arXiv:2409.10547v1 [cs.CR] 1 Sep 2024

Efficient Chrome Extension for Phishing Detection Using Machine Learning Techniques

Leand Thaçi[1], Arbnor Halili[1*], Kamer Vishi[2] and Blerim Rexha[1]

[1*]Faculty of Electrical and Computer Engineering, University of Prishtina, Kodra e Diellit, p.n., Prishtina, 10000, Kosovo.
[2]Department of Informatics, University of Oslo, Gaustadalléen 23B, Oslo, N-0373, Norway.

*Corresponding author(s). E-mail(s): arbnor.halili@uni-pr.edu;
Contributing authors: leand.thaci@student.uni-pr.edu;
kamerv@ifi.uio.no; blerim.rexha@uni-pr.edu;

Abstract

The growth of digitalization services via web browsers has simplified our daily routine of doing business. But at the same time, it has made the web browser very attractive for several cyber-attacks. Web phishing is a well-known cyberattack that is used by attackers camouflaging as trustworthy web servers to obtain sensitive user information such as credit card numbers, bank information, personal ID, social security number, and username and passwords. In recent years many techniques were developed to identify the authentic web pages that user visits and warn them when the webpage is a phish. In this paper, we have developed an extension for Chrome as the most favorite web browser, that will serve as a middleware between the user and phishing websites. The Chrome extension named "NoPhish" shall identify a phishing webpage based on several Machine Learning techniques. We have used the training dataset from "PhishTank" and extracted the 22 most popular features as rated by Alexa database. The training algorithms used are Random Forest, Support Vector Machine, and k-Nearest Neighbor. The performance results show that Random Forest delivers the best precision.

Keywords: Cybersecurity, Machine Learning, Random Forest, Phishing, Cyberattacks

1 Introduction

Nowadays, cyber-attacks are considered to be the most prevalent attacks that threatens our security and privacy on the Internet. Starting from some simple actions up to critical cases like US Elections [1], cyber security has become a topic for every person and government around the world.

According to the IBM Security X-Force Threat Intelligence Index report published in 2022, 41% of all cyber-attacks came from phishing, making it the top 1 technique most used for initial attack vector [2]. Comparing presence of this technique in report of 2021, there is a raise in 8 percent [3]. Phishing is a cyber-attack in which attackers try to steal information or sensitive data from victims, camouflaging as a reliable source. Additionally, phishing has also become a bridge for other attacks, tricking users into downloading malicious documents like viruses, malware, or ransomware.

One of our biggest motivations to further explore this topic was a gradual increase over the last two years of the number of phishing sites, as is presented in Figure 1. Victims of attacks can be from different categories, from large com-
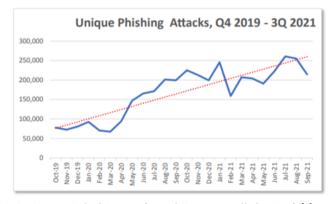

Fig. 1   Statistics in a period of 2 years (growth is present all the time) [4]

panies to individuals as ordinary Internet users. The most frequent attack on large companies is targeting the employees of that company through phishing emails. In terms of attacks which do not target a specific person or company, but are made for large masses, the most type of frequent attack is the imitation of large companies. According to statistics released by Vade Secure for the 2021 [5] and presented in Figure 2, the company which is most often imitated is Facebook which overcame "Microsoft", which was far away in first place. As of this writing, based on same report, Facebook has 2.8 billion users, a large pool of potential victims ready to bite on Facebook phishing messages and web pages. There is also a large number of phishing attacks that don't impersonate Facebook directly. Instead, the emails impersonate another brand and include links to Facebook phishing pages, exploiting user trust in the Facebook brand.

Fig. 2 The ten most impersonated brands in phishing attacks [5]

The battle between hackers and defenders on this topic is on daily basis, and the result is never clear. There is no clear index or methodology specified that would classify who is dominating the Internet ecosystem however we are witnesses of such a situation. This battle is best illustrated in Figure 3 where statistics of daily reporting within August 2022 for phishing are presented.
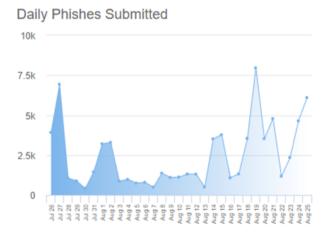

Fig. 3 Statistics of daily reports for phishing sites in PhishTank [6]

Based on this growth of phishing sites we evaluate that there is always room for improvement in terms of defense against these attacks and any research in this regard means fewer victims. Obviously, attacks of this kind cannot be solved once and for all, constant efforts must be made to study new methods of attack and to adapt protection against them. In this paper, we tend that through detailed research and analysis to provide new and efficient solutions for application by using the latest technologies and techniques. In addition to the tool development and training part, this paper gives importance also to the simplicity of using the tool. The goal is that every user regardless of their professional background in this field, be able to get the right information about the site being visited.

