DQN Agent for detecting Phishing Websites

Lorenzo Rinieri
Department of Computer Science and Engineering (DISI)
University of Bologna, Italy
lorenzo.rinieri@studio.unibo.it

Abstract

Phishing is a form of cybercrime that is defined as the art of mimicking a website of an honest enterprise aiming to acquire confidential information such as usernames, passwords and credit card details. In fact, a phisher usually setups a deceptive website, where the victims are conned into entering credentials and sensitive information. There are some characteristics that distinguish phishing websites from legitimate ones such as long URL, IP address in URL and adding prefix or suffix to domain. Inspired by the work of Chatterjee and Namin in [1], this paper introduces a DQN agent for detecting phishing website by analyzing the given URLs. It is also described how the dataset was built and what features are exploited by the deep reinforcement learning-based model.

1 Introduction

Phishing is a form of cyber attack typically performed by sending false correspondences that seem to be originated from a legitimate source. The objective of such attack is to gain access to sensitive information such as credit card numbers, credential data, or even to download and activate malware applications and viruses on the target machines. Phishing attacks typically start by sending an email that appears to come from legitimate company to victims asking them to update or validate their information by visiting a link within the email. Victims are then redirected to fake web pages that look very similar to those targeted authentic websites.
The detection of phishing attacks is mainly mapped into a classification problem. Therefore, machine learning techniques are considered as promising solutions. However, three major aspects need to be considered when adopting such techniques: selection of efficient classifiers; usage of distinguishing features; collection of representative dataset samples for training.
Hence, to solve this classification problem, a DQN agent is developed in this work. The deep reinforcement learning model is trained on a balanced and labeled dataset of legitimate and malicious URLs in which 11 features were extracted from the given URLs. The performance is measured using precision, recall, accuracy and F-measure. The key contributions of this paper are as follows:

1. Model the identification of phishing websites through Reinforcement Learning (RL), where an agent learns the value function from the given input URL in order to perform the classification task.
2. Collect phishing and benign URLs, then extract a set of features from them in order to build the dataset used both for training and testing purposes.
3. Map the sequential decision making process for classification using a deep neural network-based implementation of Reinforcement Learning.
4. Evaluate the performance of the deep reinforcement learning-based phishing URL classifier, compare its performance with the classifier of the reference paper [1] and and present a few additional considerations that may improve the results obtained by the agent.

2 Phishing detection methods

In general, as highlighted by Mohammad et al. in [2], there are two different approaches for identifying phishing websites. The first one is based on blacklist, by comparing the requested URL with those in that list. One drawback of this approach is that the blacklist usually cannot cover all phishing websites since within seconds a new fraudulent website is expected to be launched. The second approach is called heuristic-based, where several features are gathered from the website to classify it either phishy or legitimate. In contrast to the blacklist method, a heuristics-based solution can identify newly created phishing websites in real time. The efficiency of heuristic-based method depends on selecting a set of discriminative features that could help distinguishing phishing websites from legitimate ones. In their work, Hannousse and Yahiouche [3] classify website features in 3 main groups:

1. **URL-based features** are extracted by simply analyzing the text of URLs. Features of this class can also be divided into structural and statistical features. Structural-based features are concerned with the presence, position and nature of URL base elements (i.e., protocol, domain, subdomains, path, port, and top level domain). Examples of such features are the presence of ports and use of 'https' protocol. Statistical-based features are concerned with the number or distribution of URL base elements, specific words, or characters in the text of URLs. Examples of those features are the number of dots and subdomains and length of words.
2. **Content-based features** are extracted by loading the web pages of URLs and analyzing their HTML contents. They can be divided into hyperlink and abnormal content based features. Hyperlink features are concerned with the number, status, and nature of hyperlinks (i.e., internal/external) used in HTML tags. Abnormal content features are concerned with the identification of suspicious contents or scripts implementing suspicious behaviors. Examples of suspicious contents are the use of empty links and different domain names in the title tag of web pages. Examples of suspicious behaviors are submitting form contents to emails and disabling right clicks.
3. **External features** are obtained by querying reference third party services and search engines. Examples of such third party services are WHOIS[1], Alexa[2], Openpagerank[3] and Google[4]. External features are used as complementary and often considered as URL-based features. This is due to the fact that they are attained through querying external services with URLs or URL domains. Examples of this class of features are the domain age and the presence of a DNS record.

