Phisherman

CMPE 272, Fall 2021 - Team 31

Dhruv Dinesh Soni
Computer Software Engineering
San Jose State University
San Jose, California
dhruvdinesh.soni@sjsu.edu

Philip Salire
Computer Software Engineering
San Jose State University
San Jose, California
philip.salire@sjsu.edu

Meera Kumar
Computer Software Engineering
San Jose State University
San Jose, California
meera.kumar@sjsu.edu

Rugved Manoorkar
Computer Software Engineering
San Jose State University
San Jose, California
rugvedprasad.manoorkar@sjsu.edu

***Abstract*—Phishing is a major a growing type of social engineering attack. It claims millions of victims every year, yielding attackers hundreds in millions of stolen dollars from victims. Current defenses mainly involve educating users, e.g. companies hosting training programs, and email filters that prevent receipt of phishing attempts or sends them to the spam box. Clearly, these defenses have not been adequate as internet users and companies endlessly fall victim to phishing attacks. Additional defenses need to be taken.**

Our project, Phisherman, is a Chrome extension that provides active in-browser phishing protection via blacklists, machine learning URL classification, and heuristic methods of detection.

***Index Terms*—phishing, chrome app, machine learning**

Links and Installation

The landing page is hosted at: http://54.67.112.73:3000. There is a download link for the Chrome extension on the landing page. To install the extension, extract the downloaded zip and navigate to chrome://extensions in Google Chrome, click the "Load unpacked" button, and open the path to the downloaded extension. It may be necessary to refresh the extension on each browser start-up to make sure that the extension is active.

I. Introduction

Phishing is a prevalent type of social engineering attack that has been growing steadily for the past years. Monthly phishing attacks reported by the Anti-Phishing Working Group have more than doubled from roughly 75,000 to over 250,000 from October 2019 to July 2021 [1]. The FBI reported in 2020 to have recorded over 250,000 reports of phishing victims, with over $54 million dollars in loss [2].

A major form of defense being taken to counteract these phishing campaigns are user education, in which companies dedicate time and resources to teach employees how to recognize phishing attempts. In addition, active email filters are commonly used to filter out detected phishing attempts, in addition to filtering spam and overall maintaining data-loss-prevention.

However, these have clearly proven to be insufficient as phishing has only grown more prevalent and severe over the past years; With regards to user education and email filters, these are insufficient as even the most educated person can be fooled and phishing occurs commonly outside of email. Obfuscating the URL host with an IP, another domain, large host names, and misspellings are common methods attackers use to evade detection while confusing users. For example, a common form of phishing is fake landing or login pages wherein an attacker hosts a web page that is similar or identical to legitimate sites in effort to steal login credentials e.g., an attacker may host a fake Facebook login page at a domain like "facebook.help-supportt.info" that harvests any credentials that a victim user attempts to enter.

A common method to detect malicious URLs is by using a blacklist of known malicious URLs. This list can be compiled and updated over time, PhishTank being such an example. However it's impossible to maintain an exhaustive list as new phishing sites are created continuously. As mentioned before, many attacks appear to be well known websites to avoid detection. To overcome this limitation, computer scientists have applied machine learning techniques for malicious URL detection. ML models are capable of classifying a URL as malicious or legitimate which can be generalized to new URLs. So we seek to combine both methods that have shown use in different situations.

This project aims to provide protection against phishing by providing a Chrome browser extension that scans web pages for phishing by providing details of web pages that the user navigates to into a trained phishing detection model, heuristic detection algorithms, and phishing blacklists. Users are then alerted if the extension detects the navigated web page to be a phishing attempt.

II. Related Works

A. Machine Learning

The identification of phishing attacks has been extensively studied, but no one method has proven to be utterly perfect. For example, it is possible to curate a white list of safe websites based on a user's most visited legitimate websites and achieve an accuracy of 86.02% [3].

Many existing machine learning models are based on hand crafted features. One popular example is CANTINA [4]. This models gathers the most frequent words on a page using tf-idf. The top five terms are considered page descriptors that are queried in a search engine. If the domain is within a certain rang of results, the web page is considered legitimate. The issues with this approach are the time and computational cost of getting and searching the frequent words, inability to catch sites that have been promoted within a search engine, and screening newly made sites. Other approaches include SVM with 6 human made features [5], association rule mining using tf-idf to generate rules [6], and random trees. These methods have good performance with over 93% accuracy but lag in efficiency.

