Real-Time Client-Side Phishing Prevention

Giovanni Armano

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 10.08.2016

Thesis supervisor:

Prof. N. Asokan

Thesis advisor:

Dr. Samuel Marchal

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Giovanni Armano

Title: Real-Time Client-Side Phishing Prevention

Date: 10.08.2016 Language: English Number of pages: 8+98

Department of Computer Science

Professorship: Secure Systems

Supervisor: Prof. N. Asokan

Advisor: Dr. Samuel Marchal

In the last decades researchers and companies have been working to deploy effective solutions to steer users away from phishing websites. These solutions are typically based on servers or blacklisting systems. Such approaches have several drawbacks: they compromise user privacy, rely on off-line analysis, are not robust against adaptive attacks and do not provide much guidance to the users in their warnings. To address these limitations, we developed a fast real-time client-side phishing prevention software that implements a phishing detection technique recently developed by Marchal et al. [MSSA16]. It extracts information from the visited webpage and detects if it is a phish to warn the user. It is also able to detect the website that the phish is trying to mimic and propose a redirection to the legitimate domain. Furthermore, to attest the validity of our solution we performed two user studies to evaluate the usability of the interface and the program's impact on user experience.

Keywords: phishing, security, prevention, privacy

Preface

This thesis is made as a completion of the master education in Computer Engineering and it has been developed during my exchange period at Aalto university.

Several people have contributed to this work. I would therefore thank my supervisor N. Asokan and my advisor Samuel Marchal for their time, valuable input and support.

Furthermore, I would like to thank Tommi Gröndahl from University of Helsinki for his help in the management of the user studies.

Finally, I would like to thank my friends for their constructive comments to the thesis.

Otaniemi, 10.8.2016

Giovanni A.

Contents

List of Tables

List of Figures

List of Acronyms

API	Application programming interface
APT	Advanced Packaging Tool
CSS	Cascading Style Sheets
CSV	Comma-separated values
DOM	Document Object Model
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
FQDN	Fully Qualified Domain Name
HREF	Hypertext REFerence
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IT	Information technology
JSON	JavaScript Object Notation
mld	main level domain
MSI	Microsoft installation package
NCD	Normalized Compression Distance
OCR	Optical Character Recognition
OS	Operating System
OSM	Online Social Media
PDS	Phishing Detection Script
RAM	Random-access memory
RDN	Registered Domain Name
ROC	Receiver Operating Characteristic
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TF-IDF	Term Frequency–Inverse Document Frequency
TLS	Transport Layer Security
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
URL	Uniform Resource Locator
XML	Extensible Markup Language

1 General Introduction

1.1 Motivations

The origin of the practice of tricking users to steal personal or credit card information could be placed in the first days of computer networking. However, the word *phishing* was used for the first time 20 years ago to indicate the automation of the scam process that was used to steal passwords on America Online (AOL) customers [Rek11]. This phenomenon has since only increased in scale, sophistication and profit. Techniques have been developed to target multiple brands and entities and reach millions of users at the same time. Furthermore, common people do not think they can be the target of credential, credit card or identity stealing. Phishers manipulate users using social engineering tricks to convince them to disclose their personal information. They exploit the lack of technical knowledge of users and the lack of authentication methods between server and client. Phishing usually relies on a large set of websites having a short lifetime and being able to make a high profit in the first hours after their deployment. For these reasons counteracting this threat is complex and important. Current phishing prevention techniques are implemented as browser toolbars [CLTM04, Net] or through the use of blacklisting systems [goo, mic]. The former can be installed to provide information about the legitimacy of a webpage. The result is based on information extracted from the webpage, gathered from the browsing history or collected from a centralized database. Databases are used to aggregate data such as users' feedback or to rank websites according to the traffic they generate. This approach may raise privacy concerns, because the browsing history can be used for profiling users. Security toolbars, while providing a clear visual assessment of the legitimacy of a webpage, have a high rate of false positives and identification performances lower than 90% [ZECH06]. Furthermore, in case of phishing, they provide passive warnings that users are likely to ignore. Systems based on blacklists, such as Google Safe Browsing, are directly integrated into browsers. Together with a low false positive rate, they provide good performances and active warnings to prevent users' interaction with phishing webpages. However, systems based on blacklist present a flaw related to the time needed for their update. Crawlers are used to reduce this delay, as they are built to scan the web and look for malicious websites. Unfortunately this approach is subject to adaptive attacks that serve different content depending on the browser user agent, location and time in which the request is performed.

To address these drawbacks we introduce a new tool in the battle against phishing. This work is based on a new phishing detection technique developed by Marchal et al. [MSSA16] that is able to identify phishing websites and the target they are trying to mimic. It is developed to perform a client-side analysis that relies only on the information contained in the page. This allows us to avoid adaptive attacks and remove the delay for blacklist updates. It performs the computation in real-time and, in case of phishing, prevents users' interaction by an active warning. It does not rely on browsing history or centralized information and does not share information of any kind with centralized databases and, is thus, privacy friendly. Moreover, it has a

remarkably high accuracy (>99%) and a low false positive rate (<0,1%).

1.2 Contributions

This work has as main purpose the implementation of the phishing detection technique described by Marchal et al. [MSSA16] that has been published in the 2016 IEEE International Conference on Distributed Computing Systems (ICDCS). Part of the work presented in this thesis has been published in the same conference [AMA16]. The design and the evaluations of the user interface have been performed with the collaboration of Tommi Gröndahl from the University of Helsinki. He provided advice on the design of the cognitive walkthrough and the user studies, and summarized the collected data.

We claim the following contributions in the development of a client-side phishing prevention software:

- providing real-time protection and protects the privacy on the user since it is implemented completely client-side (Section 5).
- providing a user interface, built on the recommendation of several works that includes intelligible information (Section 5.5.1).
- presenting a warning message that includes the redirection to the target of the phish. This is a functionality that has never been included before in a warning message (Section 5.5.1).
- deployable in multiple operating systems and in the major web browsers (Section 6.1).
- performing data extraction and websites classification in a median time lower than 200 milliseconds (Section 6.2.1).
- using a smaller amount resources compared to the technique it is based on (Section 6.2.2).

1.3 Structure

We now present the organization of this work and the information contained in each section. **Section 2** gives a description of what phishing is, the origins of such attacks and the common vectors used to perform it. It then gives an overview of machine learning principles and presents the phishing detection techniques used in this work. A technical background introduces the reader to the technologies used. **Section 3** gives an overview of the state-of-the-art in phishing prevention and phishing detection together with a summary of recent studies related to phishing warning design. **Section 4** reviews the drawbacks of previous techniques and introduces the problems we address in this work. **Section 5** presents the core of this work, a real-time client-side phishing detection system that addresses several limitations of previous solutions. It also presents the phishing warning and the

different messages that are displayed. It illustrates the evolution of the interface design thanks to a cognitive walkthrough and compares the final version of the messages with other warning messages. **Section 6** evaluates the solution taking into consideration the requirements it must meet. Presents the two user studies that were conducted to evaluate the effectiveness of the user interface and the impact on the user experience. We conclude the document in **Section 7**.

2 Background

Section 2.1 introduces the definition, origins and evolution of phishing. We explain how the attackers adapt their techniques to the defences that have been adopted during the years and the different vectors that can be used to fool users. Section 2.2 is focused on presenting some basic principles of machine learning. We describe the classification techniques that are used in this work and some metrics to evaluate them. Section 2.3 presents the phishing detection technique that is the basis of this work, together with an evaluation of its performance. Section 2.4 gives the necessary technical background about the technologies used to fully understand the remaining part of this work.

2.1 What is Phishing

Together with the increase of Internet usage for everyday tasks, the amount of sensitive and personal data stored on-line has increased. The large number of services nowadays available on-line helps the users in various aspects:

- **Internet banking:** Having access every day at any time to the information regarding the personal bank account may help to monitor payments and the balance in real-time. The opportunity to make money transfers from the laptop saves the time needed to go to the bank office in person.
- **On-line shopping:** Through e-commerce portals, being able to choose between a large variety of products that can be bought providing credit card credentials and an address.
- **Working from home:** Getting access to sensible data and documents directly from home using repositories protected by user name and password.

The possibility to use these services through webpages helps people to save time, get access to a larger set of goods and be more productive at work. These approaches imply trust between the two parties: user and server. Proving the legitimacy of the former is a trivial problem, while this is not the case for the latter.

Crooks may take advantage of naive users to create phishing webpages, which mimic other websites to gather information and personal data from users. The purpose of this practice varies from selling the data to a third party, impersonating users and using credit card credentials to buy goods.

The term *phishing* appears the first time back in 1996 to indicate the attack in which scammers pretended to be America Online employees to send messages to ask customers for confidential information. Phishing perpetrated via email can target millions of people at the same time or can be directed against a single company or person. We refer to the last scenario with the term *spear phishing*. This kind of attack is more sophisticated and less generic compared to a normal phishing email. Phishing emails usually contain indicators such as logos, signatures or an addresses that are used to make users feel confident about the legitimacy of the message. Spear

phish go beyond that and insert specific and personal information regarding the user such as transaction that they have performed recently or other references that are strictly related to the target that can be gathered using social engineering. These attacks are usually performed against employees with high clearance or CEO. Due to the high profile of their target and the amount of money that can be gained from a single person, the term *whaling* is used to refer to these attacks.

One reason why people keep falling for phishing is that naive users do not recognize which information is important to attest the legitimacy of a website [DTH06]. Even if users read URLs pointing to a webpage, they might not be able to understand the information contained in it. URLs like *facebook.com/login, login.facebook.com, facebooklogin.com* or *login-facebook.com* might all be interpreted as legitimate Facebook login pages. While the first two are under the control of *facebook.com*, and only the first is the actual address of the login page, the third and the fourth have different domains that belong to different entities. Users are attracted to the look-and-feel of the page without knowing that they can be easily copied. Even expert users can be tricked by more sophisticated swindles. To mitigate this problem, browsers provide security indicators such a green lock in case of valid SSL certificate. The connection established using a certificate allows an encrypted end-to-end communication and has a double function: it prevents everyone except the server to read the data sent by the client and attests to the ownership of the domain. Though, not every legitimate website has a certificate, because it costs, and some malicious websites may have it, because it can be seen as an indication of trust.

While understanding if a website is legitimate is not an easy task, tutorials and tools that allow the creation a phishing websites are not hard to find, it is sufficient to search on a search engine such as Google to find several of them [cre]. After a phishing webpage is built, phishers may place it under some free web hosting service or on a compromised third party machine. Whatever the location is, moving the phishing website to another location is easy for the phisher and this frustrates attempts to stop their activity. Taking down a phishing website may not be an easy task, depending on several factors [bra]. Furthermore, the lifetime of phishing websites is in the order of hours or days [APW14, bra]. It is therefore imperative to operate fast and not allow the phisher to have a time window in which he can operate undisturbed.

2.1.1 Phishing Vectors

Several techniques can be used to perform phishing attacks. They rely on combinations of social engineering strategies and technical knowledge to make these kinds of swindles effective. Different vectors are used to spread URLs that point to phishing websites or to steal personal information and money. We now review some widely used techniques that phisher might apply.

- **Emails:** This is probably the first vector used to perpetrate phishing attacks. It consists of sending an email from an address that pretends to belong to a bank or a company in which the user is asked to update an account, confirm a payment or do other tasks that involve the disclosure of personal information.

During the years, companies and brand have instructed their users not to send personal information via email to reduce this phenomenon. Phishers responded by changing the structure of the emails and by including links that redirect to phishing websites that mimic the targeted brand. The excuses are similar: some privacy issue is detected or the limitation of the functionalities of a certain account is reached if the provided information is not updated in a small amount of time. Other approaches consist of the attachment of files that have to be downloaded and contain malware used to collect personal data.

- **Online Social Media (OSM):** Phishers' strategies have a continuous evolution. With the advent of social media they started to use Facebook and Twitter to distribute links to phishing websites. This is very effective due to the speed at which content can be spread in the network and it is hard to detect due to the obfuscating techniques used. In Twitter, in particular the URLs, both malicious and legitimate, are usually processed with a shortener service because of the maximum length (140 characters) of a single message. For the same reason, discriminating between legitimate and phishing URLs is not an easy task.

- **Fake websites:** This category contains websites that are crafted to recreate the look and feel of legitimate websites while their only purpose is to steal information. Typical targets are banks, social networks, payment websites, retail and other known brand in general. The URLs that point to these websites may rely on typosquatting [AJPN15], that is the intentional registration of main level domains (*mlds*) that are the misspelled version of popular websites. Or it can be totally different, and contain the target *mld* in a free part of the URL. Links to these kinds of webpages can be spread using vectors such as *email* or *OSM*. When users insert their credential (username and password), they are usually redirected to the legitimate websites, and therefore do not realize that they have been hijacked.

- **Content injection:** This consists of the change of a part of a legitimate website due to the injection of malicious code. This loaded content can be used to display additional forms in which personal information is requested. When the user inserts the data, the script sends it to the phisher using email or other methods. This approach is also used to perform cross-site scripting, an attack in which server side vulnerabilities are exploited to execute javascript code on the victim's webpage.

2.1.2 State of phishing threat

In the first quarter of 2016, reports of phishing attacks show that it is a threat that continues to grow [AS]. The trimestral report of the Anti-Phishing Working Group (APWG) shows how the number of unique phishing websites is increasing from Deceber 2015 (Fig. 1). These numbers are composed of the sum of the unique base URLs of phishing websites. However, every phishing website might advertise thousand of URLs.

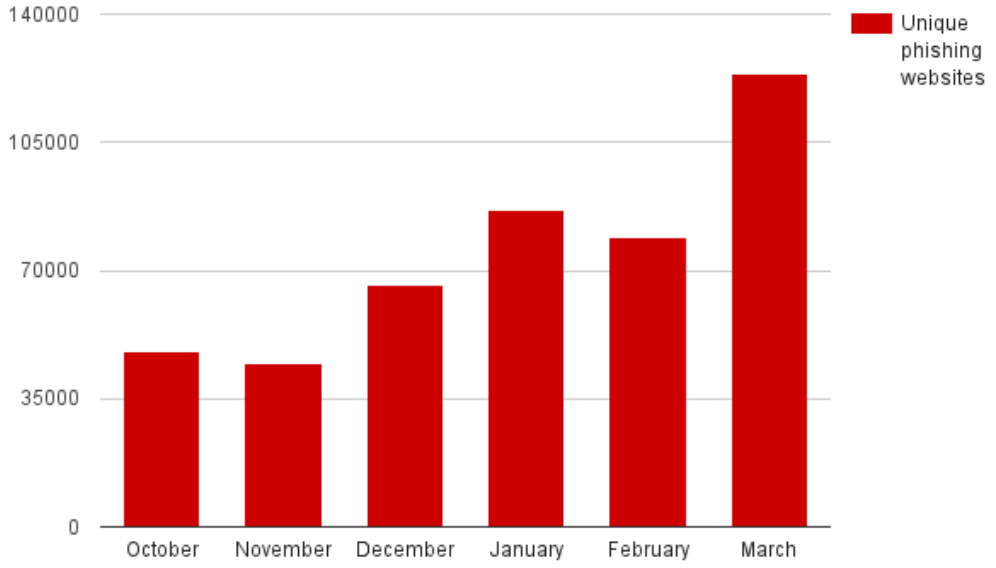


Figure 1: Unique phishing sites detected October 2015 - March 2016 (source: APWG)

Figure 1 depicts that the number of phishing websites has increased by 250% from the last quarter of last year, while the number of target brands fluctuate between 431 ad 418. The number is even more alarming if compared to the state of phishing attacks back 10 years ago [APW06], where the number of unique phishing attacks was less than 30,000 and the number of targeted brands was 130. The sectors that are most targeted by this kind of attack are Retail/Services, that alone consist of 42,71% of the recorded attacks as shown in Figure 2. Another big portion (33,41%) is occupied by the Financial and Payment service sector. However, as we can see, the sectors that are targeted by phishing are several and diverse, from Financial and Government to Social networks and Education.

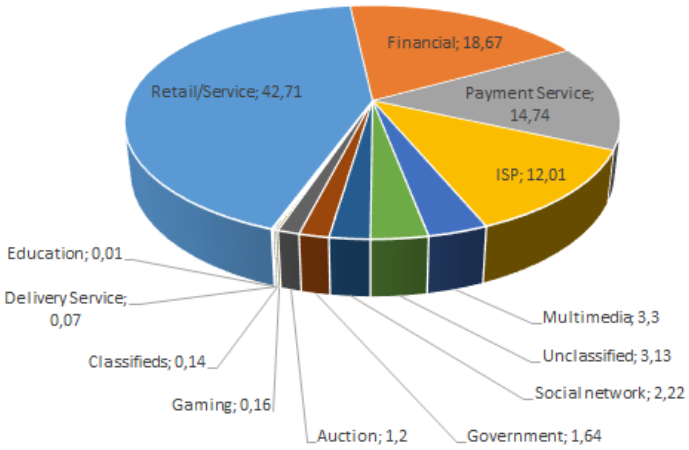


Figure 2: Most targeted industry sectors in the 4th quarter of 2015

The cost related to phishing is not limited to the direct money loss. It also impacts the reputation of the organization, productivity loss and cost to the maintenance and recovery of IT structure. The loss related to phishing and identity theft has been estimated between 453 million dollars [Tea15] and 54 billion dollars [Str12].

2.2 Machine learning

Machine learning is the study and modelling of learning processes implemented on computers [CMM83]. To disclose patterns, it uses statistical models in combination with optimization methods and traditional algorithms from computer science.

There are several types of learning problems, and we can discriminate between two major types of learning depending on the availability of labeled data. In case labels are available, the problem is called supervised learning. Supervised learning makes use of a classifier taking as input a vector of discrete, continuous, or mixed feature values and outputting a single discrete value, the class. Its potential resides in the building of a model and finding a pattern from a training set taken as an input in order to make predictions of new instances without strictly following static instructions. If labeled data is not available, we talk about unsupervised learning. This approach addresses a class of problems in which the goal is to determine how the data is organized. The output is based on similarities within a subset of the analyzed data. These two categories are not exhaustive: there are other methods, i.e. on-line learning [FRGBMR$^+$13], in which the model is updated every time new data becomes available. In the context of this thesis, we used a supervised learning approach that we present in this section together with machine learning algorithms and evaluation methods.

2.2.1 Supervised learning

In supervised learning the starting point is a set of raw data extracted from a certain source. In our particular case we have used data that has been collected using a web crawler and subsequently labeled into two classes. The classes of interest for us are phishing websites and legitimate websites. Based on the data source, we perform the feature extraction. This process is in charge of collecting a measurement (numerical values) from the sources that the machine learning method is able to process. Performing a good feature extraction, combined with the choice of good classification method, is the key aspect to obtaining good performance in prediction [Hal99]. When the feature set is ready, the model can be trained.

Once the model is trained, a second different set, called the test set, can be used as input to the classifier to predict the label of the instances.

2.2.2 Tree classifier

Decision tree classifiers process data based on a certain number of input variables, exploiting a decision tree. They navigate in the tree until they reach one leaf that contains the decision class [BFOS84]. Depending on what kind of a result is returned as output, we can divide them in two classes:

- **Classification trees:** a tree that has a finite set of target values, such as classes.
- **Regression trees:** a tree that can outputs a continuous set of values, such as real numbers.

