong class, are certainly to be explored as well.

References

1. The human factor: People-centered threats define the landscape. Tech. rep., Proofpoint (2018)
2. Afroz, S., Greenstadt, R.: PhishZoo: Detecting phishing websites by looking at them. In: 2011 IEEE Fifth International Conference on Semantic Computing. pp. 368–375 (2011)
3. Chen, T.C., Dick, S., Miller, J.: Detecting visually similar web pages: Application to phishing detection. ACM Trans. Internet Technol. 10(2), 5:1–5:38 (Jun 2010)
4. Lazar, L.: Our analysis of 1,019 phishing kits – blog — imperva. `https://www.imperva.com/blog/2018/01/our-analysis-of-1019-phishing-kits/` (Jan 2018), accessed: 2018-7-4
5. Maurer, M.E., Herzner, D.: Using visual website similarity for phishing detection and reporting. In: CHI '12 Extended Abstracts on Human Factors in Computing Systems. pp. 1625–1630. CHI EA '12, ACM, New York, NY, USA (2012)
6. Varshney, G., Misra, M., Atrey, P.K.: A survey and classification of web phishing detection schemes. Security Comm. Networks 9(18), 6266–6284 (Dec 2016)