There are multilayer perceptron (MLP) models with 94% accuracy from using features such as URL length, occurrence of the IP address, etc [7]. Another MLP used 30 features to get an accuracy of 97% [8]. These features are found in the UCI dataset, but are not always instantaneously found when a user accesses a website. Also while the accuracy of this models are high on their training test set, human designed features can fall out of date.

As many attackers design a site similar to an existing website or brand in order to fool consumers. Deep learning models such as PhishZoo, LogoSENSE, and Phishpedia focus on predicting based off of screenshots and URLs. Phishpedia has proven to be able to detect new phishing sites with prediction inference time at 0.2 seconds [10].

There are some approaches that solely analyze the URL using an RNN. Bahnsen achieved 98% accuracy with an LSTM model that we based our model on [9]. They used 2 million URLs from common crawl and PhishTank. A similar approach was made using CNN with embeddings and one-hot encoding with 99% accuracy [11].

III. Methodology

Phishing can occur anywhere on the internet. As such, our front-end approach to provide phishing detection is by extending the web browser by creating a Google Chrome extension via the Chrome Extension API. Most web browsers including Google Chrome are forks of the open-source Chromium web browser, which includes the public extension API. This means that this extension can also easily be deployed to other popular web browsers like Safari, Firefox, Edge, Opera, etc., significantly increasing the potential amount of users that this extension may defend.

Detection methods include a regularly updated phishing URL blacklist, a machine learning model to classify phishing URLs, and heuristic algorithms.

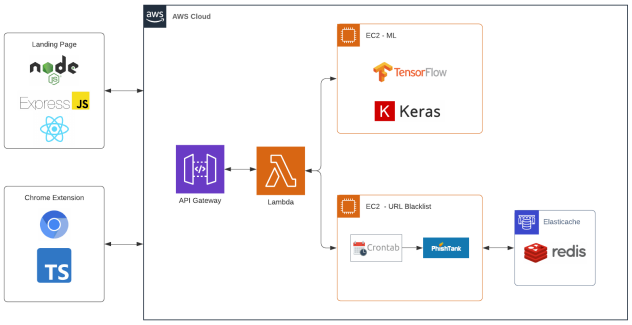

Fig. 1. Architecture Diagram

To maintain an up-to-date blacklist, it is necessary to host it on a server as the alternative of storing the blacklist locally within the extension would require the user to update his or her extension upwards of every few hours to stay fully protected, which is not a good user experience.

Similarly, the machine learning model needs to be hosted on a server. This allows updates to the model to easily be pushed without requiring the user to update, and reduces the size of the extension.

The data leveraged to detect phishing is the URLs that the user navigates to. A surprisingly significant amount of information can be extracted and inferred from just a URL. As such, the machine learning model is trained on URL data, the blacklist stores known phishing URLs, and the heuristic methods analyze the URL.

URL based detection is the simplest, but can miss a lot of positives so in future work additional data should be used. For example, the pages' HTML and related HTTP content should also be used. Screenshots may also be used since much page content often only is visible after JavaScript is executed.

Generally, the following steps are needed for the detection:

1) The user browses to a web page
2) The Chrome extension sends the web page's URL to the server
3) The server sends the URL to the blacklist or machine learning model
4) The server responds with detection results
5) If phishing is detected, the Chrome extension alerts the user

A. Architecture

The URL blacklist and machine learning model need to be hosted on a server and accessible to the Chrome extension and landing page. To accomplish this, Amazon Web Services (AWS) is used. The blacklist and machine learning model are hosted on two AWS EC2 instances, both of which exposed to the public via a REST API, hosted with AWS API Gateway. API requests are forwarded to AWS Lambda. Ultimately, Lambda processes the request and forwards it to either the URL blacklist or machine learning model on EC2, depending on which detection method was requested. Figure 1 illustrates this in an architecture diagram.

B. URL Blacklist

The URL blacklist maintains a list of known phishing URLs. Currently, it is implemented by keeping an up-to-date copy of the Phishtank database — a public, crowd-sourced database of verified phishing URLs maintained by Cisco — in AWS Elasticache Redis. To keep the local Phishtank database up to date, Crontab is used to call a Python script that fetches the Phishtank database every 24 hours and load it into Elasticache Redis. It is hosted on AWS EC2 and exposed with a Python Flask app.