In machine learning, each classifier can be described by a set of *parameters* and *hyperparameters*. In case of a tree classifier, *parameters* indicate the variables and their values are used in each node to determine the navigation to the leaf. The *hyperparameter* is the maximum depth of the tree, identified as how many nodes the process should go through before reaching a leaf containing the target value. On one hand, the deeper the tree is, the more accurate the prediction of the training set results are. On the other hand, a too fine-grained division can lead to overfitting the training set and the classifier will lose generalizability (i.e. the capability to be applied to a larger data set).

Overfitting is a common mistake in supervised learning, due to excessive tuning of a particular set of features that leads the classifier to take into account too many peculiarities of the training set, losing the ability to perform well on a different set.

Tree classifiers are usually adopted as weak learners in more complex classifiers, called ensemble classifiers, such as Random Forest [Bre01] or Gradient boosting [Fri02], in which a large number of them with relatively small depths (3 is a common parameter) is used in parallel to extract a more accurate result.

2.2.3 Ensemble Classifiers

Ensemble Classifiers are based on the idea that we should train different *weak classifiers* and combine their solutions, i.e. by majority vote, to discriminate between the classes of interest [RT14]. A *weak classifier* is a classifier that has a low *accuracy*.

To understand how to establish when a classifier is weak we can consider the following example. We suppose a two class discrimination problem where we have the same number of instances in both classes (balanced dataset). Even if we assign a class to the data that we get by flipping a (unbiased) coin, we can expect to guess correctly half of the time for a large enough sample size. This means that for a problem like this we can consider as weak a classifier that performs slightly better than a random choice.

The results of the different classifiers are evaluated by a majority vote in order to extract the right decision. To do so, m classifiers are chosen, with m being odd to avoid equality. Then, if we decide to assign the result of a single classifier to +1 or -1 depending on which label is chosen by each classifier, the problem is reduced to finding the result of the combination of the votes $f_1, ..., f_m$ using:

$$majority\ vote\ decision = sign\left(\sum_{j=1}^{m} f_j(x)\right)$$

Two assumptions are required:

- Each classifier assigns a certain label to a sample with a certain probability.

- Each classifier is stochastically independent from others, meaning that the occurrence of a certain result in a classifier does not affect the results of the others.

The compound probability of a positive outcome is the product of the probabilities of each weak classifier. Depending on the number of classifiers that we integrate into our solution we can increase the probability of success.

However, these considerations are valid only if we consider stochastically independent classifiers. Unfortunately, due to the use of a single training set, dependencies between classifiers are introduced. To overcome this problem, there are various strategies to simulate independence by modifying the training data in certain ways such as deterministic strategies, weighing data or data randomization [Fri02].

2.2.4 Gradient boosting

Gradient boosting is an ensemble classifier method used for regression and classification problems. Boosting [Sch99] is a fitting procedure that improves the prediction of binary outcomes from weak models using weighted ensembles of base-learners.

Gradient Boosting consists of an iterative process where the starting point is an initial estimation of a prediction function. During each iteration, the function is improved by fitting a new base-learner (decision tree) to the negative gradient of a pre-specified loss function. Gradient boosting improves model accuracy by selecting variables and weak models at each iteration. At the end of the iterative process, the final prediction function combines the best weak learners and variables as parts of the model.

Gradient Boosting predicts the class of an unknown instance by computing values between [0, 1] that give the confidence value of the instance belonging to a given class. A discrimination threshold is used to discriminate between the two classes. It has the default parameter of 0,5, in the middle of the possible confidence values, but can be tuned to obtain better classification performance.

In order to reduce the statistical dependency that is generated by the training process of the different weak learners over the same training set, a variation called stochastical gradient boosting was introduced by Frieman [Fri02]. This approach has the main advantage of choosing a random subset of the training data for the creation of the tree in different classifiers, reducing tree correlation.

2.2.5 Evaluation metrics

In describing an algorithm for pattern recognition, it is important to introduce some metrics to be able to quantify the value of the results. The three concepts of *precision, recall* and *accuracy* are based on two different types of error that can be made when classifying a data set. In the presence of labeled test data (i.e. the test set) we can determine the division of the data based on the correctness of the classifier prediction as shown in Table 1.

These terms will have the following meanings:

		Actual class	
		Legitimate	**Phishing**
Prediction	**Legitimate**	True negative	False negative
	Phishing	False positive	True positive

Table 1: Error matrix

- **True positive (TP):** indicates a phishing website identified correctly
- **False negative (FN):** indicates a phishing website identified as legitimate
- **False positive (FP):** indicates a legitimate website identified as phishing
- **True negative (TN):** indicates a legitimate website identified correctly

These metrics can also be represented as the following rate:

- **True positive rate (TPR):** TP/(TP+FN)
- **False negative rate (FNR):** FN/(TP+FN)
- **False positive rate (FPR):** FP/(FP+TN)
- **True negative rate (TNR):** TN/(FP+TN)

Based on these values we can estimate the metrics as:

- **Precision:** TP / (TP+FP) is the number of phish correctly identified as phish by the classifier divided by the total number of websites labeled as phish by the classifier
- **Recall:** TP / (TP+FN) is the number of phish correctly identified as phish by the classifier divided by the total number of phishing websites
- **Accuracy:** (TP + TN) / (TP + TN + FP + FN) is the proportion of true results (both TP and TN) among the total set of tested data.

Additionally, we use another indicator for our data presentation:

- F_1**-score:** is a measure for test accuracy calculated as the harmonic mean of Recall and Precision [GG05]

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

In machine learning, in order to present data derived from binary decision, receiver operator characteristic (ROC) curves are commonly used. They show how the true positives rate (TPR) and the false negative rate (FNR) change in relation to the variation of the discrimination threshold of the classifier. The TPR describes how

many phishs were correctly identified over the phishing websites present in the test set and the FNR describes how many legitimate websites were detected as phish over all the legitimate websites present in the test set. ROC analyses are used to detect optimal values of the discrimination threshold independently of the distribution of the class.

2.3 A phishing detection technique

Some phishing detection techniques [WRN10, MSSV09] are able to identify phishing webpages with an accuracy higher than 90% and at the same time have a very low rate of misclassified legitimate webpages, i.e. less than 0,1%. These techniques are machine learning based. The algorithms used to classify phishing websites, so far, rely on millions of static features, most of them using the bag-of-words approach [MSSV09]. The bag-of-words approach is a way to represent a text as a multiset (bag) of the words that it contains. It can be used to summarize a website as an unordered collection of words extracted from different sources such as text and URLs. This led to two major weakness:

- A huge amount of labeled data is required to train the model. This data is difficult to find because the uptime of phishing websites is small [APW14], they need to be manually checked to be correctly labeled, and even in this case it can be complicated to understand the real nature of the websites (poorly known brands or services).
- It is language and brand dependent and is not effective in detecting phishing attacks against new targets.

The technique that we are going to describe in this section has been presented at the IEEE International Conference on Distributed Computing Systems (ICDCS) 2016 and forms the basis of this work [MSSA16]. The text contained in this section refers to one presented in that work and it is reported here to underline its functioning and the differences introduced by this new work. The technique described in Marchal et al. paper is built of two components. The first one is a *phishing classification system* that, relying on a small sets of data all collected from the webpage, is able to estimate whether a websites is legitimate or a phish. The second one is able to perform *target identification* based on the information extracted from the webpage and return the websites that the phish is trying to mimic. This phishing detection system is based on two observations: (a) although phishers try to make a phishing webpage look similar to its target, they do not have unlimited freedom in structuring the phishing webpage; and (b) a webpage can be characterized by a small set of key terms; how these key terms are used in different parts of a webpage is different between legitimate and phishing webpages.

Based on these observations, this approach has several notable properties: it requires very little training data, scales well to much larger test data, is language-independent, fast, resilient to adaptive attacks and implemented entirely on client-side.

2.3.1 URL structure

To be able to identify a particular webpage, a Uniform Resource Locator (URL) is used. In its structure, several parts can be identified, and separated, based on their functions and characteristics.

$$protocol : //[subdomains.]mainleveldomain.ps[/path][?query]$$

The Fully Qualified Domain Name (FQDN) is used to identify the server that hosts the page and is composed of the subdomains and the registered domain name (RDN). While the former is freely editable by the website owner (or the phisher), the latter is fixed and registered to a domain name registrar. The RDN can further be divided into the main level domain (*mld*) and the public suffix (ps). The part of the URL that follows the RDN is composed of the path, which is used to locate the resource in the server, and the query, which provides additional parameters. These two are not constrained and can be edited. We are going to refer to the set composed of *subdmains, path* and *query* as *free URL* to underline that phishers are able to freely modify them to obfuscate a URL.

2.3.2 Data sources

To extract the features needed for the classification and target identification, the useful data sources that are available on the web browser have been identified as the following:

- **Staring URL:** the URL given to the client to reach the webpage. It can be distributed in an email, instant messages, texts, websites, etc.
- **Landing URL:** the last URL indicating which content the user is actually displaying in the web browser. This is the URL present in the address bar when the webpage is completely loaded.
- **Redirection chain:** a list that contains all the URLs through which the user has been redirected between the *starting* and the *landing URL*
- **Logged links:** the list of all the URLs that have been requested from the page in order to be loaded, including stylesheets, HTML and PHP pages, images and any other kind of content logged by the browser.
- **HTML:** the source code of the page combined with these contained in IFrames and Framesets.
  - **Text:** the text extracted displayed to the user, located between the <body> HTML tags.
  - **Title:** the title displayed on the tab of the browser, located between the <title> HTML tags.
  - **HREF links:** the list of the links that <a> HTML tags refer to.
  - **Copyright:** the copyright, if present, of the page.
- **Screenshot:** a snapshot of the visualized webpage.

2.3.3 Phisher limitations

Look and feel are key aspects in fooling users in phishing attacks [DTH06]. In order to create webpages that are similar to the original [XH09], phishers load content such as images and HTML code directly from the target webpage and integrate it with their portion of code [PD06]. The content loaded from external sources can include outgoing links to the legitimate webpage, together with the URL from which these resources have been downloaded. Furthermore, keywords referring to the target can be identified in several data sources, such as title, text and linksl [LMF10]. However, if the phisher wants to embed this kind of content to his webpage, there are limitations that he can not overcome.

Marchal et al. divided the data sources described in Section 2.3.2 based on the level of *control* and *contraints* that the phisher must comply to.

Control: Creating a page that embeds content from another website leads to the download of sources from other registered domain names (RDNs), that are collected as *logged links*. Furthermore, the incorporated part of the code will probably have *HREF links* pointing to the target of the phish. These data sources are divided to *internal* and *external* according to their RDNs. For this discrimination we can assume that RDNs contained in the *starting URL*, *landing URL* and *redirection chain* are under the control of the page owner and so are marked as *internal*. The remaining URLs, with other RDNs, are considered outside the control of the page owner and, because of that, marked as *external*.

Contraints: As described in Section 2.3.1, some parts of a URL can be freely modified. Phishers can use them in order to insert text that might fool the user. However, these subterfuges can not be used to modify the RDN that is registered. Based on this assumption they identify the RDN as a constrained part that can be casted in a set of features to model the phisher's limitations.

2.3.4 Feature selection

We now describe how the feature set was selected and why.

To be able to introduce a new phishing detection technique comparable to the state-of-the-art, the requirements of an effective approach were analyzed. These fall into three main concepts: *generalizability* and *adaptability*.

Taking these considerations into account, Marchal et al. selected 212 features that cover the constraints and the degree of control present in the phishing webpage creation process. These features can be divided into 5 categories as shown in Table 2.

URL: these features are based on nine different statistical features related to the URL composition (Table 3).

While feature 2 is meant to analyse whether the strings contained in the path or query look like a domain name, features 3 to 8 try to detect if any obfuscating techniques, such as long URLs built on several terms, have been used. Feature 9 is related to the Alexa list, a rank of the most visited domains.

Alexa Internet, Inc. [Inc] is a company that provides commercial web traffic data and analyses. It is possible to download from their website a list of the top one million websites ranked by Alexa based on a 1 month time window in which the

Name	Count	Type
f1	106	URL
f2	66	Term usage consistency
f3	22	Usage of starting and landing mld
f4	13	RDN usage
f5	5	Webpage content
fall	212	Entire feature set

Table 2: Feature sets

#	Description
1	protocol used (http/https)
2	count of dots ”.” in FreeURL
3	count of level domains
4	length of the URL
5	length of the FQDN
6	length of the mld
7	count of terms in the URL
8	count of terms in the mld
9	Alexa ranking of the RDN

Table 3: URL reatures

traffic data is collected. The list used in this scenario is a pre-downloaded copy containing domain names.

The 106 features of the first set are a combination of:

- The 9 features computed on *starting* and *landing URL*
- The mean, median and standard deviation computed on features from 3 to 9 on *logged links* an *HREF links* (both internal and external)
- Feature 1 computed on the 4 sets previously described

Term usage consistency: as pointed out before, in order to make the victim believe that the phishing webpage is the target, phishers incorporate a part of its original content. This set of features is based on the assumption that a page crafted with information extracted from another source leads to a distinctive distribution of the terms in different locations of the page.

Taking into account only the words with a length greater than or equal to 3, supposing T to be the set of all the terms; $T_s = \{t_{i\epsilon(1;m)} \ \epsilon \ T\}$ is extracted from a source S with t_i occurring with p_i probability. They define the term distribution D_s of S as the set of m pairs $(t_i, p_i) \ \epsilon \ T \times]0, 1], \ i \ \epsilon \ \{1; m\}$. The sources from which the different distributions are extracted are listed in Table 4.

To find some inconsistency, the different constrained, unconstrained, controlled and uncontrolled parts are compared. Without taking into account *Dcopyright* and

Distribution	Data source
Dtext	Text
Dtitle	Title
Dcopyright	Copyright notice
Dimage	Webpage screenshot
Dstart	Starting URL - FreeURL
Dland	Landing URL - FreeURL
Dintlog	Internal logged links - FreeURL
Dintlink	Internal HREF links - FreeURL
Dstartrdn	Starting URL - RDN
Dlandrdn	Landing URL - RDN
Dintrdn	Internal links (HREF and logged) - RDN
Dextrdn	External logged links - RDN
Dextlog	External logged links - FreeURL
Dextlink	External HREF links - FreeURL

Table 4: Term distribution

Dimage, 66 features (12 * 11 / 2) are extracted to represent the similarity of pairs of sources by pairwise computation of the Hellinger Distance [CY00] between their distribution. This metric allows an estimation of the similarity of two probabilistic distributions, returning a value between 0, if the distributions are the same, and 1 for complete dissimilarity.

Usage of starting and landing mld: while legitimate websites are likely to register a domain name that reflects the brand or the service that they offer (i.e. the mld for Bank of America is *bankofamerica.com*), phishers often use a domain that has no relation with the target [XH09]. For this reason, webpages built over content extracted from other sources will have less links that point to their starting and landing mld compared to the legitimate ones. The 22 features defined to shape these characteristics are the following:

- 12 binary features are set to 1 if the starting/landing mld appear in *Dtext, Dtitle, Dintlog, Dextlog, Dintlink* or *Dextlink*.
- 10 features are the sum of probabilities from terms of *Dtitle, Dintlog, Dextlog, Dintlinkand* and *Dextlink* that are substrings of starting/landing mld.

RDN usage: These features are based on the expectation that, on a legitimate website, internal *registered domain names* are more used than external ones. The 13 features of this set are related to the similarity of RDN in starting and landing URL, redirection chain, logged links and HREF links.

Webpage content: the last set consists of 5 features corresponding to the count of *input fields, terms in title, terms in text, images* and *IFrames*.

These has been selected because phishers include a small amount of text in order to avoid text-based detection techniques [RW13]. Input fields are used to collect

as much information as possible from the user and IFrame in order to incorporate content from other sources.

Marchal et al. also underline that all these features that are extracted from the different text sources are not related to any particular language, but only on their relative use in the different locations. The only features that are not extracted directly from the page are the ones related to the Alexa list, which once downloaded is stored as a local copy. The sum of all these sets creates 212 features that, considering their small amount, should not overfit the training data.

2.3.5 Phishing detection system

To compute the collected features and discriminate between legitimate and phishing websites, Marchal et al. used a supervised machine learning approach. A collection of data marked with their own classes (labeled data), is used as a training set for the classifier. The produced model is later used to discriminate between the classes of unlabeled data sets.

The selected algorithm used to build the classification model was Gradient boosting developed by Friedman [Fri02].

2.3.6 Target identification

The target identification component identifies the target website that a phishing webpage is attempting to mimic. The target detection component can help minimize false positives of the phishing detection system. Its analysis is based on the identification of terms that appear in several data sources of the page, called keyterms. The sources that are taken into account are five of the fourteen described in Section 2.3.4.

- Starting and landing URL: $T_{start} \cup T_{startmld} \cup T_{land} \cup T_{landmld}$
- Title: T_{title}
- Text: T_{text}
- Copyright: $T_{copyright}$
- HREF links: $T_{intlink} \cup T_{intlink}$

Three different procedures are used to extract the same amount of keyterm sets:

- **Boosted prominent terms:** are calculated from the intersection of each pair of sources. Every time a word is contained in both of the sources compared, it is inserted in a list with its relative frequency. The final set, with the overall frequency, is sorted and the top 5 terms are extracted.
- **Prominent terms:** are calculated using the same technique as the *boosted prominent terms*, but the intersection between text and HREF links is removed. This is done to avoid websites that contain the same terms in an <a> HTML tag and its HREF link, as many news websites do, from interfering with the target identification with irrelevant words. A maximum of 5 terms are extracted.

- **OCR prominent terms:** are the terms extracted from the optical character recognition process applied to the screenshot of the page (T_{image}). The set extracted is intersected with the five data sources to produce another list of the 5 most common terms.

  The extracted sets of keyterms are used to identify the target. Each of the following steps is based on the assumption that when a search engine is queried for certain keywords, it will not return a phishing website as result. This is supported by two observations: (i) phishing websites that are new did not have time to improve their rank and appear as first results (before the target), (ii) old ones would have already been detected and added to a blacklist or taken down. The target is inferred using the extracted keyterms through the following steps:

  1. Starting from the *boosted prominent terms (bpt)*, the process tries to determine the possible target Fully Qualified Domain Names (FQDNs). Legitimate companies and services try to register domain names that contain their name (i.e. the domain registered for Bank of America is *www.bankofamerica.com*). To do that the main level domains (*mlds*) are extracted from sources such as: *starting and landing URL, logged and HREF links*. Successively, all the possible combinations of the *bpt* (including variants with dots or dashes) are compared with the collected *mlds* to find matches. If a match is found in the *starting or landing URL*, the websites is considered legitimate, otherwise step two will follow.
  2. The set of *prominent terms* is used to query the search engine. The returned *mlds* are compared to the *starting and landing mlds*. In case of a match, the webpage is marked as legitimate, otherwise the process proceeds and the *mlds* returned are stored and will be used for step 5.
  3. The same analysis as in step 2 is performed, but with the *boosted prominent terms* instead of the *prominent terms*.
  4. The same analysis as in step 2 is performed, but using the *OCR prominent terms*.
  5. In case the page has not been validated at any step of the process, the possible targets are returned. These are computed as the top 3 *mld*, based on a descending ranking of the frequency, of each *mlds* that has been returned from the different steps.

2.3.7 Evaluation

This method was evaluated using ground truth gathered from a set of phishing and legitimate websites (Table 5).