3 Dataset

The majority of contemporary studies about machine learning techniques applied to phisihing websites detection use self-collected datasets from different sources. The reason beyond this fact is related to the short-lived nature of phishing websites.
The experiment reported in the reference paper [1] was performed on the Ebbu2017 Phishing Dataset [4]. The dataset was prepared and publicly made available by Sahingoz et al. in their phishing URL detection work [5]. They prepared a balanced dataset containing both phishing and valid URLs: it contains 73,575 URLs, out of which 36,400 are legitimate and 37,175 are phishing. This dataset has three major drawbacks: it doesn't contain any feature, only URLs; a considerable number of phishing URLs don't exist anymore; some legitimate URLs locate pdf or ppt documents.
The UCI repository[5] also provides a pre-elaborated dataset for phishing detection. However, this dataset is are not suitable for replication and experimentation with new features because it doesn't include the URLs used for building the datasets.
Due to the absence of publicly available large dataset of malicious and benign URLs and due to all the drawbacks discussed above, it was decided to write from scratch a python script in order to

[1]WHOIS service: `https://who.is/`

[2]Alexa website: `https://www.alexa.com/`

[3]Openpagerank website: `https://openpagerank.com`

[4]Google search engine: `https://www.google.com`

[5]UCI repository: `https://archive.ics.uci.edu/`

build the dataset. First of all, the program pulls of 2000 legitimate URLs from Ebbu2017 Dataset [4] and 2000 phishing URLs from a ".csv" file downloaded from PhishTank [6]. Then, each one of them is written in a text file, followed by a binary value of 0 or 1 in order to indicate if the URL is legitimate or malicious, respectively. At the end, the dataset is generated by extracting the features from all the URLs, as explained in the following section. The final dataset contains only 3988 URLs because some problems occurred in the extraction of features from particular websites which contain an overlay in the home page. At the end, the dataset is split into 9 : 1 for training and testing purposes.

4 Feature extraction

Following the reference work in [1], from each URL is extracted a set of 11 binary features, as listed below:

1. **HTTPS Protocol**. Sensitive information is transferred using HTTPS protocols and utilization of such secure protocol is typically an indication of being safe. If the URL protocol is HTTPS then this feature it is set to zero. Otherwise, it is set to 1.
2. **IP Address**. IP addresses are used in hostnames to hide the identity of websites. IPs can also be used without dots or hexadecimal encoded. Presence of IPs in any format in hostnames is considered as phishing indicator. Hence, this feature value is set to 1 if there exists an IP address in the given URL, or to 0 otherwise.
3. **Long URLs**. The aim is to construct a URL to be long in order to obfuscate the malicious part. Hence, this feature is set to 1 if the URL is longer than 54 characters and thus classify the given URL as being suspicious. Otherwise, it is set to 0.
4. **URL Containing the @ Symbol**. A browser is designed to ignore everything prior to an @ symbol in a URL. Hence, phishers can redirect a victim to a phishing website using this method. As a result, if a URL contains @ this feature receives the value of 1.
5. **Adding Prefix or Suffix**. Dash is rarely used in legitimate URL. Phishers add suffixes or prefixes separated by "-" to the domain name so that users feel they are dealing with the legitimate webpage. Also, for bypassing the search engine optimization component, phishers often add "-" to the domain name because popular search engines such as www.Google.com use "-" as a word separator. This feature is set to 1 when there is a "-" in the domain name, otherwise to 0.
6. **Sub-domains**. Phishers often add valid sub domain names in the URL to make it appear as a legitimate URL. Hence, if the number of dots (i.e., ".") in the hostname is fewer than three, this feature is set to 0. Otherwise, it is set to 1.
7. **Anchor URLs**. An anchor is an element defined by the `<a>` tag. If the domain of an anchor is different from that of the website or if the anchor does not link to any webpage (e.g.: `<a href="#">`, `<a href="#content">`, `<a href="#skip">`, `<a href="JavaScript::void(0)">`), then the anchor is classified as suspicious.
   According to [2], if the webpage has suspicious anchors more than 20% of the total of the anchors, then this feature should be set to 1 as an indicator of being a phishing website. Otherwise, it is set to 0.
8. **DNS Record**. Domain Name Server (DNS) is mandatory to retrieve the IP address of URLs for access. Therefore, URL domains must be registered within the DNS. DNS records contain information about the active domain names. Phishing websites are short lived and may not have any DNS record. This feature can be extracted from WHOIS[1] database: if the URL doesn't have any registered name servers in the WHOIS[1] database then this feature is set to 1; to 0 otherwise.
9. **Domain Age**. Since phishing websites are short lived, the age of URL domains is considered as a phishing indicator. Using the WHOIS[1] database, any website younger than one year old should receive a value of 1 for this feature. Otherwise it is set to 0.
10. **Unusual URLs**. If the URL does not exists in the registered domain names in WHOIS[1] database then this feature should be set to 1 to tag the URL as a possible phishing website. Otherwise it is set to 0.