Redis is optimal for this use case since the goal of the blacklist is to simply check whether a given URL exists in the database. Redis, being an in-memory database, is much faster and efficient than traditional databases that store to disk. This allows requests to be served quickly and also avoids much of the overhead associated with maintaining full-featured SQL or NoSQL databases.

C. Machine Learning

A long short term memory (LSTM) model is able to form some understanding and correlation within sequences of characters in a URL. While classifying a string of characters, we assume that the patterns within them influence their classification as malicious or benign. Regular LSTM only considers the context of characters moving from left to right. Bidirectional LSTM analyzes a URL from both sides. Therefore a character is analyzed for it's placement within the left and right context. In NLP tasks, context is important and used in popular models. Using a bidirectional LSTM sets our model appart from the work done in [9].

Preliminary data processing showed that length of malicious and benign URLs are about the same. So using URL length as an easy heuristic was removed. As both types of URLS are right skewed, the average length for phishing URLs was 74 characters and for safe URLS was 50 characters. Only 5% of the URLs in our dataset had over 128 characters. Therefore, character embedding was limited to the first 128 characters of all URLs. Each character is tokenized and then inputted into the 128-dimension embedding layer. The embedding layer is fed into 3 bidirectional LSTM layers. Finally classification is performed using a sigmoid neuron in a dense layer.

To be implemented into the chrome application, the best model and a Flask API are dockerized and pulled onto an EC2 instance. Lambda will then send requests to this end point.

D. Heuristics

Heuristic detection methods are implemented in JavaScript to be easily imported to both the Chrome extension and landing page to use client-side. Three heuristic methods are applied: checking the registration date of the domain, if the URL uses basic authentication, and for misspellings of popular websites.

All heuristic methods are applied locally in the user's client. Only checking the age of the domain requires externally doing a WHOIS lookup. Implementation details are described in Experiments.

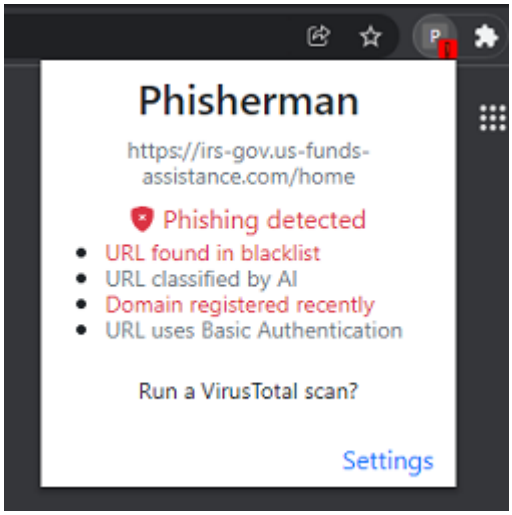

Fig. 2. Chrome extension correctly detects a phishing site

E. Chrome Extension

Chromium-based web browsers provide a JavaScript API to develop browser extensions. Specifically, the Phisherman extension is built for the Google Chrome web browser, but it may be deployed to any other Chromium-based browsers with minor changes depending on the specific browser. TypeScript is used for the code base.

As seen in figure 2, the extension displays the URL of the last visited URL, if it detected that URL for phishing, and the detection methods that flagged it for phishing, if any. Additionally, there is an option to run a VirusTotal scan of the URL.

F. Landing Page

A web page is also hosted on an AWS EC2 instance to serve as another front-end for users looking to manually search for phishing URLs that they encounter. The code base is developed with React.js with Node.js and Express.js on the back-end.

IV. Results

A. Machine Learning

1) Dataset: A dataset of real and phishing URLs was constructed for training. The original dataset was from the paper CatchPhish [12]. This dataset has 126,077 URLs with legitimate sites from both common-crawl and Alexa database and phishing sites from PhishTank. This dataset was imbalance with a ratio of 2:1 for safe to phishing URLs. To make up for this difference, we tested a model with extra 12,479 URLs from the latest PhishTank updates.