Work done for this solution combines knowledge from different fields of study, like Machine Learning, Web programming, and Web security. Machine Learning (ML) is seen as a mechanism that will enrich our tool with a way of learning and deciding based on a model that is created. The model will be created based on training data. Similar usage was seen earlier in a wide variety of applications, such as email filtering and computer vision, where it is difficult or impossible for conventional algorithms to perform the necessary tasks.

This paper's purpose is to propose a tool that can classify phishing sites. Classification falls under the Supervised Learning paradigm, where the model is trained with labeled data, a process that is often quite costly in all respects. Algorithms that will be discussed and that belong to this paradigm are:

- **SVM** (**Support Vector Machine**) - As it is presented in Figure 4 the classification boundaries are defined by the so-called support vectors. This is the perfect case when the data is sufficiently differentiable and there are no exceptions (outliers), respectively instances which are located outside the defined boundaries. If the model is trained using hard margins (hard margin classification) then the model works only if the data are linearly separated. For this reason, more flexible parameters of margins should be defined, respectively soft margin classification.

- **kNN (k-Nearest Neighbors)** - The principle behind kNN is to find a predetermined number of training samples closest to the distance from the new point and to predict from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or variable based on local point density (radius-based neighbor learning). Distance, in general, can be any metric mass: the standard Euclidean distance is the most common choice. Neighbor-based methods are known as non-generalized Machine Learning methods, as they simply "recall" all their training records.
Despite its simplicity, kNN has been successful in a large number of classification and regression problems, including handwritten figures and satellite image scenes. Being a non-parametric method, it is often successful in classification situations where the decision limit is very irregular.

- **Random Forest** - Is a flexible, easy-to-use algorithm that produces, even without adjusting parameters for specific cases, an excellent result most of the time. Random Forest is one of the most widely used algorithms,
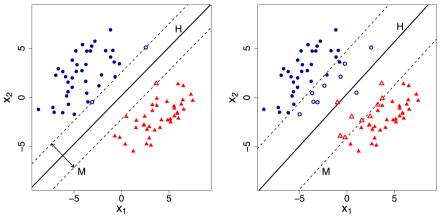
Fig. 4 Differences between soft and hard margins for SVM [7]

due to its simplicity and diversity (can be used for both classification and regression tasks). Random Forest is an advanced version of the Decision Trees algorithm, it basically uses the Decision Trees algorithm in different subgroups of the dataset and uses the average value to increase performance and avoid over fitting. This method is known as "bagging" or "bootstrap aggregating". Suppose we fit a model to our training data $Z = (x1, y1)$, $(x2,y2)$,.., (xN, yN), obtaining the prediction $\hat{f}(x)$ at input x. Bootstrap aggregation or bagging averages this prediction over a collection of bootstrap samples, thereby reducing its variance. For each bootstrap sample $Z^{*b}b$, b = 1, 2, ... ,B, we fit our model, giving prediction $\hat{f}*^b(x)$ [8]. The bagging estimate is defined by

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x) \tag{1}$$

Figure 5 presents the complete diagram of the Random Forest algorithm.

One of the first challenges when implementing Machine Learning in identifying phishing sites is the possession of a dataset which is updated with the latest data and has real data. Fortunately, there are many online databases with multiple datasets. One of the biggest anti-phishing company, where thousands of pages are submitted and validated every day is "PhishTank" [6]. Among useful information and statistics, this web page also has an API through which developers can obtain data.

The rest of the paper is organized as follows. The next section presents background information on phishing attacks. Section 3 describes the proposed model, namely NoPhish, that is developed to detect malicious websites. Section 4 details the evaluation results of the model. Finally, Section 5 concludes the paper.
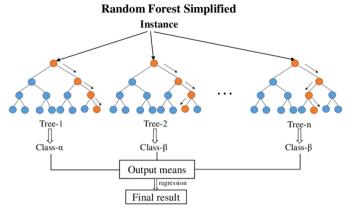
Fig. 5 Random Forest Diagram [9]

2 Related Work

Sadly, in the battle against phishing attacks, the attacking side is always one step ahead, so the defensive side needs always to keep up with the new methods of attacking [10]. The process of constant sophisticating the tools and techniques for protecting users from phishing attacks is forcing attackers to adapt and evolve their techniques. Today phishing attacks are becoming more and more polymorphic [11]. The situation becomes even worse when it is known that the most perfect attacks occur within minutes and do not recur twice by the attacker without improving for the next time. The "zero-day attacks" that by definition are attacks that have not happened before, today are not uncommon [12]. These attacks are always launched with a new set of IP addresses and a new variant of the virus.