The *legitimate* URLs were supplied by Intel Security. The scraped websites were divided in 6 different categories depending on their language.

Set	Name	Initial	Clean
Phish	*phishTrain*	1213	1036
	phishTest	1553	1216
	phishBrand	600	600
Leg	*legTrain*	5000	4531
	English	100,000	–
	French	10,000	–
	German	10,000	–
	Italian	10,000	–
	Portuguese	10,000	–
	Spanish	10,000	–

Table 5: Dataset descriptions

The collected data takes into account websites with different notoriety and traffic. This is shown by the fact that the 65,302 of the 150,000 test URLs had RDNs ranked in Alexa's top 1 million websites.

The *phishing* URLs were obtained through the website PhishTank [Phi], which provides legitimacy evaluation of websites through user's feedbacks. The amount of data that can be collected is strongly dependent on the uptime of the phish. For this reason the scraping was performed in three different sessions that produced as many data sets.

The evaluation was conducted using a learning stage on *legTrain* and *phishTrain* and a prediction on all the other datasets.

The focus of the evaluation was on three different aspects of the performances of the *Phishing detection system* and an evaluation of the *Target identification*.

- **Accuracy, recall and false positive rate:** The results shown in Table 6 were calculated over 6 different languages. We can see that the precision reached for all the languages is remarkably high, 0.95 or 0.98, and the recall, with a value of 0.958, as well. Even the harmonic mean between precision and recall ($F_1 - score$) is significantly high for every language. Hence, the FP Rate is very low: from 0.0005 to 0.004.

Language	Precision	Recall	F1-score	FP Rate	AUC
English	0.956	0.958	0.957	0.0005	0.999
French	0.970	0.958	0.964	0.0036	0.997
German	0.981	0.958	0.970	0.0022	0.998
Portuguese	0.967	0.958	0.962	0.004	0.997
Italian	0.982	0.958	0.970	0.0021	0.998
Spanish	0.982	0.958	0.970	0.0021	0.998

Table 6: Detailed accuracy evaluation for different languages

Considering that in real-word large-scale use cases, machine learning models are applied only if the precision is higher than 90/95% and recall above

50/60% [SSM+12], this method is ready to be applied in multi-lingual business scenarios.

- **Receiver operating characteristic (ROC):** ROC curves are used to show how the false positive rate and true positive rate vary in relation to modification of the discriminating threshold. The results, relative to 6 different languages, are depicted in Fig. 3.

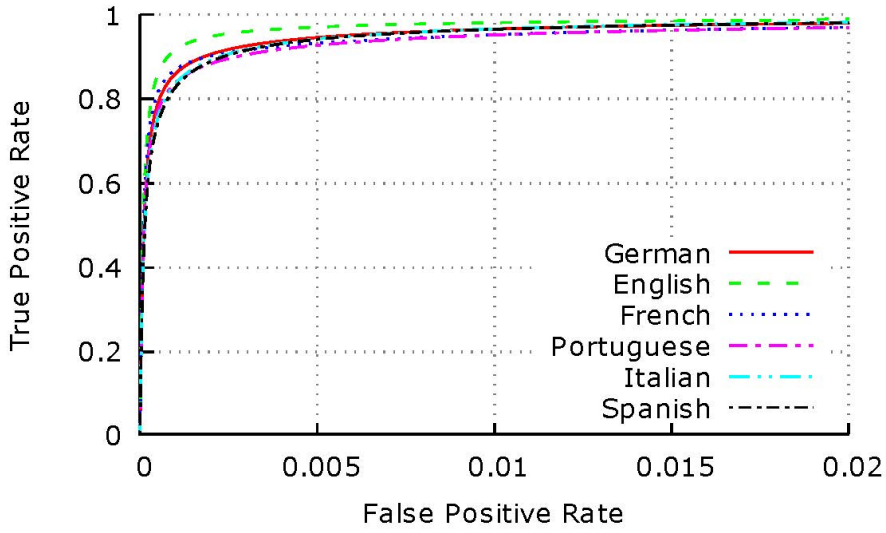


Figure 3: ROC evaluation results from 6 languages

We can see that for an already high value, around 0.9, for the true positive rate (TPR), the rate of false positive (FPR) is below 0.005, which is considered low. As we move the threshold and increase the value of true positive, the false positive rate remains very low. At the remarkably high TPR of 0.98, even if the value of false positive increases, 0,02 of FPR is considered as quite low.

- **Target identification evaluation:** To test the efficacy of the target identification, Marchal et al., used the phishing websites of *phishBrand*. For each website a manual evaluation of the target was performed to gather ground truth. 17 out of 600 pages have an unknown target. For these pages the information present did not provide any hints about the target. Maybe it was mentioned in the email or post that was used to attract the victims.

Targets	Identified	Unknown	Missed	Success rate
top-1	526	17	57	90.5%
top-2	558	17	25	95.8%
top-3	567	17	16	97.3%

Table 7: Target identification results

The evaluation took into account the likelihood of the target being in the top-1, top-2 or top-3 results given as output of the identification process. We identify

as a success the presence of the RDN of the target in the top-N RDNs identified. Table 7 shows how the success rate of the top-1 results is above 90%. If the scheme is configured to return 3 results, then the percentage rises to 97,3%.

2.4 Technical background

This section is intended to give a general overview of the technologies and the framework used. It also puts the focus on the aspects that are strongly related to their usage in this work.

2.4.1 Javascript

JavaScript [Fla06] is a high-level, untyped and interpreted programming language that is well-suited to object-oriented and functional programming styles and finds its main usage in Web programming. Nowadays every website makes use of it to specify several page behaviors such as rendering page content, communicating with the server after a webpage is loaded, playing games and more. It is part of the triad of technologies, with HTML and CSS, that are used by every Web developer.

It should not be confused with Java, with which it has only a superficial syntactic resemblance. From its birth as a scripting language, it has now become an efficient general purpose language.

The core of Javascript includes the support for working with text, arrays, dates and regular expressions while the functionality for the input and output are supplied by the interpreter that is used to run it. Every modern web browser works as a Javascript interpreter and through the supplied console, programmers can interact directly with the content displayed in the page.

2.4.2 Add-on

Add-ons, or browser extensions, are software components that add specific features to a web browser. All the popular browsers are able to integrate them to provide more functionalities over the default ones. These kinds of plug-ins are developed in JavaScript and executed concurrently with the browser. Add-ons are collections of files that are packaged with every resource that is needed for their installation and execution. Besides resources such as images, icons and videos, they contain the script that constitutes the core of the application and other files regarding configuration and permissions.

The script can be divided into two main categories, the *content script* and the *background script*, depending on their task, lifetime and privilege.

The content scripts are executed each time a webpage is loaded when the programmer triggers it, before, during or after the page rendering. They are able to interact with the webpage to read, write and modify its content and behavior.

The communication between the content and background script is possible through the use of a messaging system. In this way the content script can indirectly get access to the browser APIs.

The background scripts are meant to maintain a long-term state of the application or perform long-term tasks that run independently from the lifetime of a single tab of the browser. They are loaded and executed when the application is installed and they stop their execution when the add-on is disabled or uninstalled. They have access to the API provided by the browser as long as they have the required permission declared in the manifest file.

The information contained and required in the manifest file depends on the specification of each web browser, but they usually contain: the path to the content and background scripts, the resource folders, the permission that has to be granted to the add-on, and the information about the developer together with the support and compatibility of different browser's versions.

2.4.3 Crossrider

Crossrider [Cro] is a framework that can be exploited to build add-ons that can be executed on different web browsers. It allows cross-browser deployability and through APIs provides an abstraction layer that uniforms the different function calls.

So far it supports Internet Explorer, Google Chrome, Mozilla Firefox and Safari. It is in continuous development but currently not all the functionalities are available for every browser.

It allows the programmers to write a single version of the code that is easier to maintain in comparison to a different ones for each web browser.

2.4.4 Python

Python [VRD03] is an object-oriented programming language. It is used in numeric programming, artificial intelligence, image processing, biology modeling and other fields. It makes use of indentation as a syntactical structure.

Python is distributed together with its interpreters and is available for the most used operating systems (i.e. OS X, Windows and Linux distributions). Several third-party tools have been developed to increase its functionalities. Some of these, such as Py2exe or Pyinstaller, can be used to package Python code into stand-alone executable programs for several OSs. This allows the software to be distributed and used without the need to install a Python interpreter.

Extensions such as Numpy, Scipy and Pandas allow the use of Python for scientific computing and execution of machine learning algorithms. These modules provide the support for large, multi-dimensional arrays and matrices, data structures to store data, together with the mathematical functions to operate on them.

Another important extension is Selenium Python, which provides a set of APIs to access web browser functionalities from a Python script. It is implemented through a browser-specific driver. It is capable of launching, sending commands to a browser and retrieving the results.

2.4.5 WebSocket

The WebSocket protocol (RFC 6455) [MF15] provides full-duplex communication over a TCP connection between clients and servers. The goal of this protocol is to avoid long polling and multiple HTTP connections and allow a two-way communication channel with the server from different clients.

The protocol specifies that to open a connection, a preliminary phase is needed in which a handshake is performed. Once the opening handshake is completed successfully, clients and servers can transfer data. The data is sent through *messages*. The messages that can be exchanged are of three types: text data (UTF-8), binary data and control frames. The last one is used for protocol level signaling.

WebSockets are implemented both in Javascript and Python. The support for the Javascript is provided by *HTML5 WebSockets*. It consist of a set of APIs that enables web pages to use the WebSockets protocol. The support of *HTML5 WebSockets* is supplied since Firefox 6, Safari 6, and Google Chrome 14. The implementation of WebSockets in Python is provided by several modules, such as Autobahn.

2.4.6 Windows Installer

Windows Installer [Wil04] provides a standard way to install, maintain and uninstall software. They have a transactional behavior that allows the applications to be either fully installed on the machine or not installed at all to avoid some indeterminate state in between. The extension for this kind of file is MSI and these files are managed by Windows Installer Service. Once executed, it conventionally copies the files needed for its execution in a folder under Program Files associated with the company name and product name.

To handle the installation setup and the creation of MSI files, third-party vendors offer proprietary and open-source solutions. The one used in this work is Wix-Toolset [Fou]. It is able to take the source code, compile and link it to create an executable file. It is based on a declarative XML authoring model to provide an intuitive way to build MSI installer packages.

3 Related Work

During the last ten years, several techniques have been developed and deployed to fight phishing attacks. As expeted, together with the development of prevention techniques, phishers' strategies have also evolved to perform large scale and more sophisticated attacks. The proposed solutions to counteract phishing attacks can be divided into two main categories: phishing prevention and phishing detection techniques.

3.1 Phishing Prevention Techniques

Phishing prevention techniques have been developed to prevent users' interaction with phishing webpages. They work at different levels: while security toolbars give information about the page that the user is visiting, blacklists prevent access by blocking the request that is performed to a certain URL.

3.1.1 Security Toolbars

These security indicators have been developed to provide intelligible information to attest the legitimacy of a website. However, nowadays browsers already provide indicators that can be used to accomplish the same task:

- **URL:** The Uniform Resource Identifier displayed in the address bar can easily be spotted to understand whether the website in which the user landed is the one she was looking for.
- **HTTPS connection:** The protocol used for the communication with the server can be used to infer if the connection is secure (encrypted).
- **TLS certificate:** The Transport Layer Security certificate is used to authenticate the web server that we are communicating with. In browsers such as Google Chrome, by clicking on the green locker displayed in the address bar information about the certificate is available.

These indicators require some technical competence to be fully understood and efficiently used to detect phishing websites. Security toolbars overcome this limitation by providing clear visual information about the website.

The Netcraft toolbar [Net] provides a color code referring to the risk associated with a visited webpage. It also provides additional information such as the data in which the website was encountered for the first time, the country and the name of the host and the rank of the websites based on the number of visits.

Another example is SpoofGuard security toolbar [CLTM04]. Its evaluation is based on a stateless analysis of the displayed webpage. This approach implies the computation of features extracted from part of the page such as URLs, images, links and password requests. The computed result is shown with a color code, similarly to the Netcraft toolbar.

A different approach is proposed by Wu et al. with Web Wallet [WML06]. It consists of a browser sidebar that should be used every time a webpage requires sensitive information such as username and password. The user inserts her information into the sidebar and then Web Wallet checks the legitimacy of the website. If it is considered safe, the data is automatically submitted, and otherwise a warning is raised and no information is sent. With this approach Wu et al. were able to reduce the spoof rate from 63% to 7%.

Another approach is proposed by PhishAri [ARK12]. It is specifically developed for addressing the OSM phishing attacks perpetrated using Twitted. It relies on features extracted from the URL, WHOIS query, text and tags extracted from the posted messages and the contact network of a certain account. Their results shows that this technique is able to detect phish more efficiently (92% accuracy) than the actual Twitter defense system (84%).

Even though these approaches can be useful to reduce the risks of phishing attacks, studies [WMG06] show that the efficiency of toolbars is low since they provide too simple information that does not evoke a sense of danger in users. Web Wallet, as underlined in the user study, is sensitive to attacks in which phishers mimic the sidebar together with the target websites [WML06]. Moreover, Egelman et al. [ECH08] underline how passive warnings are not effective prevention techniques because users tend to ignore them. It is further worth underlining that, as shown in Zang et al. [ZECH06], approaches based on toolbars reach low performance in phishing identification (90%) together with an high false positive rate (42%).

Solutions like Netcraft toolbar introduces a delay in the evaluation of a phishing webpage due to the ranking process used to estimate the webpage legitimacy. Furthermore, they are not effective solution for adaptive attacks that might serve legitimate content to certain uses while phishing content for other. Another aspect to take into account is the lack of guidance that these systems provide. Users might proceed to the malicious webpage just because they want to achieve to their primary task, i.e. login in their on-line banking system or other services.

3.1.2 Blacklist

Blacklists are a popular and widely-deployed technique implemented natively in all of the most used web browser [W3s]. Google Chrome and Mozilla Firefox rely on Google Safe Browsing [goo] while Internet Explorer exploits Microsoft Smart Screen [mic]. Blacklists can be used for either blocking connections to phishing websites or as filters for phishing emails. These approaches rely on lists that are composed of URLs or domain names that have been attested to point to phishing websites. Usually these lists are manually composed after the attestation of a URLs illegitimacy. Thanks to this approach the number of false positives should be almost zero. On the other hand, the time needed for the verification of the websites increases the risk of new phishing attacks. In fact, only 20% of newly created phishing websites are detected [SWW+09]. This detection rate increases to 80% after the first 12 hours.

To overcome this limitation, Liu et al. proposed an approach that is able to increase the human verification of URLs [LXP+11]. Another approach is to build a

blacklist through the prediction of new phishing websites based on the information gathered from already known phishing URLs. Xie et al. [XYA+08] were able to use regular expressions to compute signatures for malicious websites over sets of URLs collected from the same phishing campaign. Another approach [MFSE12] leverages Markov chain modelsand semantic relatedness to build a proactive blacklist of phishing domain name.

On one hand, approaches based on blacklist outperform the false positive rates of security toolbars. On the other hand they introduce a delay that is needed to classify a website as a phish. Sinha [SBJ08] pointed that blacklists are missing a large amount of phishing websites. This is predictable because before having the chance to identify a webpage as a phish, a certain URL must be reported. To address these drawbacks, large-scale phishing detection techniques have been introduced.

3.2 Phishing Detection Techniques

This set of techniques has been developed to address the small usage of toolbars and the delay and lack of coverage of blacklists due to their processes of advertisement, attestation and insertion in the list. Phishing detection techniques are able to identify threats in real-time and, based on the analysis of characteristic features extracted from URLs and webpage content.

3.2.1 Content-based Analysis

Phishers are used to imitating the look and feel of their target to fool users. Based on this assumption, several solutions have been developed to analyze the content and the structure of webpages to detect phishing websites.

CANTINA [ZHC07] is an example of a content-based approach. It relies on the term frequency–inverse document frequency (TF-IDF [SM86]), an information retrieval algorithm, to examine the text contained in a webpage and identify the most important words. The five terms with the highest TF-IDF are taken as a signature for the analyzed website. Subsequently, these terms are used to query a search engine. The first n websites returned are checked against the URL of the analyzed webpage and, if no match is found, it is identified as phish. This technique has a 97% TPR and a 6% of FPR.

More specific techniques rely on the similarities between phishing websites and their target [MKK08, CSDM14]. Medvet et al. [MKK08] present a solution that is able to extract an identificative signature based on the structure of a webpage. Two sets of information are extracted from the DOM. The first is related to the text that is contained in the leafs of the DOM tree and its attributes such as style, color, size, absolute position in the page and font-family. The second takes into account the images with their size, color, source and absolute position in the page. When the signature is extracted, it is compared to the known websites and in case of a high similarity, the unknown instance is considered as a phish.

Chen et al. [CSDM14] rely on the visual similarity of legitimate and phishing webpages to protect a small amount of targeted websites. The system contains an

image stored locally of each website that it wants to protect from spoofing. Every time an unknown webpage is visited, it renders and extracts an image from it. The classification is based on features that are extracted from the Normalized Compression Distance (NCD [CV05]). It is calculated between the stored images and the one from the unknown instance to identify possible phishing websites.

Visual similarity approaches [MKK08, CSDM14] have a small range of action since they rely on information that must be collected from the legitimate webpages that they want to protect. However, CANTINA showed an high accuracy and a low rate of false positives without the need of information about the legitimate webpages.

3.2.2 URL Analysis

To amplify the phishing detection range to vectors such as emails, techniques for URL and domain name analysis have been developed [Mar15]. Since URLs contain information about the resources that they refer to, phishers use obfuscation techniques to make phishing webpages look legitimate.

Taking into consideration the tricks that phishers may perform to obfuscate URLs, different studies have been conducted to outline their techniques. McGrath et al. [MG08] observe that a common practice is the use of long URL with short *mlds* composed of a small amount of alphabetical characters. These URLs often contain the name of the targeted brand or website in a free part of the URL. Attackers also resort to shorten URL services to completely hide the URL that points to the phish.

For the amount of information that can be extracted from URLs, several techniques [LMF11, BWSW10, MFSE14] have been developed to extract lexical features from them. Blum et al. [BWSW10] use a bag-of-words approach. Some features are extracted dividing the URL into different regions (protocol, domain and path) and extracting tokens from each of them. These tokens are extracted within each regions and identify the parts delimited by non-alphanumerical characters. Other features are related to the length of the URL and the number of level domains. Tokens are also used to analyze the URL by Le et al. [LMF11]. Their approach has the advance of not impacting the page loading time while the lexical features are extracted.

There are hybrid approaches that are based both on content and URLs. One example is Monarch [TGM+11]. It performs a real-time analysis on lexical information extracted from the URL, Javascript events and dynamic content loading. This technique presents a delay of around 5 seconds and an accuracy of 91%.

3.3 Phishing warnings

Besides key aspects such as high accuracy and low false positive rate, it is important to present to the users an interface that makes them trust and understand the result of the analysis and warns them of the threat.

Netcraft and SpoofGuard toolbars provide a color scheme, while Google Chrome and Mozilla Firefox present a warning message that impedes access to the webpage.

Several studies [ECH08, AF13] analyze and compare different approaches regarding how the information displayed to users is taken into consideration. Egelman et

al. [ECH08] focused on Internet Explorer and Mozilla Firefox. They show that passive warnings provided by the former are effective only 13% of the time, while active warnings reach 79% success in steering users away from phishing websites. Akhawe et al. [AF13] compare different types of warnings displayed by Google Chrome and Mozilla Firefox. They found these active warnings successfully persuaded the user not to proceed to the malicous websites. However, the clickthrough rate was around 18% for Chrome and 9% for Firefox.