11. **Request URL**. A phishing page usually has a high percentage of object references, such as images, to its target (the real site) in order to show similar appearance. By contrast, a legitimate web page usually requests its objects from its own domain. To develop a rule for this feature, it is necessary to calculate the rate of URLs in source code of the website that have different domain than the domain typed at the address bar. This is done by extracting the keyword <`src=`> from the website source code and check whether the domain in the URL is different from that in <`src`>. If the rate is greater than 20% the website is classified as phishing and this feature it is set to 1. Otherwise, it is set to 0.

The first 10 features are directly taken from the reference paper [1], in which a total of 14 features are used. In this work, the features related to Link Hiding, Page Redirects, Pop-up Windows and Server from Handlers are not considered for the sake of simplicity and due to the high amount of time requested by the implementation of an automatic script able to extract them. The Request URL feature, instead, is extracted as described by Mohammad et al. in [2]: it was decided to insert it in the set because is a very common characteristic of phishing websites. In the implementation of the script used for building the dataset, WHOIS service is interrogated by using python-whois module [7].

5 Deep Reinforcement Learning model

5.1 Problem Statement

The problem of detecting phishing URLs can be formulated as a binary classification problem, in which the prediction classes are "phishing" or "benign". Let's denote a training dataset with *T* URLs along with data and class labels in the form of $(u_1, x_1), (u_2, x_2), ..., (u_T, x_T)$ where:

- u_i for $i = 1, 2, ..., T$ denotes a given URL in the training set *T*.
- $x_i \in \{0, 1\}$ for $i = 1, 2, ..., T$ corresponds to the label of the underlying URL where $x_i = 0$ implies benign and $x_i = 1$ indicates a phishing URL, respectively.

After the feature extraction, each URL u_i is represented as a d-dimensional (with $d = 11$) vector of features: $v = \{v_1, v_2, ..., v_i\}$, such that $v_i \in \mathbb{R}^d$. The Deep Reinforcement Learning algorithm can be seen as a function $f : \mathbb{R}^d \to \mathbb{R}$ which is applied on v to predict the class label. All the feature vectors and the class assignment of each URL in training set are normalized to binary values 0 and 1.

5.2 DQN

The training and classification algorithm is presented in Algorithm 1. This algorithm is based on the original Deep Q Network (DQN) learning algorithm proposed by Mnih et al. in [8]. The DQN uses experience replay for learning. The experience replay is the information about the state transition, action, reward for q-value learning. The learning process uses an experience memory *M* to store the information (s_1, a_t, r_t, s_{t+1}) and samples mini batch (Bm) from *M* to perform gradient decent as per the loss function $L(\theta)$:

$$L(\theta) = \sum_{(s_1, a_t, r_t, s_{t+1}) \in Bm} (y - Q(s, a, \theta_k))^2 \tag{1}$$

Where y would be the desired approximation of q-function and takes the form of:

$$y = \begin{cases} r_j & terminal_j = T \\ r_j + \gamma max_{a_{t_1}} Q(s_{t+1}, a_{t+1}, \theta_{k-1}) & terminal_j = F \end{cases} \tag{2}$$

Where j is a sample from M, $terminal$ is the condition when state-action pairs have maximum cumulative rewards and F and T are Boolean values.
The agent receives the input vector representation of the URL, one at a time, perform the actions and obtains the reward for that action. In this phishing URL classification problem, the state s_t at time step t is defined by the vector representation of the current URL extracted from the dataset T: the vector is composed by 11 elements, i.e. the features. The rewards are based only on the current state and on the chosen action. The action $A = \{0, 1\}$ is a binary output referring to the class label (i.e., Phishing or Benign) of the URL u_t. A reward r_t corresponding to an action a_t signifies if the agent has classified u_t correctly or incorrectly. If the class label of a particular u_t is denoted by l_t, then R is defined as:

$$R = \begin{cases} 1, & a_t = l_t \\ -1 & a_t \neq l_t \end{cases}$$

Algorithm 1: Deep Q-learning with Experience Replay

```
Initialize experience memory $M$;
Initialize Q-value function with random weights $\theta$;
Define episodes $K$;
Initialize target Q-value function with random weights $\theta^* = \theta$;
for episode $= 1, ..., K$ do
    Initialize $s_1 = v_1$;
    Initialize the pre-processed sequence function $\phi_1 = \phi(s_1)$;
    for $t = 1, ..., T$ do
        Perform action $a_t$ based on $\epsilon$-greedy: $a_t = \begin{cases} \text{Random-action} & \text{probability}\epsilon \\ argmax_a Q(\epsilon(s_t), a, \theta) & \text{otherwise} \end{cases}$
        Observe $r_t$ for action $a_t$;
        Set $s_{t+1} = s_t, a_t, v_{t+1}$;
        Set pre-processed sequence function $\phi_{t+1} = \phi(s_{t+1})$ ;
        Save $(\phi_t, a_t, r_t, \phi_{t+1}, terminal_j)$ in M;
        Randomly sample from M and set $y_j$ using equation 2;
        Perform gradient decent using equation 1;
    end
end
```

5.3 Neural network architecture

The deep neural network, according to the reference paper [1], is composed by 4 layers with a plot shown in Figure 1. The input layer is a fully connected layer which accepts the vector of feature as input. Then there are two fully connected layers with 32 hidden units and with ReLU activation. The output layer, instead, is a fully connected layer with softmax activation and two outputs (i.e. the Q-values for each action). The learning rate was set at 0.001. The adam optimizer was used for optimizing the parameters of the neural network.