The most common method of detecting phishing sites is by updating URLs and IPs addresses listed as phishing in the antivirus database, this method is known as "Blacklist" [13]. To avoid blacklists attackers use creative techniques to cheat the system by modifying the URL to look legitimate through obfuscation and many techniques other simple methods including the "fast-flux" method [14]. Through this method proxies are automatically generated to host the web page and new URLs are generated through algorithms. The main drawback of this method is that, it can not detect "Zero-day attacks". Thus, there must be at least one victim who will report this site and will blacklist that page. One of the most popular tools of this type is "Netcraft Extension" [15]. This gadget consists of a large community, and expects the first victim to report a page in order to inform other users not to visit that page.

Another method for detecting phishing sites is a system with predefined rules. Where the tool analyzes the content of a site and based on the rules it has predefined and decides whether that site is considered phishing or not [16]. The problem with this type of system is, although in principle it can stop "zero-day" attacks, predefined features are not guaranteed to be found in every attack so the number of false positives is very high and usage remains very

narrow. The most popular tool of this type of system is "Phish Detector"[17]. A tool that prevents phishing attacks in online banking. It is a "rule-based" system and claims zero false negative predictions. The disadvantage is that it is limited to the domain of online banking.

The third method, detecting phishing sites using Machine Learning, tends to stop "zero-day" attacks, including as many attacking domains as possible. Nowadays there are plenty of tools with this method, but there is always room for improvement. One of the early tools of this kind is the "Cascaded Phish Detector" which similar to this paper consists of a client-side component and a server-side component [18]. This gadget only takes into account the HTML content of the page, we think there are also other kind of features that can be extracted from a webpage and are informative as mentioned in Section 1.

The authors in [19] used Machine Learning to predict phishing. We find missing in this paper that there is no explanation of the ratio used to divide the training and testing set. Furthermore, there is no concrete implementation of this algorithm; it is not part of any tool, or we have not found any concrete implementation and ease of usage, which is one of the critical points of our work.

An interesting approach is found at [20]. Here an architecture which is mainly referred to a preprocessing of data is proposed. However, it only took into consideration one algorithm and could lead to a over fitting.

[21] used a technique of search for classification of a web page as phishing or not. Pseudocode presented on the paper shows that algorithm instead of learning is using services through API and direct requests to make a decision. In this case Google search engine has the main role since top six Google search results are used to make decision. It worth mentioning that in most cases, top six Google search results change very rare, hence, it could present a high potential for wrong classification.

We found interesting work of [21], which ranked Random Forest to be best (together with XGboost) among 12 algorithms used. However, even here there is no concrete implementation to link this result with a concrete tool.

Furthermore, from our research made on current work and state of the art, best to our knowledge, we could identify a research gap in this intensive field. There are three principles that we took into consideration:

- effectiveness of the solution,
- easiness of implementation and usage, and
- innovation.

From studies and work done until now we found solutions that are effective (like [20] but there is no implementation or concrete use. We found work that was easy to use [19] but not enough effective. Our solution tends to fulfill all these three principles and tends to do it in a novel, innovative way, nor presented until now by any author.

3 Our Approach - NoPhish

Since the classification of whether the site is suspicious or not will be done by the tool trained with Machine Learning, the question arises on what the tool will base its prediction. The main thing to look for when detecting suspicious phishing sites is the URL of the site. Nowadays it is not enough just to look at whether the site uses the secure HTTPS protocol for communication. According to the latest report from APWG (Anti-Phishing Working Group), about 80% of phishing sites use the protocol HTTPS [4]. Therefore, some properties are extracted from the URL based on which a page can be considered suspicious. Other features are derived from the content of the site, where the authors of these sites often focus only on copy the look of the real page and leave many flaws which can be identified. And the last category of site features are extracted from searching this site in the most popular databases of statistics about web pages, such as the Alexa Database [22]. The most important part of the tool is definitely the training algorithm, based on which the tool makes decisions about how to classify the pages it handles. Within Machine Learning we have a large diversity of algorithms which are adequate for different situations. For this paper the variety is narrowed down to the 3 most adequate algorithms, which will be tested and compared. They are: Random Forest, SVM (Support Vector Machine), kNN (k-Nearest Neighbor). Chrome Extension will play the role of the middle man between the user and the classifier. Google Chrome offers the ability to add a tool add-on which can be used on any page you browse, it is called a Chrome Extension. The question is, why exactly Google Chrome? According to November 2020 statistics [23], Chrome browser dominates with 65.3% of worldwide use.