Other studies focused on brain activity and eye movement while users encountered warning messages. Neupane et al. [NRSH15] pointed out that users spend more time looking at the login form compared to key areas such as the URL for understanding the trustworthiness of a website. The gaze pattern analysis shows that they actually pay attention and read the warnings. However, as described by Akhawe et al., they might not understand the information that is displayed.

Finally, common recommendations for the design of warnings are interrupting the primary task, using active warnings, providing clear choices to the users and displaying warning messages only when needed to prevent habituation.

4 Problem Statement

Twenty years have passed since the first use of the term *phishing* and now it has become one of the most lucrative cybercrime activities [JM06]. Researchers and companies have been working to deploy effective solutions to face this threat. Browser level solutions make use of blacklists and centralized servers to filter requests sent to users and to present a warning message instead of the webage. Toolbars or add-ons are available to collect users' feedback in order to rank pages according to their level of security. Other solutions make use of machine learning algorithms [WRN10, ZHC07].

These solutions have several drawbacks:

- They do not analyze the webpage content actually displayed to the user to make the decision, but only check the URL of the page against a blacklist. They are thus vulnerable to websites that perform adaptive attacks and serve either harmless or malicious content depending on who performs the request.
- Blacklists are periodically updated, but the delay to label a webpage as a phish gives the attacker a time window in which he can trap victims.
- Centralized solutions may compromise user privacy and disclose browsing history or metadata that can be used to profile users.
- Existing machine learning approaches require a large set of labelled data to be trained. Big sets are not easy to gather due to problems such as the short uptime of phishing websites.
- Classification relying on bag-of-words features make these solutions dependent on languages and brands that have been used to train the model.
- None of the existing approaches propose alternatives to the user except proceeding to the website or closing the tab when a phishing webpage is encountered.

To address these drawbacks we will develop a software that implements the system described in Section 2.3. Doing this in real-time, preserving privacy and providing a good user interface presents several challenges that we list hereafter:

4.1 Deployability

Some of the solutions proposed so far have the drawback of lacking deployability in multiple platforms. Some solutions [YW10, DT05] were developed only for Mozilla Firefox or Internet Explorer [CLTM04]. To publish an effective solution that is able to protect as many users as possible we must integrate *Off the Hook* in the most used browsers and as many operating systems as possible (**Requirement 1**).

4.2 Performances

Phishing is in constant grow because users keep falling for it. As long as the information gathered will lead to profit, attackers will try new ways to fool their victims. Our aim is to protect users from revealing information to phishers, and to achieve this we have to prevent their interaction with phishing websites. To deploy this software client-side the impact on the system should be minimal. Even though in today's devices the memory available is usually higher than 4 Gb, is not unlimited. Thus, it is mandatory to reduce the amount of resources drained (**Requirement 2**). Furthermore, the analysis performed by the system should be executed in a small amount of time. This time window needs to be smaller than the time a generic user needs to start typing her information in the corresponding input fields. We estimate that the time needed for interaction is a few seconds from the moment the webpage is loaded. However, our goal was to complete the evaluation within one a second (**Requirement 3**).

4.3 Reliability of decision

The technique that is the basis of this work can detect a phishing website with 99,9% accuracy and 0,05% FPR (Section 2.3). One of the main challenges of this work is to apply this technique on the user's machine while they are browsing. To achieve this, architectural changes are needed. Variation in the modules, data structures and even languages in which the script is implemented might have an impact on the *data sources* extracted from the webpages. We must extract the same data from the page to be able to recreate a scenario as similar as possible to the one depicted in Marchal et al. [MSSA16] to achieve similar classification performances (**Requirement 4**).

4.4 Ease of usage

It is crucial for any application to be intuitive to use and easily understandable. A lack in either of these properties will result in failure, regardless of how well the program is capable of accomplishing its task. Designing an effective user interface is not a trivial task. Several studies have been conducted to understand users' reactions to warning messages [ECH08] and their behaviour even up to neuro-physiological level approaches [NRSH15]. Two of the problems observed have been that, though users read warning messages, they might not understand them, and passive warnings are not taken into consideration as real threats to their security. We want to design an active warning that instills a sense of danger in the user and is written using words that naive users can understand (**Requirement 5**).

4.5 Usefulness of functionalities

Phishing prevention techniques, like those deployed in Google Chrome and Mozilla Firefox through Google safe browsing, prevent the user from proceeding when a suspected malicious website is loaded in the browser. The user can then decide to read the warning message, look for further explanation, leave the page or proceed to

the suspicious URL. We want to increase these continuation options by exploiting the characteristic of the phishing prevention technique that our system implements, i.e. providing a redirection to the website that has been identified as the target of the phish (**Requirement 6**).

5 Off the Hook - A phishing prevention software

Off the Hook is a software utilizing machine learning techniques that computes features from data sources retrieved only by the web browser (*starting URL, landing URL, redirection chain, logged links* and *HTML source code*). The feature set models phisher limitations and measures the consistency in term usage in the different data sources.

5.1 Design

Different solutions were taken into account to achieve the maximum in terms of deployability and code maintainability. We decide to split the system in two parts: three *background processes*, that handles the machine learning computation, and an *add-on* that handles the interaction with the browser (Fig. 4).

Add-on: runs in the browser and is meant for browser interaction and displaying the classification results. It is written in Javascript and is composed of the *content script* and the *background script*. The former interacts with the web pages, is loaded concurrently with them and can read, modify, or interact with them, to modify its content or the information displayed. The latter is meant for long-running tasks, is loaded with the extension itself and stays active until the extension is disabled or uninstalled. It is able to maintain the state of the application and can communicate with the *content script* in order to send and receive information.

Background processes: are written in Python and run on the host operating system. Their purpose is to analyze the information collected by the *add-on* and send back the result after computation. Using Python for the classification part allows us to: use the same libraries used in the phishing detection system (Section 2.3) to limit the *reliability of decision* problem and also to reduce the *deployability* problem, because the same code can be executed on every operating system.

The *background processes* consist of three modules: a *phishing detector* to classify webpages as phish or legitimate, a *target identifier* to infer the potential targets of a phishing page, and a *dispatcher* that handles the data exchange within the *background processes* and the *add-on*. To reduce the workload, a *whitelist* of hashes built upon the *starting URL, the landing URL* and the *HTML source* identifies trusted webpages that bypass the analysis.

All the components communicate with each other using WebSockets [MF15], that provides full-duplex communication over a TCP connection.

5.2 Implementation

We now present each component, how they interact, the messages exchanged and the work-flow of the information and results. We analyze first the *add-on* and then the *Background processes*.

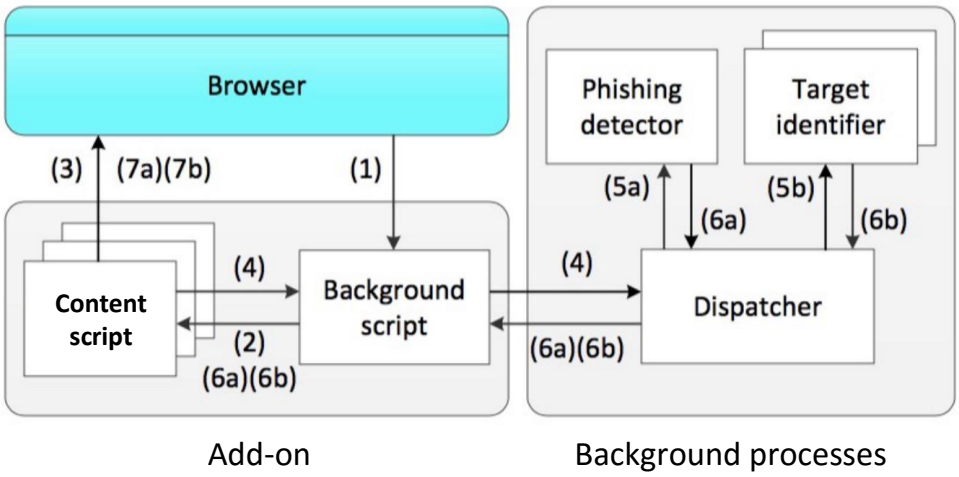


Figure 4: Application architecture

5.2.1 Background script

The *background script* runs concurrently with the browser application. It is loaded when the extension is installed or every time the browser is launched. It takes care of the connection between the *add-on* and the *background processes*. In particular, it opens the connection to the *dispatcher* and acts as a bridge between the *content script* and the *dispatcher* for the exchange of raw data and results. It also stores, as depicted in Figure 4 (1), the request that a page performs when rendered (*logged links*), the first URL that is requested (*starting URL*) and the redirections performed (*redirection chain*). It listens for three different events:

- **Before navigation:** intercepts the requested URL that the user wants to visit, in our implementation this will be identified as the *starting URL*.
- **Redirect:** is triggered when a redirection is executed to store the visited links and compose the *redirection chain*.
- **Request:** is triggered every time a request is made in order to download a content: images, HTML pages, Javascript scripts, CSS stylesheets, JSON objects, or others, to compose the *logged links* list.

The collected *logged links, starting URL* and *redirection chain* are stored into an associative array with the identifier of the tab that triggered the events. When a page is loaded, the *content script* is executed. It performs a request to the *background script* that responds by sending the data corresponding to that tab (there is a different instance of *content script* for each browser tab) and erases it from the memory (2). Erasing the collected data every time a new page is requested prevents the increase of the *add-on* memory usage.

The *background script* later forwards the results from the *phishing detector* and *target identifier* to the *dispatcher* and to the *content script* (6a)(6b).

Even though the system is completely client-side, it is useful, if the user allows it, to log data to estimate the usefulness of the continuation options, the execution times on different machines and to collect data from phishing websites that can be used to retrain the model. The collected data is stored without any reference to the source. Three levels of logging are available:

- **None:** default setting, it stores no information about either the browsing or the results of the analysis.
- **Metadata collection:** allows the collection of quantitative data, number of pages, referring to how many visited webpages were marked as phishing or legitimate. The collected data is: execution time (divided between the different modules) and decisions taken by the user when a warning message is displayed (which option was chosen). None of this information is related to any URL or browser history.
- **Website data collection:** besides the *metadata*, it stores the *data sources* collected from the webpages detected as phishing and sends them to a server. Due to the complexity of gathering ground truth and data about phishing webpages, the collected data is useful for potential classification model retraining.

5.2.2 Content script

The *content script* has one instance injected in each open tab of the browser and runs for every loaded webpage. Each tab is identified by a unique identifier (*tab id*) within the browser, the same *tab id* is used to discriminate between the different instances of the *content script*.

This module retrieves the *data sources* collected by the *background script*: *starting URL, redirection chain* and *logged links*, as depicted in Figure 4 (2). After that, it parses the page to extract other data from the current webpage (3):

- **Input field:** the number of input fields displayed. The more information a phisher wants to collect, the more input fields are needed.
- **Images:** the number of images, which can be logos, icons, background or others.
- **Title:** the string displayed on top of the browser's tab.
- **Text:** the concatenation of all the strings displayed in the page.
- **HTML source:** all the code embedded into the displayed page and the code downloaded from other HTML or PHP documents.

All the *data sources* are inserted in a JSON object that includes also the *tab id* and *page id*. The latter is a randomly chosen number that refers to a particular page loaded from the same tab. When ready, this object is sent to the *dispatcher* through the *background script* (4).

All the *content scripts* are registered to the same listener for incoming messages from the *background script*. Each instance filters messages by *tab id* and *page id* in order to collect only the messages addressed to it.

Since the execution of the *phishing detector* and the *target identifier* are done in parallel, the results are collected separately (6a)(6b). Once the results are available they are stored in a local variable, and, depending on the order in which they are received and the result itself, the behaviour of the application changes (7a)(7b).

We can define four different behaviours:

- **Green icon:** the user continues to browse normally and no banners or messages are displayed. The legitimacy of the page is attested by a green badge over the add-on icon in the top bar of the browser. The time between the page loading and the final result doesn't impact the user experience due to the asynchronous execution of the script.

- **Loading icon:** a loading gif is displayed. This action is taken only in case the webpage has been marked as phish by the *phishing detector*. User interaction with the web page is prevented and the system waits for the result of the *target identifier*. This is a temporary state that will evolve either into the *safe banner* or the *warning banner*.

- **Safe banner:** on the top right of the page, a green version of the program icon is displayed for a few seconds to attest its legitimacy. Concurrently a green badge is placed over the add-on icon. After that, the webpage is added to the *whitelist* to avoid the *loading icon* from being triggered in future visits.

- **Warning banner and red icon:** an alert message is displayed to the user. Interaction with the webpage is prevented by a layer between the banner and the page content. The banner is composed of general information about phishing and additional information extracted by the *target identifier*. Furthermore, a red badge on the icon of the add-on reminds the user of the malicious nature of the webpage even in case the banner is dismissed.

The decision trees displayed in Figure 5 and Figure 6 show the *add-on* interaction with the user interface based on the results of the computation. Due to the parallel execution of the *phishing detector* and *target identifier*, there is a race condition that modifies the *add-on* behaviour depending on the order in which the results are received, as depicted in the two trees.

Figure 5 depicts how, in case of a legitimate response from the *phishing detector*, the *add-on* attests the legitimacy of the page by displaying the *green icon*. The behavior is different in case of phishing, where the *loading icon* is displayed while the *add-on* is waiting for the result of the *target identifier*. When ready, the targets found are checked against the *landing URL* of the visited page in order to find a correspondence. If one of the targets matches the main level domain (mld), the page is added to the *whitelist*.

In the tree displayed in Figure 6, we can see that in case the first result is returned by the *target identifier* there is no perceivable action in the browser. Though, the

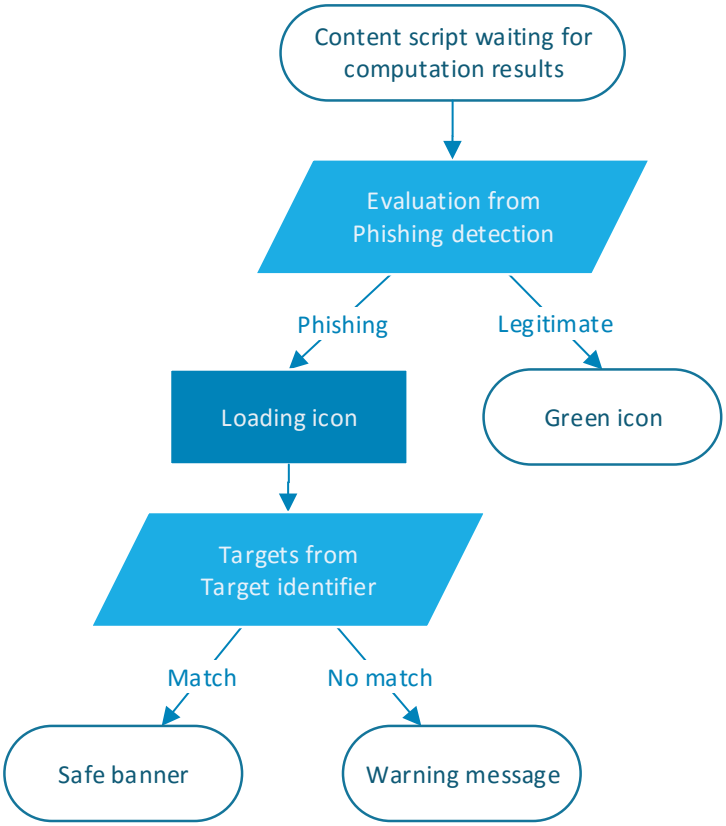


Figure 5: Add-on behaviour in case the first result is received from the *phishing detector*.

returned targets are checked against the *landing URL* and the result is stored in a local variable. When the results of the *phishing detector* are available, in case no match was found and the classification has marked the webpage as phish, the *warning message* is triggered. In all other cases the *green icon* is displayed. A detailed analysis related to the different banners is made in Section 5.5.

It is important to notice that no action is performed to block the user interaction with a webpage until a positive response from the *phishing detector* is received. In this way we let the browsing experience proceed unaltered in case legitimate webpages are visited.

The main purpose of *Off the Hook* is to prevent users from disclosing personal information to phishers. To achieve this, it is important to impede interaction between these two parties. Independently of the order of the results, no action is taken before the end of the computation by the *phishing detector*.

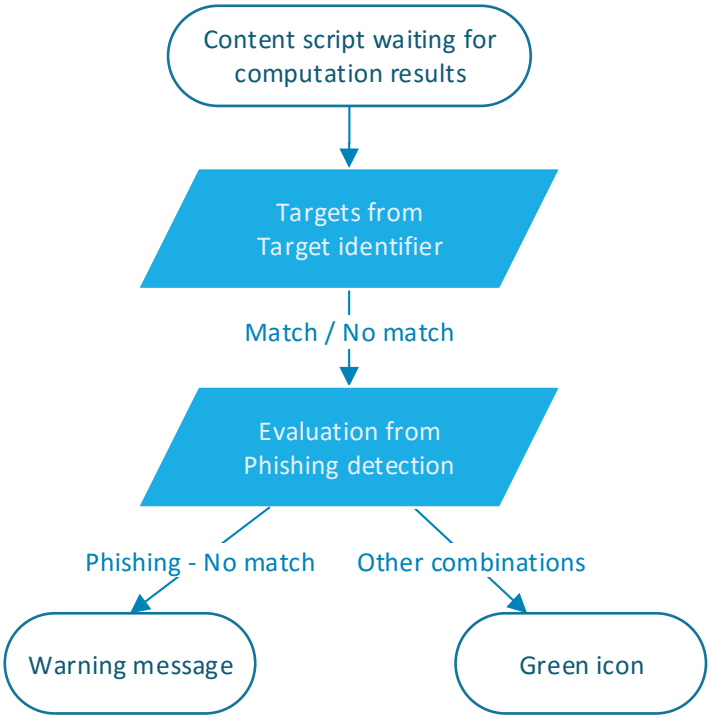


Figure 6: Add-on behaviour in case the first result is received from the *target identifier*.

5.2.3 Dispatcher

This module is one of the *background processes*. It is written in Python and communicates with the *background script* using WebSockets protocol. This protocol implies a client-server architecture and the server in our network is located in the *dispatcher*. When running, it waits for incoming connection from the *background script*, the *phishing detector* and the *target identifier*. It can handle multiple instances for each of them. For a good trade off between memory consumption and performances there is only one instance of the *phishing detector* and two of the *target identifier*. The number of *background scripts* does not depend on how many browsers are open. The *dispatcher* is capable of handling different browsers at the same time.

Its main task is to receive data collected by the *content script* (4) and compute the hash of the webpage for checking against the *whitelist*.

The *whitelist* contains a list of hashes computed from the *landing URL*. Its purpose is to filter the request from the *add-on*. If a match is found, it sends a message attesting the legitimacy of the webpage directly back to the *content script*. In this way, it reduces the CPU workload and the delay of the answer. This list is stored locally on the machine of the user. It is loaded every time the *dispatcher* is

launched (when the system is booted) and remains in the memory to reduce the delay when used. The impact on memory usage is negligible. It consists of an associative array with a small amount of entries (depending on the number of whitelisted pages).