2) Experiment Design: We optimized the model across various learning rates and changes to the input dataset. The performance evaluation was done using standard evaluation metrics:

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
- Recall = $\frac{TP}{TP+FN}$

- Precision = $\frac{TP}{TP+FP}$
- AUC

3) Results: The model was built with a tensorflow backend with 10 epochs and trained on SJSU's HPC with 256 GB RAM. As seen in Table I, our model was able to reach an accuracy of 92% consistently. This is comparable to other models mentioned in the Related Works. Initially, the flask app was deployed with the model training on an unbalanced dataset since the metrics appeared to be better. But when using the Chrome extension, the model flagged many legitimate websites as unsafe. Due to double the amount of safe URLs during training, the model appeared to work well when tested with samples set aside from the same dataset, but failed to accurately predict safe URLs it hasn't seen before. So training was done after balancing the dataset. The D3 balanced dataset had 81,336 URLs and the D3 + Phishtank dataset had 106,294 URLs. The models resulting from the balanced datasets showed better results during real time use. While the D3+Phishtank model had slightly lower recall during training, this model was able to flag URLs with slight typos as malicious which is promising for zero day detection of a malicious URL.

TABLE I

Dataset	lr	AUC	Accuracy	Recall	Precision
D3 (unbalanced)	0.001	0.9719	0.9214	0.9153	0.9259
D3 (balanced)	0.001	0.9684	0.9216	0.8470	0.9110
D3 (balanced)	0.008	0.9655	0.9179	0.9138	0.9215
D3+Phishtank	0.008	0.9656	0.9123	0.8992	0.9213

Metrics of bidirectional LSTM model using various inputs and learning rates (lr). Increasing the lr showed improvements in the recall and precision.

B. Heuristic Approaches

1) Age Check: Phishing websites typically are short-lived as defenses and entities against phishing are constantly on the look out for phishing sites and updating defenses and providing awareness to protect against new known threats. As defenses are updated and more awareness generated, less victims are claimed as would be victims are protected and/or are aware the attacks. Therefore, phishing sites rarely last long, typically ranging from a few days to upwards of a year, as opposed to most legitimate websites like Google, Yahoo, Youtube etc. that were registered many years ago in the 1990s or early 2000s. In short, young websites are most suspicious to host phishing attacks.

To account for this, the age of any website can be looked up in WHOIS. As a simple heuristic check, if a website's domain was registered less than 60 days ago according to WHOIS, Phisherman classifies the website as a potential phishing site. This value was determined by trial-and-error on testing validated phishing URLs in the Phishtank database. It was observed that most true positives can be captured with this value. Still, many phishing sites were registered up to 100 days ago, but the higher the threshold value used, the more false positives of legitimate websites are introduced. Therefore, 60 days proved to be a good middle ground.

This age check detection method falls short when accounting for phishing websites hosted on web hosting service domains. For example, weebly.com, a legitimate site to build and host your own websites, is very popular among phishing sites; weebly.com is recorded very often in the Phishtank database.

2) Basic Authentication Check: Basic Authentication is type of authentication method over HTTP. This accomplished with the "Authorization" HTTP header with the "Basic" authorization scheme, that is:

```
Authorization: Basic <credentials>
```

This was once a popular authentication method, and so to make authenticating with Basic Authentication more convenient, a standard URL scheme was introduced that lets browsers parse for username and password to use in Basic Authentication. It follows the following format:

```
https://username:password@host/
```

This format has been widely abused by phishing attackers. Attackers may create a link like `http://google.com@192.168.1.1` to make it look like that it goes to `google.com`, when in reality `google.com` is the Basic authentication username, and will navigate to the defined host. In this case, the URL navigates to IP address `192.168.1.1`, not Google.

The URL can be more deceptive by using more convincing usernames and obfuscated IP addresses. For example, both

```
http://howsecureismypassword.net@3232235777
https://www.facebook.com+settings&tab=
privacy@%3192%2E%3168%2E%31%2E%31
```

may appear to be safe URLs that navigate to howsecureismypassword.net and facebook.com, but in reality both navigate to IP address `192.168.1.1` that is obfuscated. In this way, attackers can construct legitimate URLs that navigate to their own malicious IP addresses.

This type of authentication method is rarely used nowadays due to security concerns since it passes user credentials in plain-text. In fact, Basic Authentication in the URL is probably used more in phishing attacks than legitimately. Therefore, Phisherman classifies any URL that uses the Basic Authentication format as a potential phishing site.