The tool [1] consists of two main components: the "client-side" component with which the user will interact and, the "server-side" component which will be the classifier that decides whether a site is considered phishing and the web server which will play the role of mediator between the two components and will create the environment for the execution of the classifier.

Since the key point of this paper is Machine Learning, it was decided to use the Python language (version 3.9) for the implementation of the classifier component. One of the advantages of this language is definitely the large number of libraries. The Python language is today recognized as a leader in the field of Machine Learning, through the skicit-learn library the implementation of algorithms for Machine Learning is easier than ever. In addition Python also simplifies the data manipulation procedure for extraction of site features. The numpy library is used for manipulations with large multidimensional arrays, while matplotlib is used for graphical representations of results and statistical data needed for visualization. The "client-side" component will be developed as "Chrome Extension" for very easy user access. The reason for selecting the Google Chrome browser is the large percentage of users in the market. The

---

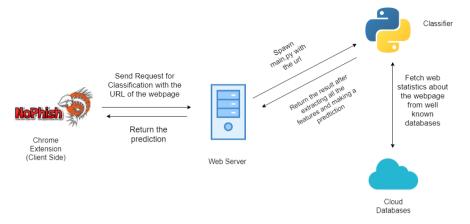[1]NoPhish Repository: https://github.com/LeandThaqi/NoPhish

Fig. 6 NoPhish architecture

Webserver who will make the link between the two components will be developed in the Node.Js language, the reasons being good performance and the ease of use.

Today the Internet world is rich in data. Many different sites contribute to the fight against phishing with different datasets. The dataset selected for the tool training for this paper was donated to the University of California, Irvine [24]. It was originally donated with data for about 2400 pages in 2015 but has been constantly updated and today this number has reached up to about 11000. The database contains data on thirty page features.

3.1 Determining the features based on which pages will be classified

Because the dataset contains features that help detect sites not only for phishing but also for other risks. Twenty-two of the thirty features listed bellow are considered to be phishing information.

1. **Using IP address instead of domain** - If an IP address is used as an alternative of the domain name in the URL, such as "http://125.98.3.123/fake.html", users can be sure that someone is trying to steal their personal information. Sometimes, the IP address is even transformed into hexadecimal code as shown in the following link "http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index .html".
2. **Long URL to Hide the Suspicious Part** - Phishers can use long URL to hide the doubtful part in the address bar
3. **Using URL Shortening Services "TinyURL"** - URL shortening is a method on the "World Wide Web" in which a URL may be made considerably smaller in length and still lead to the required webpage. This is accomplished by means of an "HTTP Redirect" on a domain name that is short, which links to the webpage that has a long URL.

4. **URL's having "@" Symbol** - Using "@" symbol in the URL leads the browser to ignore everything preceding the "@" symbol and the real address often follows the "@" symbol

5. **Redirecting using "//"** - The existence of "//" within the URL path means that the user will be redirected to another website.

6. **Adding Prefix or Suffix Separated by (-) to the Domain** - The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage

7. **Sub Domain and Multi Sub Domains** - If a page has many subdomains then it can be considered that the attacker is trying to disguise the site domain.

8. **Domain Registration Length** - Based on the fact that a phishing website lives for a short period of time, we believe that trustworthy domains are regularly paid for several years in advance.

9. **Favicon** - A favicon is a graphic image (icon) associated with a specific webpage. Many existing user agents such as graphical browsers and newsreaders show favicon as a visual reminder of the website identity in the address bar. If the favicon is loaded from a domain other than that shown in the address bar, then the webpage is likely to be considered a Phishing attempt.

10. **The Existence of "HTTPS" Token in the Domain** - The phishers may add the "HTTPS" token to the domain part of a URL in order to trick users.

11. **Request URL** - Request URL examines whether the external objects contained within a webpage such as images, videos and sounds are loaded from another domain. In legitimate webpages, the webpage address and most of objects embedded within the webpage are sharing the same domain.

12. **URL of Anchor** - An anchor is an element defined by the ¡a¿ tag. This feature is treated exactly as "Request URL". However, for this feature we examine:
    (a) If the ¡a¿ tags and the website have different domain names,
    (b) If the anchor does not link to any webpage.

13. **Links in ¡Meta¿, ¡Script¿ and ¡Link¿ tags** - Given that our investigation covers all angles likely to be used in the webpage source code, we find that it is common for legitimate websites to use ¡Meta¿ tags to offer metadata about the HTML document; ¡Script¿ tags to create a client side script; and ¡Link¿ tags to retrieve other web resources. It is expected that these tags are linked to the same domain of the webpage.