If no match is found in the *whitelist*, the data received from the *content script* is forwarded to the *phishing detector* (5a) and to a task-free instance of *target identifiers* (5b) for processing. It later addresses the results of each operation to the *content script* (6a)(6b). Besides this, it is also responsible for:

- Uploading to the server the information collected by the application according to the logging level chosen by the user: *none, metadata collection* or *website data collection*.
- Storing information about:
  - In case the *metadata collection* is enabled:
    - *Computation time:* a file that contains a JSON object regarding the execution time of each modules for each analyzed webpage.
    - *User decision:* a file in which the information is stored concerning which continuation method (described in Sec. 5.5) the user chose when she encountered the warning message.
    - *Pages visited:* three numbers of visited webpages that have been marked as legitimate, phishing or contained in the *whitelist*.
  - In case the *website data collection* is enabled:
    - *Phishing websites:* the *data sources* regarding the websites classified as phishing.

5.2.4 Phishing detector

This module is in charge of receiving the JSON object from the *content script*. Based on the *data sources* contained in it, the *phishing detector* extracts the information needed to build the feature vector and performs the classification of the webpage (5a) as described in Section 2.3. Once the page has been labeled as phishing or legitimate, this result is sent to the corresponding instance of the *content script* (6a).

5.2.5 Target identifier

This module is in charge of inferring the potential targets of the phishing webpage from the received data sources (5b). It performs the keyterm extraction described in Section 2.3.6 to collect the *boosted prominent terms* and *prominent terms*. With these two sets it performs three steps of query to the search engine to collect candidate targets. These *mld* candidates are ranked based on how many times they appear in the data sources of the webpage. The top-3 targets are sent back to the *content script* (6b). If one identified target matches the main level domain of the current website, it is considered safe regardless of the decision of the *phishing detector*, and the *loading icon* will therefore disappear. Otherwise, the *warning message* is displayed with

the potential targets. Due to the time consumed by this task, two *target identifier* processes are involved to share the workload.

5.3 Deployment

Different solutions were taken into account in order to achieve maximum deployability and code maintainability. We decided to split the system into two parts: a *background process* that handles the phishing detection and target identification and an *add-on* that handles the interaction with the browser.

5.3.1 Cross OS

Using Python as a language for the *background processes* allowed it to run on every OS as far as the correct version of the interpreter and all the related modules were installed.

Providing different installation packages to the user is possible. They should be specific for the target operating system, in order to download and install the proper version of Python and the modules. However this approach presents several drawbacks related to the system requirements and compatibility of different version of the same module. Libraries used for numerical computation are not easy to install on operating systems like Windows. This happens because Windows does not provide any package manager like the Ubuntu Advanced Packaging Tool (APT). Furthermore, the time required for the download and installation of the requirements is above 30 minutes.

To overcome these limitations we used a program that was able to convert Python programs into stand-alone executables: PyInstaller [PyI]. It is able to create single executable files, totally self-contained, which run without any external dependency. It only needs an environment in which the application is able to run in order to collect the necessary modules that it has to embed in the executable file.

We recreated a working environment in the following operating system:

- Virtual machines:
  - Windows 8 x64 (compatible with Windows 8.1)
  - Windows 10 x64
  - Ubuntu 12.04 x32 (compatible with Ubuntu >= 12.04, Fedora >20)
  - Ubuntu 12.04 x64 (compatible with Ubuntu >= 12.04, Fedora >20)
- Physical machine:
  - OS X El Capitan

The executables produced thanks to this process were not the last stage of the deployment. Two approaches differentiate the building of the installation packages that are provided to the end user. On one hand, for unix-like OS, we choose to ship the code in a ZIP files containing a folder with the installation and uninstallation

script written in bash. On the other hand, for Windows, we choose a more visually oriented approach through the use of Wix toolset [Fou]. This tool allows the creation of installers for Windows Installer, the Windows installation engine. It is able to create a Microsoft installation package (MSI) that allows the user to install the application by simply double-clicking on the package. For the uninstallation process it is sufficient to go the the program menu and uninstall the application that is present in the programs list. In both scenarios the application is added to the startup programs in order to be launched at every boot and be ready to compute the JSON object from the *add-on* without introducing initialization delays.

5.3.2 Cross browser

Add-ons are integrated directly into the browser, and there is no need for different versions of the code to port it among different OSs. However, the architecture of these softwares changes between browsers, due to the different APIs that they support. To overcome this limitation we built our solution, in the first version of the application, on a framework: Crossrider 2.4.3. It provides a level of abstraction on top of the different APIs that allows the programmer to write a single code for all the supported browser. So far it supports Internet Explorer, Safari, Google Chrome and Mozilla Firefox. However, not all functionalities are available for all of them, and this led to a stable deployment of the add-on only for the two most used browsers [W3s]: Chrome and Firefox.

With the advancement of the project, we decided to develop the add-ons natively. This kind of implementation allows the programmer to directly use the APIs provided by a determinate browser without the need to go through the abstraction layer provided by the Framework. This choice, even though it increases the cost of maintainability of the code, allows to achieve higher flexibility. For this test we decided to target the new opportunities provided by Firefox Web Extension, a new browser extension system introduced in the version 48 [Cor] that tries to make easier the porting of add-ons to and from other browsers and simplify the use of the APIs.

5.4 Performance optimization

Before the final version, the system has gone through several revisions and optimizations. We are going to present the comparison between the performance of one of the early stages of development, from now on called *phishing detection script (PDS)*, and the current version, *Off the Hook*. The modifications that the workflow has been subjected to in order to optimize time consumption are depicted in Figure 7.

The development of a portion of the software as an add-on, allows the programmer to have APIs for monitoring the events triggered during the navigation and collect the *logged links* during page rendering. This, combined with the use of a multi-process approach helps to parallelize the computation whenever possible (i.e. *phishing detection* concurrently with *target identification*).

One of the main disadvantages of a stand alone applications is that we have to download the page that we analyze. To collect the *data sources* the page must be

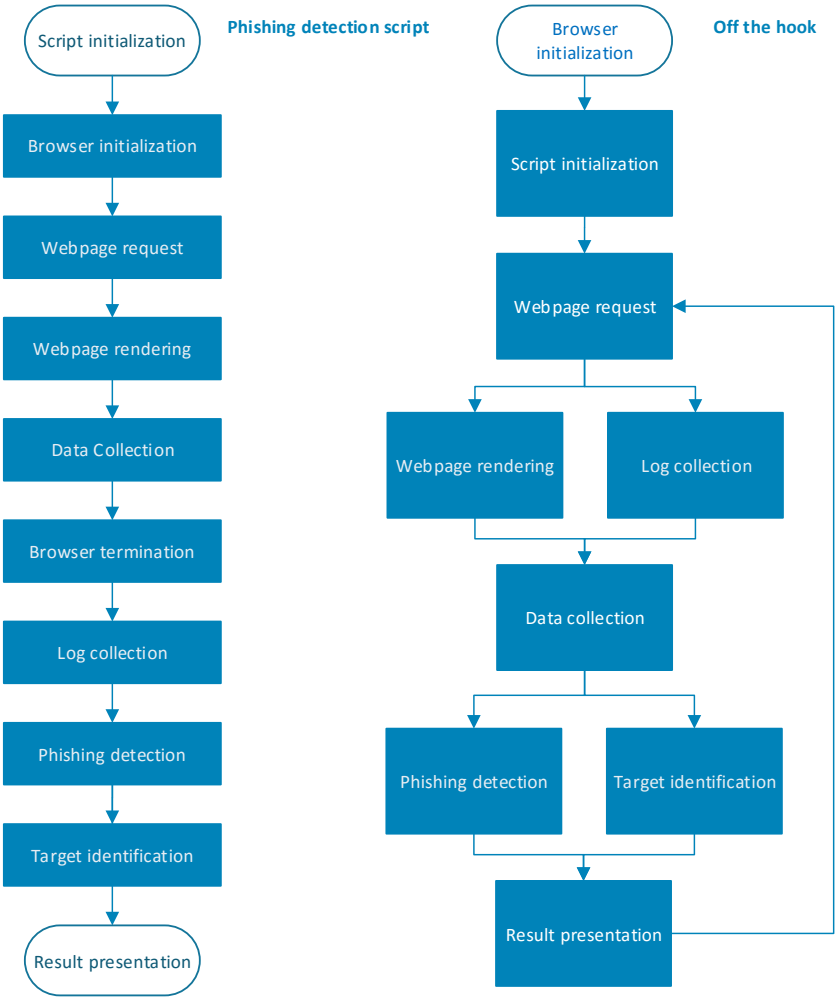


Figure 7: Comparative workflow

fully rendered. While some information can be extracted directly from the page, we need to redirect the log of the browser to a log file to extract the *logged links*. To be sure that this data refers to the same page we have to collect the information one page at a time. Once the webpage is loaded we can extract information such as *text*, *landing URL* and the *screenshot* can be captured. Then, the browser can be closed and the *logged links* collected from the log. Then the feature vector can be built and the *phishing detector* can make its evaluation. Thereafter, in case of phishing, the *target identifier* is executed and when its computation is finished, the result is returned.

On the other hand, using the advantages provided by the browser APIs, we can extract the information directly when the user is browsing, removing the overhead of a separate download. When the page is rendered, we can extract the information

without preventing user interaction, and in this way the computation is transparent to the user. The collected data is sent to the *phishing detector* and *target identifier* to be computed and the results are returned to the *add-on*. With this kind of implementation we do not capture the screenshot of the webpage. This due to the complexity of performing it in Javascript and the time needed for the OCR that is in the order of seconds.

5.4.1 Execution time

Subsequent to the architecture optimization, the problem of time consumption was shifted to the construction of the feature vector used in the classifier. To tune the feature computation, it has been divided in 17 milestones to estimate the critical parts. The milestones are the following:

- **Initialization:** includes the load in memory of the top 1 million visited websites contained in the Alexa list from a CSV file.
- **Starting URL:** extraction of the features from the *starting URL* such as the protocol used, the number of dots, the length of different parts, the number of terms and Alexa ranking.
- **Landing URL:** extraction of the features from the *landing URL* such as the protocol used, the number of dots, the length of different parts, the number of terms and Alexa ranking.
- **Redirection chain:** extracting the *mld* from the *redirection chain* that are used to discriminate between internal and external sources.
- **Logged links feature processing:** extraction from the *logged links* of information such as the protocol used, the number of dots, the length of different parts, the number of terms and Alexa ranking to compute the features.
- **Logged links:** computing the mean, median and standard deviation on the features extracted from the *logged links*.
- **HREF links features processing:** the same process used for the *logged links* is repeated with the *HREF links*.
- **HREF links:** computing the mean, median and standard deviation on the feature extracted from the *HREF links*.
- **Visible text:** extraction of the term distribution of the *text*
- **Title:** extraction of the term distribution of the *title*
- **Text in HTML source:** extraction of the term distribution of the *text* extracted from the HTML source code
- **Inputs and images:** calculating the number of input fields and images

- **Terms usage consistency:** extraction of features related to the similarity of the different term distributions (Sec. 2.3.4).
- **Title words in starting and landing URL:** features regarding the presence of *title* words in the *mlds* of the *starting* and *landing URL*.
- **External logged links words in starting and landing URL:** feature regarding the presence of *logged links* words in the *mlds* of the *starting* and *landing URL*.
- **External HREF links words in starting and landing URL:** feature regarding the presence of *HREF links* words in the *mlds* of the *starting* and *landing URL*.
- **De-initialization:** closure of the CSV and saving the data sources on a JSON file.

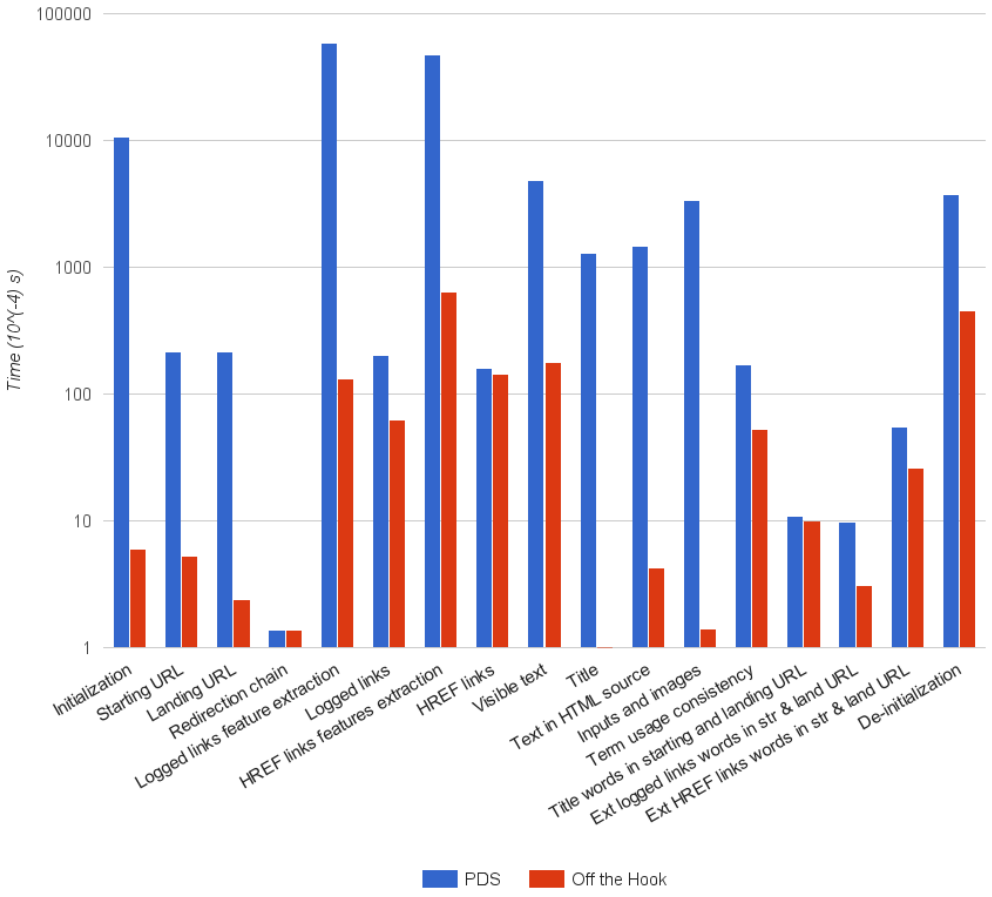


Figure 8: Delays before and after the optimization on amazon.com

The result comparison of the early stages of application with *Off the Hook* are shown in Fig. 8. It is easy to see that the most time consuming fractions were the following:

- **Initialization:** the main problem of this part of the code was the loading of a comma-separated values (CSV) file containing the one million Alexa list. In this particular case it was used to check the name of the domain from extracted URLs against the most visited websites according to Alexa. The time for loading a million entries was around 1,2 seconds. That alone was enough to prevent a real-time execution. The first adopted solution was to load the file in the memory at the startup of the program, but as described in Section 5.4.2, this leads to a memory consumption problem.

- **Computation of logged links and HREF links:** these were the causes of the large amount of time wasted in the milestones: the use of the module Selenium, for the parsing of the webpage, and the inefficient management of the checks performed in the code. After a deep analysis of the function called from the module, we re-implemented the ones that we needed directly in the *add-on*. This ad-hoc approach allowed us to perform additional controls at each iteration of several nested loops. Thanks to that, in the best case scenario, we are able to avoid an $O(n^2)$ computation performing a single conditional statement.

- **Text-related features, input fields and images:** all the features related to text or data extraction were performed using Selenium. It was used to locate *title, text, input fields* and *images* within the page source code. The drawback of this approach was the time overhead. It was introducing a delay to each function call proportional to the size of the HTML source code analyzed. To reduce the time consumption, Selenium was removed and the extraction of these *data sources* was moved to the *content script*. Operated directly in the browser, the execution time is shorter.

- **De-initialization:** the removal of (i) the Alexa list and (ii) the need to save the *data sources* in a JSON file on disk, allowed us to reduce the time used in this final step of the features vector construction.

Figure 8 depicts how for milestones such as *initialization, text in HTML source* and *inputs and images* the time consumption is 1000 times lower. Remarkable optimization can be seen in the *starting URL, landing URL* and *HREF links features extraction* that decrease the time required for their computation by a factor of 100. Furthermore, the preprocessing of the *title* makes us save 1,2 second only in this milestone.

5.4.2 Memory consumption

Although nowadays computers are equipped with at least 4 Gb of RAM, it remains important to minimize memory usage as much as possible. With the improvement of

the performances we aim to memory consumption optimization as well. To achieve this goal we went through three main refactoring:

- Import optimization
- Replacement of Selenium with the ad-hoc functions
- Removal of Alexa list

Import optimization: with the code refactor performed from the early versions of the application, we were able to move some computation from one script to another. Due to these changes, we were able to remove several imported modules used in the moved code.

Replacement of Selenium with the ad-hoc function: due to its large amount of delay and memory drain we preferred to completely remove this module. This was possible thanks to the re-implementation of the same functions both in the add-on and in the script.

Removal of Alexa list: the problems related to the 1 million most visited websites list were multiple. In the first instance, the file containing the list, a CSV file, was alone occupying more than 22 MB on disk. That had to be loaded in memory, consuming more than 1,2 seconds and the space required to store it in an appropriate data structure. In order to have a fast lookup in the structure a list was not a good option, because for each lookup the complexity is O(n) and the estimated time for a miss in that list was about 0,025 sec. This time alone does not constitute a huge problem, but for an average webpage, the URLs to check against that list are around 20, and the number can be ten times bigger. The best option is a dictionary, as an associative array with the string containing the domain as key and the rank as key. Using this approach the complexity is reduced to O(1). Even though the implementation of a dictionary in Python consumes a relatively small amount of memory, the size of the list carries a space occupation problem. For these reasons, we decided to modify the model by removing the features related to the Alexa list.

5.4.3 Features vector

To increase the performances of the model, together with the improvement in the execution time and memory consumption, we performed some modification of the features vector. From the 212 features described in Section 2.3.4, some features were added and other removed to reach the final number of 210 as shown in Table 8.

The differences can be analyzed based on their belonging to a particular feature set:

- **URL:** From the starting 106 features we remove the ones related to the Alexa list. The page ranking was calculated on the *starting* and *landing URL* and used to calculate the mean, median and standard deviation on the ranks of the *logged* and *HREF links* (both internal and external). Once removed, the number of URL related features decreased to 92.

Name	PDS	Off the Hook	Type
f1	106	92	URL
f2	66	68	Term usage consistency
f3	22	32	Usage of starting and landing URLs
f4	13	13	RDN usage
f5	5	5	Webpage content
fall	212	210	Entire feature set

Table 8: Comparison of features sets

- **Term usage consistency:** After an analysis of the features with a high impact on the classifier, two of them appeared to be really helpful in the discrimination between phishing and legitimate websites. To stress their importance, we insert two binary features that refer to the presence of words extracted from internal and external *mlds* in the words extracted from the *title* of the page.
- **Usage of starting and landing URLs** Besides the 22 previously computed, we add 10 more features that are calculated as the sum of probabilities from terms of *Dtitle, Dintlog, Dextlog, Dintlink* and *Dextlink* that are substrings of *starting* and *landing free URL*, but not substrings of *starting* and *landing mld*.