3) Spell Check: A common method for attack is registering a domain with a slightly changed structure known as "typosquatting." Typosquatting is the omission or addition of one character in the address, removing dots between the subdomain and domain, or adding dashes to the domain and subdirectory to legitimate domains and URLs. There are also homograph attacks where a letter is replaced by similar looking letters or letters from other languages. In order to detect these kinds of attacks or maybe user mistakes, we used a database of the top 500 used domains worldwide. Separating the different parts of a URL and matching to popular websites was not an efficient approach to this problem. Some URLS have top level domains with multiple sections such as $co.jp$. Others will lack

subdomain or protocol to match with. Therefore, we utilized a spellchecking library, enchant, which ran on a personal word list of most visited webpages. A quick check against the URL's extracted root (subdomain, domain, and top-level domain) can verify if it's a known page or misspelled. This feature suggests to users the real page they assumed they were going to in case of a spelling or click on a malicious link. This algorithm is able to verify URLs with top-level domains for other countries.

V. Conclusion

Defenses against phishing need to be strengthened as more and more people continue to fall victim to phishing each year. Email filters work well, but phishing occurs anywhere on the internet, not to mention outside of the internet like in-person, over the phone, social media, etc. User education helps raise awareness, but still anyone can be deceived by a phishing actor.

Phisherman provides protection across the entire internet by extending the web browser with the Chromium extension API to provide active phishing detection. Users are protected by the extension as it passes URLs that they navigate to through a blacklist of known phishing URLs, heuristic methods such as checking the domain age, checking for Basic Authentication URLs, and checking for misspellings of popular URLs. Any positives returned by any detection method causes the extension to redirect the user away from the phishing web page.

Future work includes extending current or adding new detection methods to account for more data other than the URL, especially the HTML content of the navigated page and screenshots. There is a lot that can be extracted and inferred from a URL as covered in this paper, but in the case that a true positive is missed by URL detection, an abundance of data can subsequently be extracted from the page's HTML or screenshot to create other detection methods.

References

[1] "Phishing Activity Trends Report, 3rd Quarter 2021," APWG. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q3_2021.pdf. [Accessed: 08-Dec-2021].

[2] "2020 Internet Crime Report." [Online]. Available: https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf. [Accessed: 08-Dec-2021].

[3] Jain, A.K., Gupta, B.B. A novel approach to protect against phishing attacks at client side using auto-updated white-list. EURASIP J. on Info. Security 2016, 9 (2016). https://doi.org/10.1186/s13635-016-0034-3

[4] Y. Zhang, J.I. Hong, L.F. Cranor Cantina: a content-based approach to detecting phishing web sites Proceedings of the 16th International Conference on World Wide Web, ACM (2007), pp. 639-648

[5] M. Zouina, B. Outtaj A novel lightweight url phishing detection system using svm and similarity index Hum.-centric Comput. Inf. Sci., 7 (1) (2017), p. 17

[6] S.C. Jeeva, E.B. Rajsingh, Intelligent phishing url detection using association rule mining. Hum.-centric Comput. Inf. Sci., 6 (1) (2016), p. 10

[7] R.M. Mohammad, F. Thabtah, L. McCluskey, Predicting phishing websites based on self-structuring neural network. Neural Comput. Appl., 25 (2) (2014), pp. 443-458

[8] F. Feng, Q. Zhou, Z. Shen, X. Yang, L. Han, J. Wang, The application of a novel neural network in the detection of phishing websites, J. Ambient Intell. Humaniz. Comput. (2018), pp. 1-15

[9] A.C. Bahnsen, E.C. Bohorquez, S. Villegas, J. Vargas, F.A. González, Classifying phishing urls using recurrent neural networks, 2017 APWG Symposium on Electronic Crime Research (eCrime) (2017), pp. 1-8, 10.1109/ECRIME.2017.7945048

[10] Y. Lin et al., "Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages", in 30th $USENIX Security Symposium (USENIX$ Security 21), 2021.

[11] Wei Wei, Qiao Ke, Jakub Nowak, Marcin Korytkowski, Rafał Scherer, and Marcin Woźniak. 2020. Accurate and fast URL phishing detector: A convolutional neural network approach. Comput. Netw. 178, C (Sep 2020). DOI:https://doi.org/10.1016/j.comnet.2020.107275

[12] Rao, R.S., Vaishnavi, T. & Pais, A.R. CatchPhish: detection of phishing websites by inspecting URLs. J Ambient Intell Human Comput 11, 813–825 (2020). https://doi.org/10.1007/s12652-019-01311-4