14. **Server Form Handler (SFH)** - SFHs that contain an empty string or "about:blank" are considered doubtful because an action should be taken upon the submitted information. In addition, if the domain name in SFHs is different from the domain name of the webpage,

this reveals that the webpage is suspicious because the submitted information is rarely handled by external domains.

15. **Submitting Information to Email** - Web form allows a user to submit his personal information that is directed to a server for processing. A phisher might redirect the user's information to his personal email. To that end, a server-side script language might be used such as "mail()" function in PHP. One more client-side function that might be used for this purpose is the "mailto:" function

16. **Abnormal URL** - This feature can be extracted from WHOIS database. For a legitimate website, identity is typically part of its URL.

17. **IFrame Redirection** - IFrame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the "iframe" tag and make it invisible i.e. without frame borders. In this regard, phishers make use of the "frameBorder" attribute which causes the browser to render a visual delineation.

18. **Age of Domain** - This feature can be extracted from WHOIS database (Whois 2005). Most phishing websites live for a short period of time.

19. **DNS Record** - For phishing websites, either the claimed identity is not recognized by the WHOIS database (Whois 2005) or no records founded for the hostname (Pan and Ding 2006). If the DNS record is empty or not found then the website is classified as "Phishing", otherwise it is classified as "Legitimate".

20. **Website Traffic** - This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit.

21. **Google Index** - This feature examines whether a website is in Google's index or not. When a site is indexed by Google, it is displayed on search results (Webmaster resources, 2014).

22. **Statistical-Reports Based Feature** - Various companies like PhishTank or StopBadware form statistical reports exposing the IPs and domains most used for phishing. This feature checks if the host is part of these reports.

3.2 Choosing the most efficient algorithm for our specific case

For this paper we took under consideration three classification algorithms, they are: SVM, kNN and Random Forest. All will be tested and the algorithm that performs the best will be selected. Figure 7 shows the results of the accuracy of the algorithms from tests by dividing the dataset into different proportions (training/testing): 50/50, 70/30, 90/10.
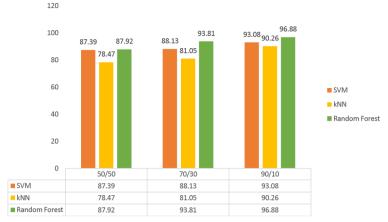
Fig. 7 Performance results for accuracy from the three algorithms

Table 1 Performance results for precision, recall, and confusion matrix

	Precision	Recall	TN	FP	FN	TP
SVM	0.913	0.969	422	57	19	601
kNN	0.901	0.904	431	48	59	561
Random Forest	0.921	0.970	435	44	17	603

Based on the extracted statistics it can be seen that the Random Forest algorithm outperforms other algorithms in this experiment in terms of accuracy. The efficiency of algorithms in ML is also measured by several other parameters such as: precision score, recall score and confusion matrix.

Based on the test results it is seen that Random Forest is the most accurate algorithm for this case. And considering the "bagging" feature of this algorithm which mitigates the problem of overfitting, the tool will be developed with the Random Forest algorithm. The following code snippet trains the model with the Random Forest Algorithm, this model will be used to predict if the websites are phishing (Listing 1).

```
1  clf4 = RandomForestClassifier(verbose=2, oob_score=True)
2  clf4.fit(features_train, labels_train)
3  predictions = clf4.predict(features_test)
4  accuracy = 100.0 * accuracy_score(labels_test, predictions)
5  importances = clf4.feature_importances_
6  std=np.std([tree.feature_importances_ for tree in clf4.estimators_],axis=0)
7  indices = np.argsort(importances)[::-1]
```

Listing 1 The code that trains the model using the Random Forest Algorithm

3.3 Feature Importance

The Random Forest algorithm also offers us the importance of each feature from the training. Through the *feature_importance_object*, the features are ranked based on how informative they were during the training. These percentages are calculated by the algorithm from MDI (Mean Decrease in Impurity) so as that feature reduces the impurity of the dataset. Figure 8 presents the ranking of the features according to the MDI.
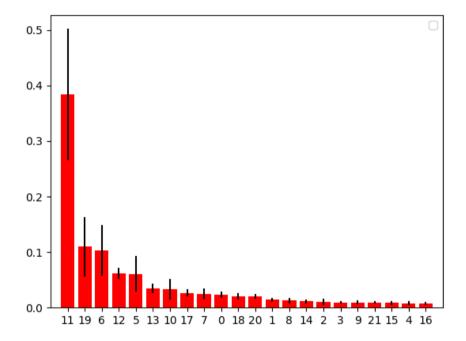

Fig. 8 Importance of features based on MDI

A problem that has emerged recently is that the importance of features based on impurity can inflate the importance of numerical features. Furthermore, the impurity-based feature importance of random forests suffers from being computed on statistics derived from the training dataset: the importances can be high even for features that are not predictive of the target variable, as long as the model has the capacity to use them to overfit. Alternatively, the importance of features can be calculated from permutation importances as illustrated in Figure 9.