5.5 User Interface

There are several studies regarding the user's perception of warning messages [AF13, ECH08, NRSH15], the effectiveness of which depends on different factors. While some of these are related to gender and age [SHK+10], others are linked to the way in which the warning is displayed and the effect of the habituation [AF13, ECH08]. As shown in Dhamija et al. [DTH06], even for competent users there is a chance of falling for sophisticated attacks.

Section 5.5.1 describes how we used some of the suggestion provided by previous works and from other phishing warning to design our user interface. Section 5.5.2 presents a cognitive walkthrough that was used to improve the efficacy of the initial interface. Section 5.5.3 presents the interface of *Off the Hook* and in Section 5.5.4, it is compared to Chrome and Firefox banners.

5.5.1 Design of the interface

The interface of the application was designed to meet a trade off between informativity and ease of understanding. As underlined in previous studies, warning messages should be distinguishable from other banners [Wes08]. The text only mentions facts that are immediately relevant to the user, with a *More info* link available for further information. The sentences were made as short as possible, dividing different pieces of information to separate clauses. We also aimed to keep technical terminology at a minimum, so that even a user unfamiliar with *phishing* would understand the purpose of the warning.

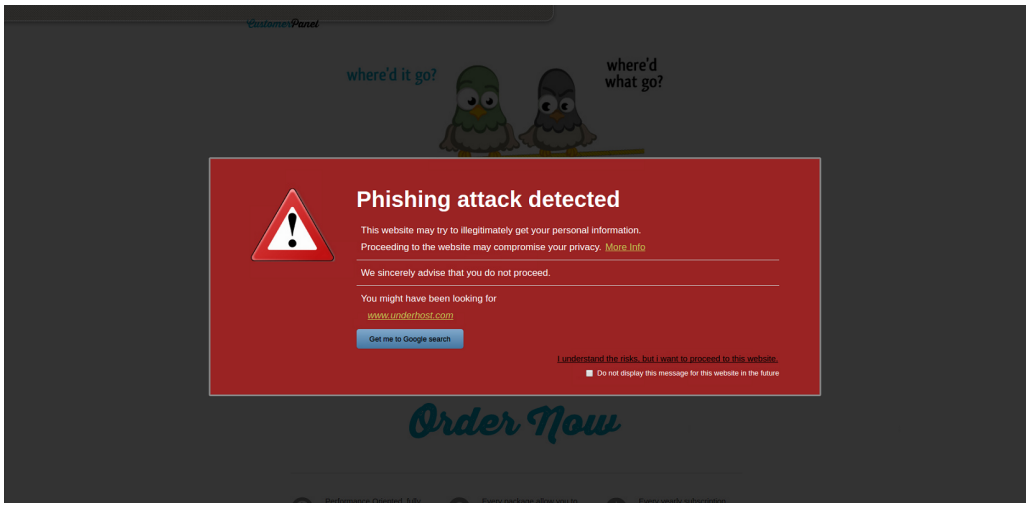


Figure 9: Warning message

Different phishing warnings give varying amounts of information to the user, as well as differ in the possible continuation links they offer. We aimed to find a balance between the amount of technical information and the length of the text by explicating the concept of phishing and showing the target website, but also eliminating everything irrelevant for immediate decision making. Learning more about phishing is not something the user should be expected to do from the banner itself, although it can motivate her to do so, i.e. by visiting the *More info* link.

The most important addition that we provide is a continuation option (a link) to the target site. This is among the most important novel features of our application. Further, we give the option to proceed to the site detected as phishing. This link is black, purposefully making it hard to see and hard to choose accidentally. Next to it there is a check box that allow the user to add the webpage to the whitelist to prevent the user from having to see that banner on that website on future visits.

The last continuation option is a blue bottom *Get me to Google search*. It redirects the user to Google performing a query with the identified target.

The red background color was a standard choice as a general indication of danger. The shade was chosen by its high visibility that encourages alertness. The text color is white due to its contrast with the red background, and the yellow in the target link stands out from the rest of the text.

The image is a standard warning triangle reminiscent of traffic signs and similar to images from other web warnings. Its interpretation can be assumed to be universal and unambiguous. It was made large enough to be easily noticeable, automatically grabbing the user's attention, but not so large as to divert attention from the text.

The area that surrounds the banner is a semi-transparent layer that allows the user to see the websites that she is landing to. It is useful for preventing user's interaction with the webpage, while it gives an idea about what the webpage looks like.

5.5.2 Cognitive walkhrough

To improve the usability of the *Off the Hook* user interface, we conducted a cognitive walkthrough with four participants: two developers of the program, a research assistant, and a naive user without expertise either with the program or security software in general. The cognitive walkthrough is a usability evaluation method focusing on learnability, understandability and the experience of a first-time user of the product tested. As a usability evaluation method, it is fast, low-cost and problem oriented, focusing on psychological factors of a first-time user, such as perception, attention and memory load [WBJF92, WRLP94].

The purpose of the walkthrough was to shed light on the following questions:

- Is the interface understandable to a first-time user?
- Is phishing explained well enough for someone not familiar with the concept?
- Are all the continuation options useful, and should other alternatives be added?
- Are the continuation links understandable to a first-time user?
- Is the text readable for a non-specialist?
- Is the placement of the text and the links appropriate?
- Is all of the text necessary, or should some of it be removed?
- Should information be added to the text?
- Is the visual appearance of the banner professional or appealing?
- Are the size of the banner and its placement on the screen appropriate

The interface of the banner prior to the walkthrough is shown in Fig. 9. Numerous issues were discovered in the walkthrough, which resulted in modifications of the interface. These are described in this section.

Text In the *warning message*, among the main problems detected was the amount of technical details and terminology provided to the user. As Egelman et al. [EMC+10] show, users are more likely to ignore security warnings if they use technical language that is hard to understand.

The term *phishing* should not be assumed to be understandable to a non-expert, making the original headline problematic. The main challenge here is how to indicate that the software has not detected e.g. a virus attack but a phishing site, using only ordinary non-technical language. The phrasing *Privacy threat detected* was concluded to best serve this purpose, since it contains no IT lingo, is short and concise, and leaves little room for misinterpretation.

Further, it was concluded that the text in the *warning message* was not clear enough to make the user realize she had been misled by the original source of the link, most likely an email. In the pre-walkthrough add-on, the text above the

redirection link to the target website said: *You may have been looking for:*. This can be misunderstood to mean that the email was legitimate, and only the link was somehow flawed. The user could then proceed to the actual website and attempt to do the tasks the email wanted her to do on the phishing website (e.g. update their personal information in certain ways). Obviously this is not a desirable outcome; the user should realize that not only the link but the email itself may be illegitimate. The text was changed to: *This website may try to mimic:*. The link to the target originally only showed the name of the company, but was changed to the full URL for clarify that is actually a link.

Functionality On the bottom left of the *warning message* there was a link to Google search. However, since it is unlikely that the user had entered the phishing site via Google, it is also unlikely that she would like to Google the site after exiting the phishing site. The Google link was originally intended to provide a safe place the user could go to, but this was not clear enough. The link was changed into a *Close tab* button, which simply closes the illegitimate website. The colour of the button was also changed from blue to white.

Appearance The text *We sincerely advice that you do not proceed* was moved to immediately below the headline to be more easily detected by the user, and the part *[..]do not proceed* was italicized to be highlighted. Below, two lines of text say *This may be a 'phishing' website*, and *It may try to illegitimately get your personal information*. The *More info* link was move after the last sentence. A line was added to visually separate the texts from the links provided below them. The Aalto logo was introduced for the icon displayed in the toolbar and as a sign of trust and legitimacy in the banner. This choice was only temporary and was decided that it would have been replace with the final logo. Along with the modifications reviewed above, a few minor changes were made to line spacing of the text and the size of the *warning message*. The end result is depicted in Figure 10, that was the warning used in the usability tests 6.

In addition to the *warning message*, the *safe banner* indicating the safety of the website was evaluated in the cognitive walkthrough. At this stage, its design contained the text *Scanning complete*, and under it the smaller text *This website is safe* (see Fig. 11).

Based on the general requirement that the *safe banner* should be minimal, the suggestion was made that the banner should merely present the standard *correct* sign on a green background, with the addition of the hopefully upcoming program logo (Figure 12). We further decided to incorporate the logo to all the banners and the *loading icon* to make it clear to the user that they are generated by the same program.

Loading icon In the version of the *add-on* studied in the walkthrough, both the *warning message* and the *safe banner* were preceded by a yellow banner containing the text *Checking for phishing attacks* with three animated dots to indicate an ongoing processing (Figure 13).

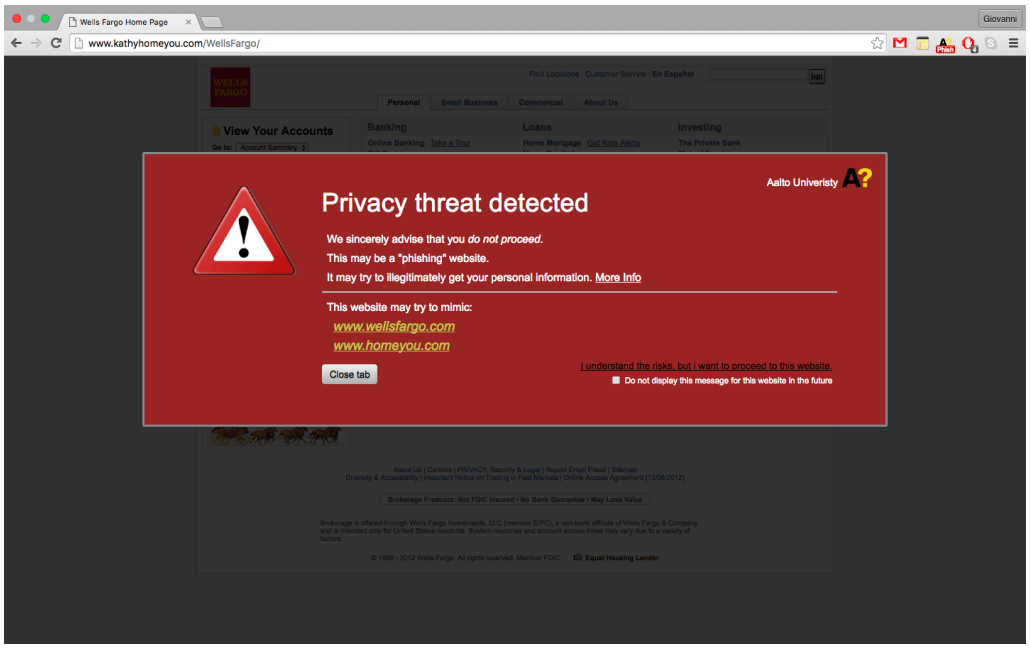


Figure 10: *Warning message* after the cognitive walkthrough.

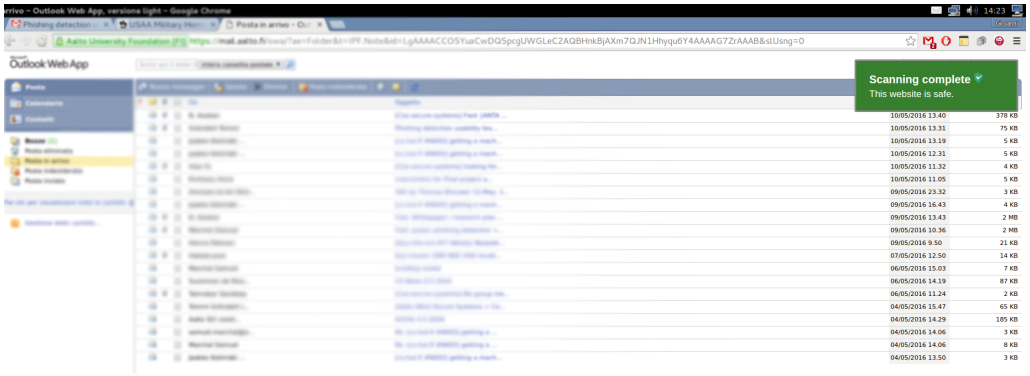


Figure 11: *Safe banner* evaluated in the cognitive walkthrough

However, the processing was too fast for the text to be readable for a first-time user, with the speed of the connection used in the walkthrough. Further, the term *phishing* belongs to the technical jargon of IT and security, and its meaning is not known by naive users. It was agreed upon that the yellow loading banner should be replaced with a semi-transparent logo of the program with an animation to indicate loading. In the usability tests the *loading icon*(Figure 14) was an animated circle..

5.5.3 Final design

Once the name *Off the Hook* was chosen and the logo designed, we modified the user interface consequently. In the *warning message* the Aalto university logo was substitute with the new developed logo for the program, depicting a fishing hook (Figure 15). This is in line with the name of the program: *Off the hook*, furthering

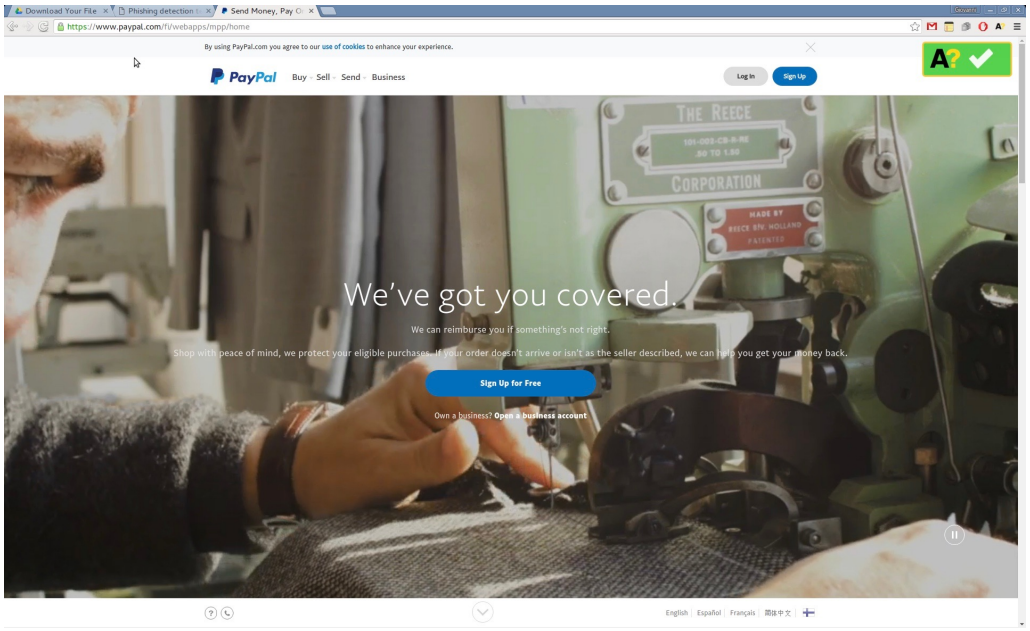


Figure 12: *Safe banner* after the cognitive walkthrough

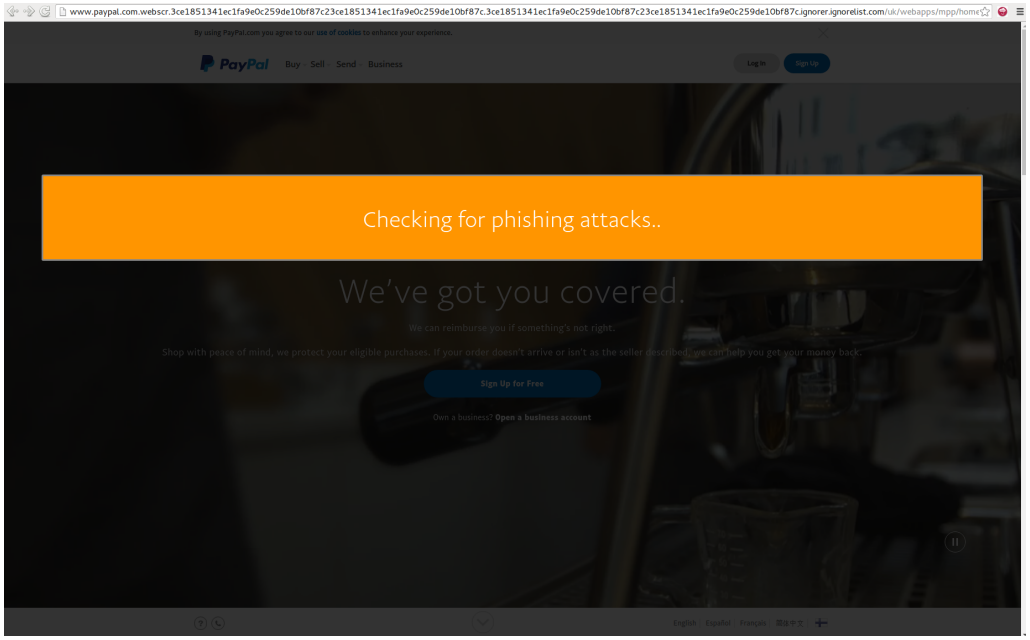


Figure 13: *Loading icon* evaluated in the cognitive walkthrough

the intuitive connection between the logo and the program. The name was chosen due to its high memorizability as an idiomatic phrase and its association to fishing.

The *loading icon* and the *safe banner* also incorporated the new logo, and the size of the latter was reduced (Figure 16). The loading animation was changed from a circling animation to a wave-like loop (Figure 17).