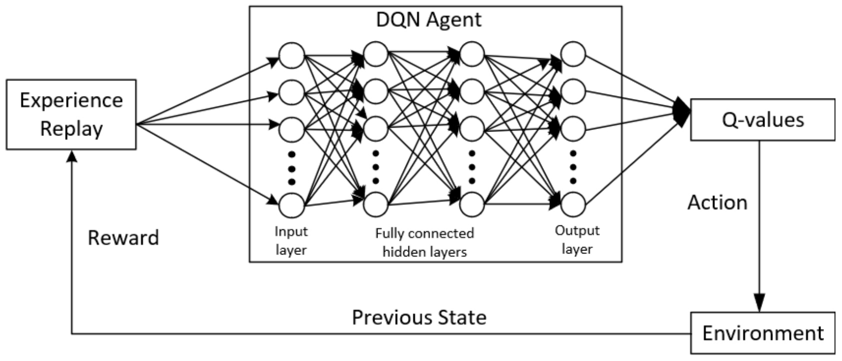

Figure 1: Deep Q Network.

5.4 Evaluation Metrics

As in the reference paper [1], the performance of the proposed model has been assessed using relevance measures like precision, recall, accuracy and F-measure. To calculate these measures, it is necessary to calculate the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) predictions. TP and TN refer to the correctly classified results while FP and FN are the misclassified data. Using these 4 values, the performance measures are obtained as follows:

- $Precision = \frac{TP}{TP+FP}$. In classification problem a precision value closer to 1 implies the predicted labels are closer to truth.

- $Recall = \frac{TP}{TP+FN}$. A recall value closer to 1 implies the all the testing samples could be predicted using the specified model.
- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$. A accuracy score closer to 1 implies a high performance of the system.
- $F - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$. It is the harmonic mean of precision and recall and it represents the model's resilience.

6 Results and conclusions

After training the agent for 90 epochs, which correspond to 5 hours approximately, the agent is able to correctly classify 331 URLs out of the total 389 in the testing set. Training the agent for more epochs gives worst results, maybe due to overfitting. In the following Table 1 are reported the results obtained and compared with the ones of the reference work in [1], which are almost equal:

Table 1: Comparison of the performance measures between this work and the reference one.

	Precision	Recall	Accuracy	F-Measure
Results of this work	0.851	0.813	0.851	0.831
Results in the reference paper	0.867	0.88	0.901	0.873

In order to obtain better results, the experiment may be repeated in the future using a larger dataset, adding or removing some features and modifying the reward function to explore the performance of the model.

References

[1] M. Chatterjee and A. Namin, "Detecting Phishing Websites through Deep Reinforcement Learning", *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, pp. 227-232, 2019. Available: 10.1109/compsac.2019.10211.

[2] R. M. Mohammad, F. Thabtah and L. McCluskey, "An assessment of features related to phishing websites using an automated technique," *2012 International Conference for Internet Technology and Secured Transactions*, 2012, pp. 492-497.

[3] A. Hannousse and S. Yahiouche, "Towards benchmark datasets for machine learning based website phishing detection: An experimental study", *Engineering Applications of Artificial Intelligence*, vol. 104, p. 104347, 2021. Available: 10.1016/j.engappai.2021.104347.

[4] "Ebbu2017 Phishing Dataset", *GitHub*, 2021. [Online]. Available: `https://github.com/ebubekirbbr/pdd/tree/master/input`. [Accessed: 01-Jul-2021].

[5] O. Sahingoz, E. Buber, O. Demir and B. Diri, "Machine learning based phishing detection from URLs", *Expert Systems with Applications*, vol. 117, pp. 345-357, 2019. Available: 10.1016/j.eswa.2018.09.029.

[6] "PhishTank | Join the fight against phishing", *Phishtank.org*, 2021. [Online]. Available: `http://phishtank.org/`. [Accessed: 30-Jun-2021].

[7] "python-whois", *PyPI*, 2021. [Online]. Available: `https://pypi.org/project/python-whois/`. [Accessed: 02- Jul- 2021].

[8] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. "Playing atari with deep reinforcement learning". *arXiv preprint*, 2013. Available: arXiv:1312.5602, 2013.