As we can see from both the graph extracted from different methods that *feature 11* which is the URL of the anchor tags is the most important feature. Logically the danger of phishing sites is the redirection into suspicious sites.
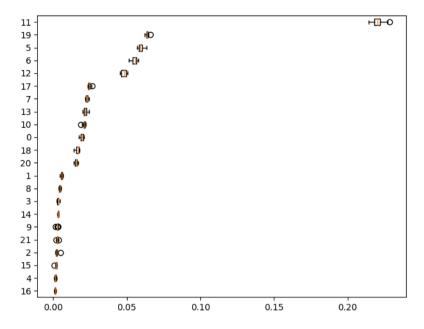
Fig. 9 Importance of features based on permutation

3.4 Client-side Component (The face of the tool)

The component with which the user will interact will be developed as a Chrome Extension. Chrome Extension development is simple, the three elements needed for a functional extension are:

1. *popup.html* - where HTML elements are defined,
2. *popup.js* - where functions are written in javascript, and
3. *manifest.json* - where extensions details are written. The following code snippet represents the manifest.json file (Listing 2).

```
1   {
2       "manifest_version": 2,
3
4       "name": "NoPhish",
5       "description": "This extension will analyze a page and predict if it is a phishing site or not , by
            ↪ using machine learning!",
6       "version": "0.8",
7       "browser_action": {
8        "default_icon": "icon.png",
9        "default_popup": "popup.html"
10      },
11      "icons": { "16": "icon.png",
12          "48": "icon48.png",
13          "128": "icon128.png" },
14      "permissions": [
15       "activeTab"
16      ]
17  }
```

Listing 2 manifest.json (chrome extension configuration file)

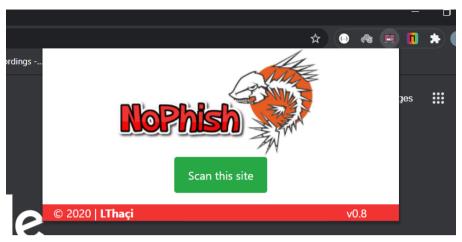The main purpose of Chrome Extension is to be as simple as possible.

Fig. 10 First view of the tool

The design is made in a manner that with a click of a button all the necessary information is obtained. Figure 10 shows the first view of the tool. The "Scan this site" button sends a request to the web server with the URL of the site, where user is currently located and waits for a response. The following code snippet sends the request to the web server (Listing 3).

```
1       chrome.tabs.getSelected(null, function(tab) {
2           let xhr = new XMLHttpRequest();
3           let url = "http://localhost:3000/detectphishing";
4           xhr.open("POST", url, true);
5           xhr.setRequestHeader("Content-Type", "application/json");
6           var data = JSON.stringify({"url": tab.url});
7           xhr.send(data);
8           ..
9       });
```
Listing 3 Sending the request to check if the website which the user is browsing is phishing

The webserver accepts the request (This can be seen in the following code snippet 4) , spawns the script which predicts if the site is phishing (This can be seen in the following code snippet 5) then returns the response , as presented in Figure 11, the prediction is initially displayed which notifies the user whether the tool evaluates the visited site as phishing or not. The calculated probability of how suspicious the site is is displayed and also for each feature of the site it is indicated which one failed during testing.

```
1         get(url).then((pageContent) => {
2           fs.writeFileSync('../innerHTML.txt',pageContent)
3           const pythonProcess = spawn('python',["../main.py",url]);
4           pythonProcess.stdout.on('data', (data) => {
5             response.writeHead(200, {'Content-Type': 'application/json'});
6             response.end(data.toString('utf8'));
7           });
```
Listing 4 Webserver accepting a request and spawning the python script which evaluates if the website is phishing

```
1     def main():
2       url = sys.argv[1]
```
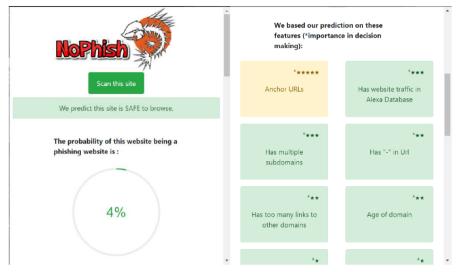
Fig. 11 The safe prediction information displayed

```
3    features_test = features_extraction.main(url)
4    # 2d array per scikit-learn
5    features_test = np.array(features_test).reshape((1, -1))
6    clf = joblib.load(LOCALHOST_PATH + DIRECTORY_NAME + '/classifier/
     random_forest_new_4.pkl')
7    prediction = clf.predict(features_test)
```

Listing 5 The python script which predicts if the website if phishing

Based on the principle that a false positive prediction is less harmful than a false negative prediction, in addition to predicting whether it is phishing or not, a warning area has been added. This area indicates that the site is predicted to be secure but there are questionable signs in some features, so attention is required from the user to review which site it is on. Besides when the prediction returns safe as presented in Figure 11, two other cases are presented in Figure 12.