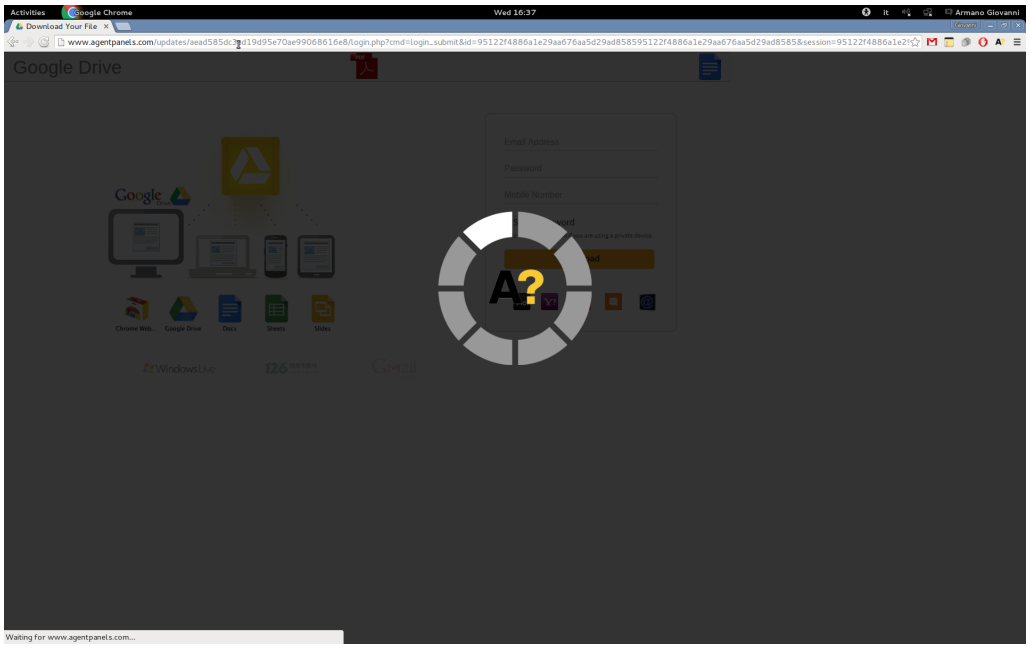


Figure 14: *Loading icon* after the cognitive walkthrough

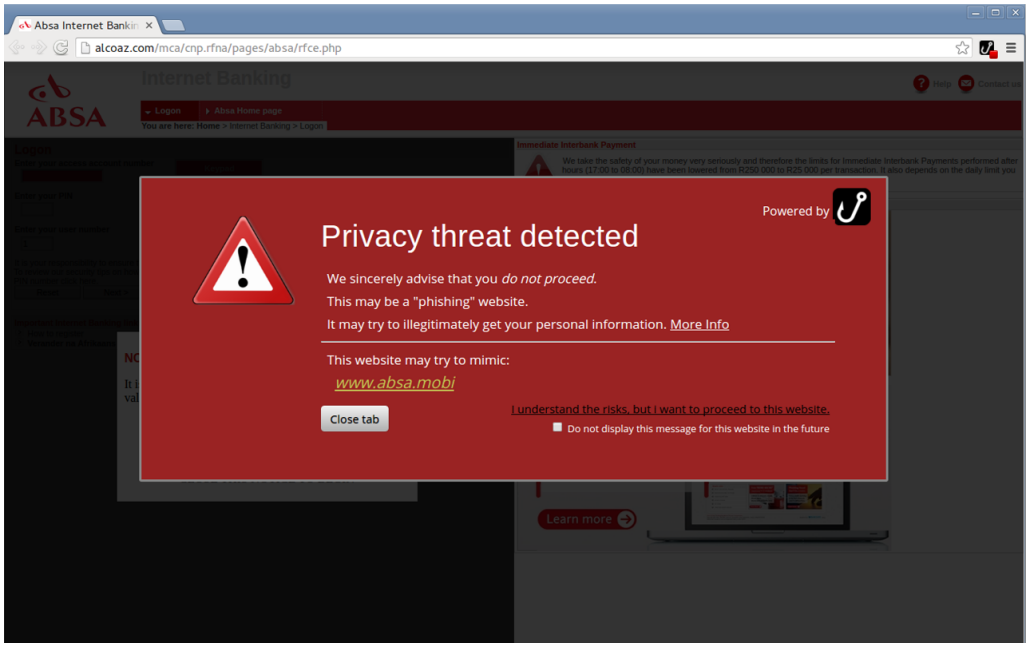


Figure 15: Final version of the *warning message*

5.5.4 Comparison to leading phishing warnings

The *warning message* used in the usability study (Figure 10) is compared to the phishing warnings of the two leading browsers, Firefox and Google Chrome. These are presented in Figures 18 and 19, respectively.

Evidently, the appearance of each banner is unique. All are red, indicating danger, but the shades differ, Chrome being the brightest and Firefox the darkest. The size

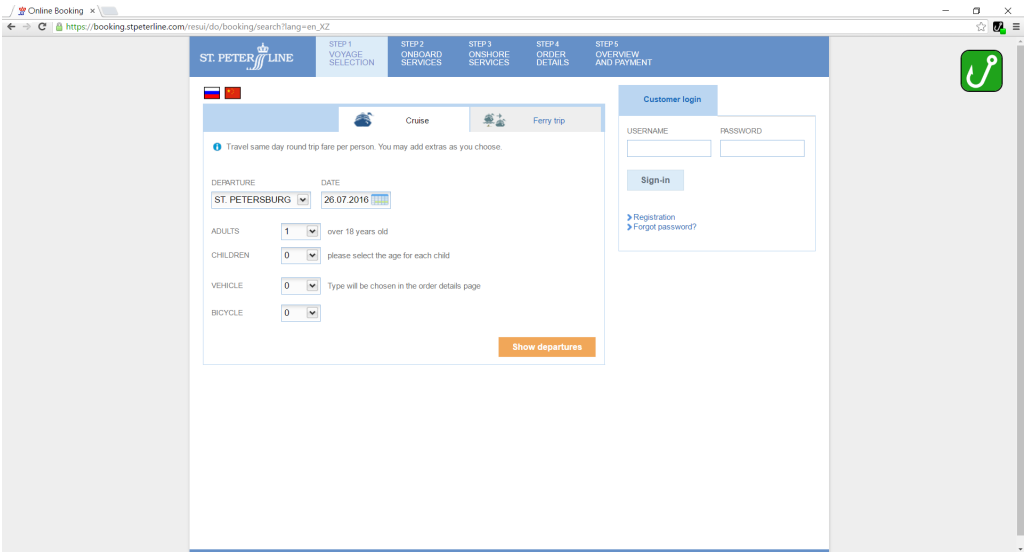


Figure 16: Final version of the *safe banner*

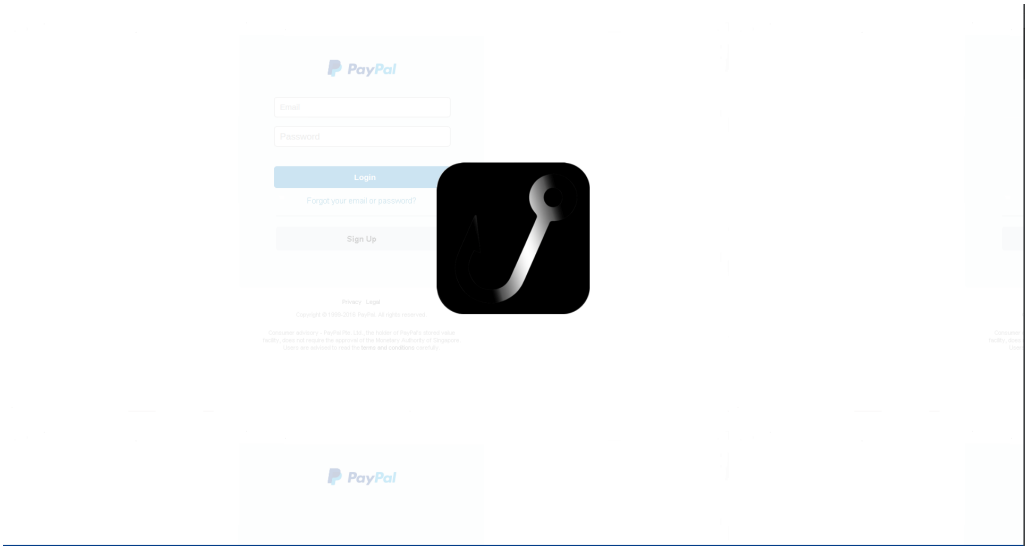


Figure 17: Final version of the *loading icon*

is another clear difference between the banners: Chrome's banner (although not the text) takes up the whole screen, and Firefox' is smaller than our banner despite of displaying more text. The positioning of the text and links also varies.

In addition to the differences in appearance, the banners present the user with somewhat varying information and different alternatives to continue. Chrome only offers two options: *Back to safety* and *Details*. Firefox offers both in different terminology (*Get me out of here* and *Why was this site blocked*), and additionally allows the user to ignore the warning and proceed to the website. Beside the similar three options, our banner further allows the user to enter the site detected by *Off the Hook* as the target of the phish. Hence, if the phishing site tried to imitate e.g. Adobe's website, the banner includes a link to the main site of Adobe. Such a

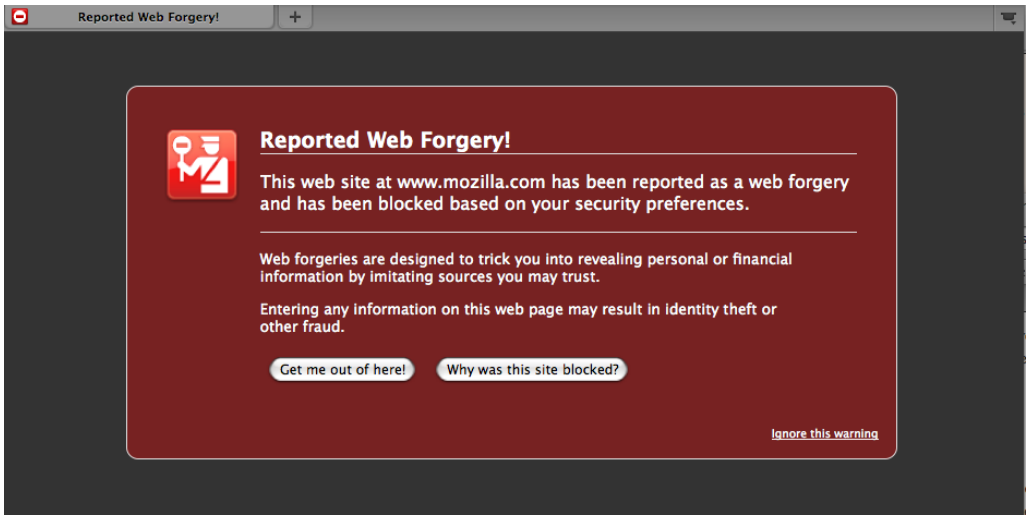


Figure 18: Firefox' phishing warning

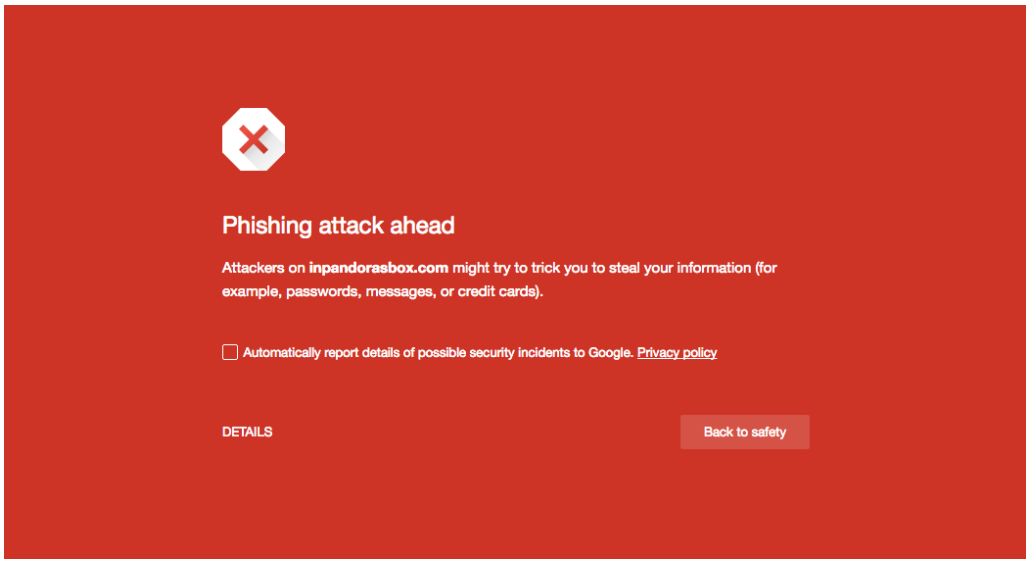


Figure 19: Google Chrome's phishing warning

functionality is abscent from both browser warnings.

Additionally, while both Firefox and our banner allow the user to ignore the warning and proceed to the website detected as a phish, our banner further gives the option of not showing the banner on particular websites in the future. This is achieved by an additional check box immediately below the option of ignoring the banner.

Concerning the text content, Chrome's message is the shortest and contains the least content, whereas Firefox gives more information. Our banner fits in between these, giving approximately the same amount of information as Firefox, but with shorter sentences and less technical terminology. The term *phishing* is only explicated in our banner; it is merely mentioned in Chrome and absent from Firefox. Thus, unlike the other banners, our warning both utilizes the term and defines it for people

unfamiliar with it.

Appearance-wise, among the main differences between our banner and both Chrome and Firefox is displaying the site detected as phishing semi-transparently in the background. Firefox only displays a gray background, and Chrome's red banner fills the entire screen. Other differences in the visual appearance of the banner are easily detectable from the images provided above.

Overall, our banner provides the user with as much information as the Firefox banner, and more alternatives to choose from. Nevertheless, it uses shorter sentences and minimal technical terminology in communicating the warning message.

6 Evaluation

This section presents the evaluation of the software described in Section 5. It is divided into five subsections, each addressing different requirements presented in Section 4. Section 6.1 analyzes the coverage in terms of web browsers and OSs. Section 6.2 evaluates the time and memory consumptions. Section 6.3 validates the new model and compares the data extracted from the add-on to the data extracted using Marchal et al. technique to prove the validity of the results. Section 6.4 and Section 6.5 present two usability studies that asses the understandability of the messages, the usefulness of functionalities that our software offers and the impact of the program on user experience.

6.1 Deployability

A key aspect during the evaluation of a software is its capability of being deployed in as many devices as possible. To reach a large number of user, we targeted the two most used browsers: Google Chrome and Mozilla Firefox. Table 9 depicts the web browsers usage in May 2016, it shows how the percentage of users that we are able to cover is close to 90%.

Chrome	**Firefox**	**IE**	**Safari**	**Opera**	**Others**
71,4 %	16,9 %	5,7 %	3,6 %	1,2 %	1,2%

Table 9: Browser usage from w3schools data (May 2016) [W3s]

Since our solution is built of two components, we need to maximize the coverage of both. The *background processes*, written in Python, have been packaged in stand-alone executable files in 5 different operating systems. These executable has been tested and run on 9 different OSs (Windows 8/8.1/10 x64, Ubuntu 12.04/14.04 x86 and x64, Fedora 20 x64, OS X El Captain). Though, considering the common root of Linux distributions, we can predict their compatibility with even more systems. The number of operating system supported and the high percentage of Internet users reached with this implementation addressed Requirement 1.

6.2 Performances

The performances analysis is composed of two parts. The first section describes the results in terms of time execution after the optimization of the new architecture. The second section estimates the reduction of the memory consumption.

6.2.1 Execution time

The core of this application resides in the machine learning algorithm that is used to discriminate between legitimate and phishing websites. The classification based on the feature vector is in the order of milliseconds. Unfortunately this is only a part of the process. To evaluate the webpage, the feature vector has to be built and, for the

phishing detector, this is the most time consuming task. In this section we evaluate the improvements described in Section 5.4.1.

We analyze the computation over 1320 samples collected during a two weeks period on an OS X machine (2,7 GHz Intel Core i5 processor and 16GB memory) using the Firefox version of the add-on. The webpages are divided into:

- 1200 legitimate webpages.
- 120 phishing webpages randomly collected from Phishtank.

The collected data is divided into three different groups and each of them refers to a particular part of the system execution:

- **Information collection:** this is the time that elapsed from the rendering of the page to the moment when the JSON object containing the page information is sent for analysis.
- **Target identification:** this is the time between the rendering of the page and the moment when the results are collected by the add-on.
- **Phishing detection:** the same time as the *target identification* calculated on the *phishing detection* result.

The overall time consumed in the computation can be calculated as the time needed for the *information collection* (1-4) plus the maximum between the *target identification* (5-7a) and *phishing detection* (5-7b). However, this measurement is not the most important for our purpose. The aim of the system is to prevent information disclosure, hence the time taken into account is the one during which we can stop user interaction with the malicious website. The interaction is blocked only if the phishing detector identifies the webpage as a phish. This means that the delay we take into account is the one reported in column (1-7b) of Table 10.

		(1-4)	**(5-7b)**	**(5-7a)**	**(1-7b)**	**(1-7a)**
Legitimate	**Average**	98	1891	276	1988	373
	Median	36	1174	98	**1223**	**141**
	Stdev	464	2485	581	2599	807
Phishing	**Average**	52	2185	162	2237	214
	Median	36	2011	91	**2037**	**130**
	Stdev	52	2048	275	2064	285
Overall	**Average**	94	1917	265	2011	359
	Median	36	1242	96	**1300**	**140**
	Stdev	443	2450	561	2556	775

Table 10: Processing time (milliseconds)

The time required for processing phishing websites is lower than the time required for processing legitimate websites, and phishing warnings are displayed in less than

a fifth of a second (130 ms). The overall median time of 0,14 seconds is 100 times smaller than the 14 seconds reported by Marchal et al. for scraping, building the features vector and perform the classification. The target of a phish is globally identified in 2 seconds or less after the page is loaded. In case of a contradictory decision of the *target identifier* regarding false positives of the *phishing detector*, the warning message is removed in around 1 second (1223 - 141 = 1082 ms). This result satisfies Requirement 3.

6.2.2 Memory consumption

From the earliest stage of the application, the amount of memory required was conspicuous [AMA16]. While the *dispatcher* was using 22 Mb and the *target identifier* 88 Mb, the *phishing detector* was around 300 Mb, which alone is more than the overall consumption of *Off the Hook*. The modification described in Section 5.4.2 led to the following reductions on the memory usage:

- **Import optimization:** The amount of saved resources can be estimated from the memory consumption of the *target identifier*, that has been reduced by 12 Mb.
- **Replacement of Selenium with the ad-hoc function:** The removal of Selenium allowed us to save around 60 Mb.
- **Removal of Alexa list:** the problems related to the 1 million most visited websites list were multiple. In the first instance, the file containing the list, a CSV file, was alone occupying more than 22 MB on disk. That had to be loaded in memory, consuming more than 1,2 seconds and the space required to store it in an appropriate data structure. The main goal of the list was to calculate a score, between 1 and 1,000,001, associated to the website ranking (1 - 1,000,000) or its absence in the list (1,000,001). In order to have a fast lookup in the structure a list was not a good option, because for each lookup the complexity is O(n) and the estimated time for a miss in that list was about 0,025 sec. This time alone does not constitute a huge problem, but for an average webpage, the URLs to check against that list are around 20, and the number can be ten times bigger. The best option is a dictionary, as an associative array with the string containing the domain as key and the rank as key. Using this approach the complexity is reduced to O(1). Even though the implementation of a dictionary in Python consumes a relatively small amount of memory, the size of the list carries a space occupation problem.

  Dictionaries are implementations of a hash table. An estimation of the memory used can be calculated based on the following assumptions:

  – An integer is a fixed-sized object, it contains a reference count, a type pointer and the actual integer. In total it typically takes 24 bytes on a 64bit architecture, not taking into account extra space possibly lost through alignment.

- A string object is variable-sized, which means it contains a reference count, type pointer, size information, space for the lazily calculated hash code, state information and a pointer to the dynamic content. In total it takes at least 60 bytes on 64bit system, not including space for the string itself.
- The dictionary itself consists of a number of buckets, each containing the hash code of the object currently stored (that is not predictable from the position of the bucket due to the collision resolution strategy used) a pointer to the key object, a pointer to the value object. In total it takes at least 24 bytes on 64bit architecture.
- The dictionary starts out with 8 empty buckets and is resized by doubling the number of entries whenever its capacity is reached.

In our scenario we have 1,000,000 strings with total approximate size of 22 MB and numbers from 1 to 1,000,000 associated with them in a 64 bit architecture. String/integer combinations:

$$1,000,000 * (60 + 24) bytes = 68,13\ MB$$

String contents:

$$22MB$$

Hash buckets (after resizing 17 times):

$$1,048,567 * 24\ bytes = 24MB$$

For a total estimate consumption for the Alexa structure of:

$$114MB$$

The structure optimization resulting from this work led to a large memory saving as shown in Fig. 20.

The component affected the most by these changes was the *phishing detector*. Even though the module's reduction was performed in even the *target identifier*, the removal of modules like Selenium and the Alexa list did not affect the latter, since these modules were only used in the *phishing detector*. The low memory consumption of 220 Mb shows that we address Requirement 2.