4 Experimental Results

In order to test our tool against current solutions to this problem we did some experimental work. First we selected our testing subject which will be twenty-seven websites, fourteen of them are verified phishing websites taken from PhishTank[6] and the other thirteen are well known websites proven that they dont have any malicious intent.

The tools that we did compare our work with were: Google Chrome's built in firewall and the firewall of a world wide agency whose name will be undisclosed for privacy reasons. The methodology of testing was as follows, each website was scanned with the three tools. In the end we gathered all the information and displayed it as the confusion matrix(as presented in Figure 13) which we think is the best indicator of accuracy for this particular case.
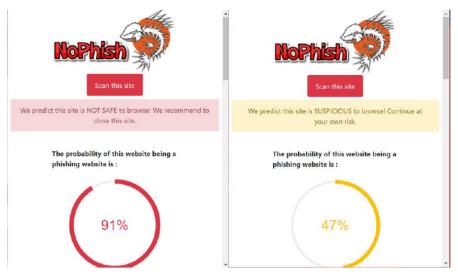
Fig. 12 The dangerous and warning prediction information displayed

Analyzing the results presented in Figure 13, one can conclude that NoPhish is the most efficient tool out of these three. If one analyzes the TP (True Positive) cases, one can conclude that NoPhish gives the best result, which means it detected more phishing sites than the other two tools. However, when one analyzes about TN (True Negative) and FP (False Positive) cases, it is easy to conclude that NoPhish is performing a bit worse comparing with other two mechanisms. NoPhish was found to have the least number of TN cases and the highest number of FP cases (even though it is a low number, difference is only one). This is a conservative part of our tool, always trying to put the client on the safe side. The reason for this behaviour is that NoPhish besides the safe and dangerous prediction has a warning zone. In warning zone are all links, that NoPhish technically predicts as safe but are very near to the threshold, thus NoPhish warns the user. These warning predictions were counted as TP. This method allows NoPhish to have zero FN (False Negative) predictions and rates NoPhish above the other two tools taken into comparison.

5 Conclusion

From the evaluation results, one can conclude that developing a classifier by applying Machine Learning gives satisfactory results. Besides good results, there is a great potential for more extensive research in machine learning and phishing detection techniques.

This paper's results are satisfactory by achieving high accuracy with a small number of false-negatives. Furthermore, the Random Forest algorithm proved more efficient in this research; it does not mean that there could not be another more successful algorithm in the future in other circumstances with extensive evolution of attacks. The features on which the forecast is based
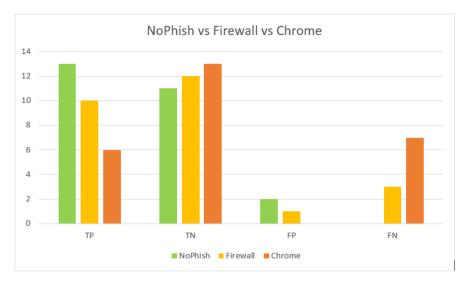
Fig. 13 Statistics based on the confusion matrix

should always represent the state-of-the-art attack properties. Such as the number four feature where if the URL contains "//" it is considered suspicious, and it can be removed in the future. Because some browsers nowadays have already stopped redirecting to other pages if the URL contains "//", but as long as this security "bypass" is present in all browsers can not be neglected.

6 Declarations

Ethical Approval
Not applicable.

Competing interests
Personal nature interest (continuous work on security research)

Authors' contributions
A.H. wrote the introduction sections text, contributed to related work, and the main manuscript text. K.V. wrote related work and contributed to the main manuscript text. L.TH. wrote mostly on the main manuscript and also dealt with experimental results. B.R. initiated the idea by writing the abstract and conclusion and was the main reviewer. All authors reviewed the manuscript.

Funding
Not applicable.

Availability of data and materials
https://github.com/LeandThaqi/NoPhish

References

[1] Press, A.: US official: Hackers targeted election systems of 20 states. [Accessed on 07.09.2022] (Retrieved from https://apnews.com/article/ c6f67fb36d844f28bd18a522811bdd18)

[2] IBM: X-Force Threat Intelligence Index 2022. [Accessed on 09.09.2022] (Retrieved from https://www.ibm.com/downloads/cas/ADLMYLAZ)

[3] IBM: X-Force Threat Intelligence Index 2021. [Accessed on 07.09.2022] (Retrieved from https://www.ibm.com/security/data-breach/threat-intelligence)

[4] APWG: Phishing Activity Trends Report. [Accessed on 07.09.2022] (Retrieved from https://docs.apwg.org/reports/apwg_trends_report_q3_2020.pdf)

[5] Secure, V.: Phishers' Favorites - 2021 Year-in-Review. [Accessed on 13.09.2022] (Retrieved from https://shorturl.at/lvzGJ)

[6] PhishTank: Statistics about phishing. [Accessed on 26.08.2022] (Retrieved from https://www.phishtank.com/stats.php)

[7] Kirchner, A., Signorino, C.S.: Using support vector machines for survey research. Survey Practice **11**(1) (2018). https://doi.org/10.29115/SP-2018-0001

[8] Hastie, T., Tibshirani, R., Friedman, J.H., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction vol. 2, pp. 282–283. Springer, New York, NY (2017)

[9] Gao, G., Wang, M., Huang, H., Tang, W.: Agricultural irrigation area prediction based on improved random forest model (2021). https://doi.org/10.21203/rs.3.rs-156767/v1

[10] Alkhalil, Z., Hewage, C., Nawaf, L., Khan, I.: Phishing attacks: A recent comprehensive study and a new anatomy. Frontiers in Computer Science **3**, 563060 (2021)

[11] Chauhan, R., Sabeel, U., Izaddoost, A., Shah Heydari, S.: Polymorphic adversarial cyberattacks using wgan. Journal of Cybersecurity and Privacy **1**(4), 767–792 (2021). https://doi.org/10.3390/jcp1040037

[12] Bursztein, E., Oliveira, D.: Deconstructing the Phishing Campaigns that Target Gmail Users. Black Hat 2019, Las Vegas, USA ([Accessed on 07.09.2022]). https://elie.net/talk/deconstructing-the-phishing-campaigns-that-target-gmail-users/

[13] Aljofey, A., Jiang, Q., Rasool, A., Chen, H., Liu, W., Qu, Q., Wang, Y.: An effective detection approach for phishing websites using url and html features. Scientific Reports **12**(1), 1–19 (2022)

[14] Rana, S., Aksoy, A.: Automated fast-flux detection using machine learning and genetic algorithms. In: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 1–6 (2021). https://doi.org/10.1109/INFOCOMWKSHPS51825.2021.9484614

[15] Prettejohn, M.: Netcraft Extension (Anti-phishing toolbar). [Accessed on 07.09.2022] (Available at: https://www.netcraft.com/apps/browser/). https://www.netcraft.com/apps/browser/

[16] Lin, Y., Liu, R., Divakaran, D.M., Ng, J.Y., Chan, Q.Z., Lu, Y., Si, Y., Zhang, F., Dong, J.S.: Phishpedia: a hybrid deep learning based approach to visually identify phishing webpages. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 3793–3810 (2021)

[17] M., M.: PhishDetector - True Phishing Detection. [Accessed on 07.09.2022] (Retrieved from https://chrome.google.com/webstore/detail/phishdetector-true-phishi/kgecldbalfgmgelepbblodfoogmjdgmj)

[18] Digitaldream: Cascaded Phish Detector. [Accessed on 07.09.2022] (Retrieved from https://chrome.google.com/webstore/detail/cascaded-phish-detector/pfngencjknacaakdekdhfjgoalpjnjni)

[19] Alam, M.N., Sarma, D., Lima, F.F., Saha, I., Ulfath, R.-E.-., Hossain, S.: Phishing attacks detection using machine learning approach. In: 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 1173–1179 (2020). https://doi.org/10.1109/ICSSIT48917.2020.9214225

[20] Gowtham, R., Krishnamurthi, I.: A comprehensive and efficacious architecture for detecting phishing webpages. Comput. Secur. **40**, 23–37 (2014). https://doi.org/10.1016/j.cose.2013.10.004

[21] Varshney, G., Misra, M., Atrey, P.K.: A phish detector using lightweight search features. Computers & Security **62**, 213–228 (2016). https://doi.org/10.1016/j.cose.2016.08.003

[22] Internet, A.: Statistics About Website Traffic. [Accessed on 07.09.2022] (Retrieved from https://www.alexa.com/)

[23] W3Counter: Browser & Platform Market Share November 2020. [Accessed on 07.09.2022] (Retrieved from https://www.w3counter.com /global-stats.php)

[24] Mohammad, R.M.A., McCluskey, L., Thabtah, F.: Phishing Websites Data Set. Data retrieved from UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/phishing+websites (2015-2020)