6.3 Reliability of decision

To highlight the differences between the model described in Section 2.3.2 and our new one, we used the same train and test sets. However, the improvements introduced in the architecture and the substitution of modules with function implemented ad-hoc raised a problem related to the data sources gathering method. The data sets used to asses the classification performances in Marchal et al. 2.3 work and in *Off the Hook* were gathered through a web scraper. This is in contrast with the final version of *Off*

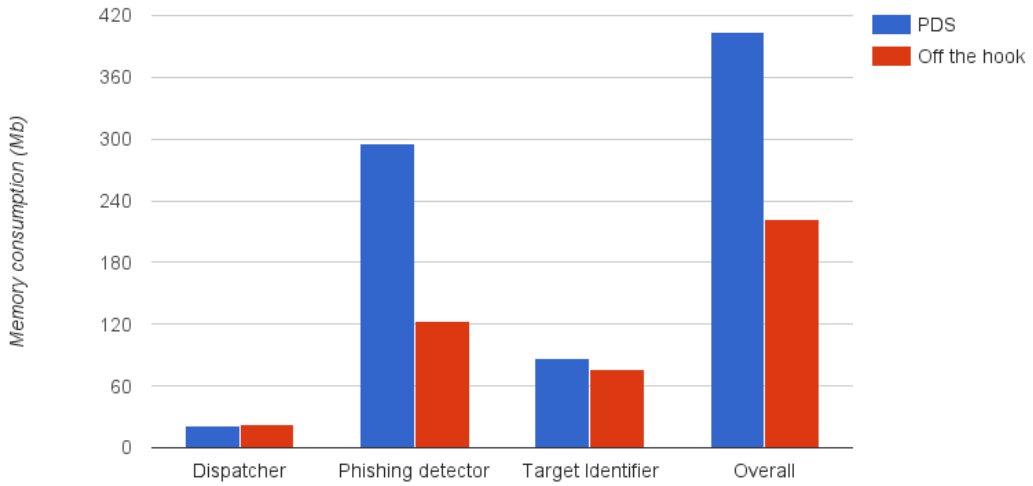


Figure 20: Comparative memory consumption

the Hook, in which the data are collected from the add-on and the collection method is slightly different. It is therefore important to analyze how big the differences are between the data collected from the two pieces of software to validate the results of the new model.

Differences in the collected data To attest this we used two lists of websites: the first one is a list of legitimate websites, while the second is composed of available phishing websites collected from Phishtank. The *data sources* needed for the classification were extracted in parallel from the scraper used in Marchal et al. work and *Off the Hook* add-on. We then performed two different comparisons:

- **Collected data:** we used a subset of *data sources* to estimate the differences between the two approaches. We describe in detail only the relevant differences encountered in the *text* and *logged links*, while for *starting URL, landing URL, redirection chain* and *title* we assume that the result are the same by virtue of being collected at the same time.

	Text			**Logged links**		
	Mean	**Median**	**Stdev**	**Mean**	**Median**	**Stdev**
Legitimate	0.0766	**0.0283**	0.1516	0.1017	**0.0503**	0.1388
Phishing	0.1547	**0.0866**	0.1965	0.1135	**0.0664**	0.1106

Table 11: Similarity of the extracted data

Table 11 shows the mean, median and standard deviation of the Hellinger Distance computed on the distribution of the words extracted from the *text* and the *mlds* extracted from the *logged links*. We used the *mlds* because a direct comparison was not possible due to the randomization in the creation of certain parts of the URL in phishing websites and the use of parameters in the query part of the URL in dynamic legitimate websites. This data refers to a sample of 112 legitimate websites and 72 phishing websites. We can see how the mean distance for phishing websites is less than 0,09 for the *text* and around 0,06 for the *logged links* attesting that the two sets are very similar. The values for legitimate websites are even lower and reaches 0,028 for the words contained in the *text*.

- **Classification results:** We went beyond the simple analysis of the *data source* and compared the final results returned by the classification using the the two different data sets collected in the previous point and the model developed for *Off the Hook*.

	Mean	**Median**	**Stdev**
Legitimate	0.0020	**0.0002**	0.0061
Phishing	0.0536	**0.0115**	0.0820

Table 12: Similarity of confidence values

The results presented in Table 11 show the mean, median and standard deviation of the absolute difference between the result of the classification performed on the two datasets. The values obtained for phishing websites are comparable for the two data sets with only 0,015 median variation. Approximately the same results are obtain for legitimate websites. With the exception of the small variation in the extracted data, the results of the classification for every instance, with a threshold of 0,7, were the same. This allowed us to address Requirement 4.

Evaluation of the new model The results of the computation over datasets of 6 different languages are reported in Table 13. These values can be compared with the results showed in Table 6 (Section 2.3.7).

Language	**Precision**	**Recall**	**F1-score**	**FP Rate**	**Accuracy**
English	0.937	0.928	0.932	0.0008	0.998
French	0.960	0.928	0.944	0.0047	0.988
German	0.981	0.928	0.954	0.0022	0.990
Portuguese	0.968	0.928	0.948	0.0037	0.989
Italian	0.966	0.928	0.947	0.0039	0.989
Spanish	0.958	0.928	0.943	0.0049	0.988

Table 13: Detailed accuracy evaluation for the new model

These performances were obtained with a classification threshold of Gradient Boosting set to 0,7. The data in the table shows how precision is high and varies from 0,93 to 0,98. The recall is significant as well (0.92). The false positive rate for all the languages is below 0,0049, which is a remarkable achievement. These values, though slightly below the ones illustrated in Marchal et al., are the result of the strong improvement on the execution time and memory consumption. Furthermore, they retrace the same trend shown in the previous work where the results for the English dataset are below the other languages for precision, but obtain the lowest FP rate. It is worth underlining that for the German dataset we were able to obtain the same performance as before, while for Portuguese both the precision and the FP Rate are better. These results are comparable to the ones described in Marchal et al. work and thus show that we address Requirement 4.

6.4 Add-on usability

The purpose of the first usability study was to investigate the user experience and intuitive understandability of the banners. The main questions studied were:

- Do users comprehended the content of the banners?
- How do users choose to continue after being shown the phishing warning?
- What opinions do users have of the content of the banners?
- Do users have suggestions of how the banners should be modified?
- Do users prefer our banner to Firefox' phishing warning?

With the exception of the last question, the test was thus qualitative, providing evidence for how users react to the functioning of the application. Its main motivation was to find potential usability problems, see which preferences users had concerning how to proceed when given alternatives, and test whether there was a preference pattern between our *warning message* and Firefox's banner.

We conducted a one-hour usability study on 10 users (excluding the pilot). Most users were recruited via an email sent to student mailing lists both in Aalto university and University of Helsinki, and some by contacting them in person.

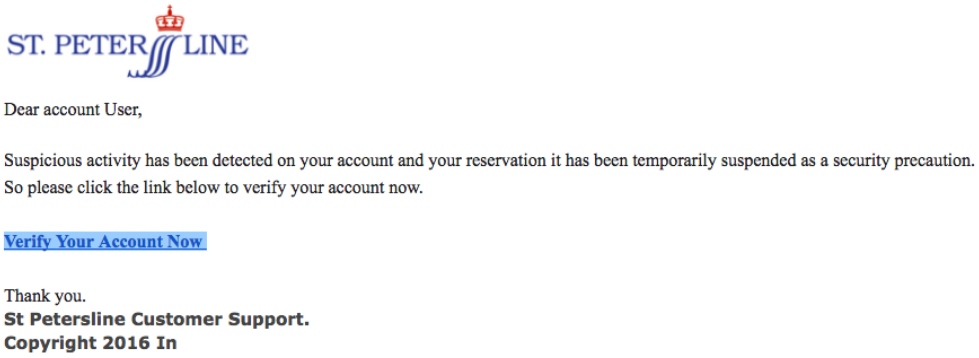
All the recruited users were between 20 and 30 years old. Seven users were male and three female, and all of them had a university degree or were university students at the time of the test. Half of the users had Windows as their main operating system. Five used Chrome as their main browser, followed by four that use Firefox and one Safari user. In daily laptop use, there was variance between under one hour and over seven hours, but for the majority laptop usage exceed 5 hours. Most users used the laptop primarily for either studying or work (some for both). Daily Internet use was just a portion of total amount of time spent on the machine, with a concentration between one and four hours.

6.4.1 Pilot study

Prior to the usability study, a pilot test was conducted, resulting in refinement of the test protocol and minor changes to the *warning message's* design. The pilot test was not incorporated to the results of the usability test. The largest modification the pilot study motivated was changing the Google link to a "Close tab" button. This was strongly argued by the pilot user to improve the user interface, and received further support from the cognitive walkthrough (Section 5.5.2).

6.4.2 Test description

The test was conducted on a university laptop, with Windows 8.1 and Google Chrome, and lasted one hour. An email account was created, containing fake emails from real companies, e.g. Facebook, Apple, Adobe, and St. Petersline (Figure 21). Each email contained text claiming that a privacy issue had been detected, and the user should enter the website to verify their personal information. At the bottom of the emails was a link that the phishing email claimed to lead to a registration site on the website of the company.


ST. PETER LINE

Dear account User,

Suspicious activity has been detected on your account and your reservation it has been temporarily suspended as a security precaution.
So please click the link below to verify your account now.

Verify Your Account Now

Thank you.
St Petersline Customer Support.
Copyright 2016 In

Figure 21: St. Petersline phishing email used in the study

The test can be divided into three parts based on the banners displayed to the user.

Green icon: First, the user was asked to open a link in the email that result in the *green icon*. This was done to familiarize the user with the test setting, and also to check whether she noticed the presence of the icon on the top right in the browser. To inquire this, the user was asked whether they see anything unusual or surprising about the site. None mentioned the icon at this stage.

Warning banner: Next, the user was instructed to open a second link that lead the *add-on* to first display the *loading icon* (Figure 14) and then the *warning message*(Figure 10). The user was then asked several questions, which were discussed in detail. These were (with minor modifications between tests depending on what the users focused on):

- Why do you think this banner came up?
- What is the banner notifying you of?
- What was the first thing that captured your attention?
- Did you read all the text?
- What order did you read the text in?
- What does each part of the banner mean, and why is it relevant (or is it irrelevant)?
- If this were to happen to you, what would you do?
- What continuation methods does the banner offer you? Would you choose any of them?
- After reading the text, do you understand what it is telling you?
- Is there something missing from the text that should be added?
- Is there something on the banner that you think should be removed?

At this stage, if the user had not already noticed the *add-on* icon on the top right of the browser displaying a patch of red and the hover-text *This website is unsafe*, this was pointed out to her by asking whether there is something related to the banner in the browser. Next, the user was shown a picture of the Firefox phishing warning (Figure 18), and asked to compare the two banners, with the following questions being asked:

- Do the banners warn of the same thing or is there a difference?
- What does the text on the Firefox banner say, and are there substantial or stylistic differences between it and our banner?
- Is there something that is present on one banner but not in the other?
- Firefox' banner lacks the *Do not display this message for this website in the future* check box; would its presence improve the banner?
- Firefox' banner lacks the partial visibility of the website in the background; would its presence improve the banner?
- Firefox' banner lacks the link to the detected target website; would its presence improve the banner?
- In terms of appearance, which banner looks better and why?
- Which banner looks more trustworthy?
- Which banner is better overall?

Loading icon and Safe banner The user opened a third email and the link it contained, leading to the appearance of first the *loading icon* and then the *safe banner* (Figure 12) with the check mark on the top right of the screen. The user was first asked whether she understood what happened, and then her opinion on the visual appearance, placement, duration and overall usefulness of the *safe banner*. The following questions were asked:

- Did you understand what happened?
- Was the banner present for an appropriate time?
- Was the banner located appropriately?
- Was the size of the banner appropriate?
- Should the banner contain more information?
- It is clear that the banner is produced by the same program as the red phishing warning?
- Does the banner look good?
- Is there something you would like to be changed on the banner?
- Why do you think the banner appears on some sites but not others?

Further, it was inquired whether the users found the *safe banner* useful at all, since its information content is in principle redundant with that of the *add-on* icon on the browser. This question divided the users.

Since the *loading icon* was a prototype, its appearance was not a focus of the study. However, its placement in the center of the screen and its blocking of access to the website were discussed. The main issue here was whether the user liked the current situation where the website is blocked by the loading, or whether she would prefer to access the website and the loading to be done on the side. After the test, the users filled out a questionnaire A.2 containing similar questions to the ones discussed freely during the test.

6.4.3 Warning message

A significant majority of users understood what the banner was notifying them of, although some were initially confused as to what the warning was about (e.g. a virus instead of phishing). In the questionnaire it was asked how the users initially interpreted the banner, are results displayed in Table 14.

One user chose both *Warning of a scam* and *Warning of a website that is illegal to visit*, hence the sum of answers being 11 instead of 10. The two users who chose *Other* had understood the banner correctly as a phishing warning, indicated by their freely written responses. Hence, their choosing *Other* instead of the correct alternative (*Warning of a scam*) should not be interpreted as them having misunderstood the banner.

Question	# users
Warning of a detected virus	2
Warning of a scam	6
Warning of a website that is illegal to visit	1
Warning of hardware malfunction	0
Warning of browser malfunction	0
Other	2

Table 14: Initial interpretation of the *warning message*

Most users were familiar with the concept of phishing, although for many the understanding was only superficial. There were no major misunderstandings of the textual content of the *warning message*. However, all users did not initially read all the content of the banner, but some jumped straight from the headline to one of the links. All alternatives were not easily detected in the warning banner by all users. *Close tab* was easily detected, but i.e. *More info* was confused in the text.

The yellow link to the target site was noticed easily, but its correct interpretation was more laborious. Some users did not initially understand why it was there, but after reading the text on the banner, its purpose became relatively clear to all. Some mentioned that its purpose in the banner was less than obvious, since it is unlikely the user would want to go to a site a phishing site is trying to mimic, given that the link to the phishing site is probably illegitimate. On the other hand, the link helped many users understand the message of the banner more quickly. Showing the target website in the warning gives the user more information about the status of the site as a phish, and makes it clear for the user that dismiss the banner would not lead her to the legitimate website.

Every user chose to close the tab, the majority (6) from the browser and the rest (4) from the banner. Many indicated that while some of the other links might be useful as well, the reliability of the banner is unclear. Thus, selecting a link on the banner might lead to another illegitimate site. This also motivated many users to close the tab from the browser rather than the banner, since the functioning of the browser is known ahead of time.

Due to the frequently mentioned *unprofessional* look of the banner, many felt that the links may not necessarily be trustworthy. This was one of the major reasons why many users would have closed the tab from the browser rather than the banner (the other being habituality). Users generally treated the *More info* link as useful, but differed somewhat on how intuitive they felt the page it took them to was i.e. Phistank [Phi]. Some thought it would have taken them to a manual related to the *add-on* rather than an external site.

The text was generally treated as easy to understand and clear but a few commented that the order of sentences should be changed.

After the test, the users filled a questionnaire (Appendix A.2) inquiring similar questions as the ones asked during the test, with pre-given answer options, typically from *Strongly agree* to *Strongly disagree* on a five-point scale. The sentences evaluated of the red banner are given in Table 15, in which, *Agree* comprises both *Strongly*

agree and *Agree* in the questionnaire, and *Disagree* comprises both *Disagree* and *Strongly disagree*.

Sentences	**Agree**	**Neutral**	**Disagree**
The text was easy to understand.	8	0	2
The text was quick to read.	6	2	2
It was immediately clear what the banner notified me of.	7	2	1
The colour of the banner is appropriate.	9	0	1
The banner looks professional.	7	0	3
I would find the program displaying the banner useful on my computer.	9	1	0
If the program was preinstalled on my computer, I would uninstall it.	1	5	4

Table 15: Evaluation of the warning message

The questionnaire also asked which of the links on the *warning message* were useful, on a five-point scale from *Completely useless* to *Very useful*. In Table 16, results are collected such that *Completely useless* and *Somewhat useless* are pooled together as well as *Somewhat useful* and *Very useful*.

Option	**Useful**	**Neutral**	**Useless**
More Info	8	1	1
This website may try to mimic:	8	1	1
Close tab	7	0	3
I understand the risks, but I want to proceed to this website	9	1	0
Do not display this message for this website in the future	6	3	1

Table 16: Evaluation of the continuation alternatives in the warning message

While most continuation options were rated as useful on average, it is noticeable that the *Close tab* link's practical value was brought to question by many due to its redundancy with closing the tab from the browser. Nevertheless, it is important to have a closing button on the *warning message*, and in the test almost half of the users chose it (4). The results indicate that the continuation options are useful, and none stood out as unnecessary for the users tested. Furthermore, since our software is the first one that presents continuation links to the target websites, it is valuable that the majority of the users considered it as useful. There results address Requirements 5 and 6.

6.4.4 Comparison to the Firefox banner

In the questionnaire, seven out of ten users overall preferred our warning to the Firefox one. Additionally, on all the individual questions (1.8.1 - 1.8.8, see Appendix A.2)

there was either a preference of our warning or a tie between users preferring it and these preferring Firefox' version (Figure 18). While our sample of ten is too small for the results to be statistically significant, no overall preference of the Firefox banner over our warning message was detected by the tests. These result might be significant if we take into account the familiarity that users might have for Firefox's banner. Their previous experience with the message might bias the preferences towards it. On the qualitative side, among the main differences the users detected between the banners were the following:

- Firefox' banner utilizes more technical lingo and is thus more difficult to understand. Especially the term *web forgery* it uses instead of *phishing* was considered unintuitive by many users
- A majority of users thought the presence of the link to the target on our add-on was useful, and its lack in the Firefox banner was therefore a defect. However, a number of users also said its presence made no difference to them.
- The check box *Do not display this banner for this website in the future* divided the users into two. A subset of users thought it was useful and hence a plus for our warning message against Firefox' banner; but others thought it would be better for the banner to appear each time a potential phishing site is loaded, and its accidental removal from all future visits to the site to be too much of a risk.
- Our banner was often considered visually less professional than the Firefox banner. Firefox' warning was deemed by many to be more *authentic* in appearance, giving a stronger impression of reliability. Conceivably, this may have been partly due to recognizability or habituation, as three users had Firefox as their main browser, and one used both Firefox and Chrome.

6.4.5 Safe banner

Users were divided on whether they found the *safe banner* useful at all, given that the toolbar icon already displays the relevant information (*This website is safe*). Some thought it was nice, but others found it unnecessary and redundant.

A notifiable negative result was, however, that no user understood why some safe sites brought up the *safe banner* but others did not. The confusion resulting from the banner appearing on some safe sites but not others was a common argument for reducing the safety indication entirely to the icon.

Most users found it acceptable that the loading icon stops the user from entering the site, but some remarked that this may be annoying if it happened often and lasted long. However, the low false positive rate shown in Section 5.4.3 shows that the banner should appear on less than the 0,5% of the visited websites.

6.5 Impact on user experience

A second user study was conducted to investigate whether the presence of the application has a negative impact on user experience in general, and how users react to the banners if they are exposed to them during normal laptop use.

6.5.1 Test description

The second study was conducted in the course of two weeks by installing the application on the user's own laptop for one week and a fake version of it for another week. The *fake version* was a browser add-on that was displaying the same icon as the real version without implementing any functionalities. The ordering of the real and fake application was divided 50–50 between users. Prior to the test, the users were only told the information provided below:

- Two versions of the program are installed for one week each.
- The program looks up certain information about the website visited.
- No personal information is collected.
- No specific actions are needed from of the user.

Users were not aware of the purpose of the software and no information about any security related application was provided.

The same questionnaire was filled after both weeks, asking the user to choose between *Strongly disagree* and *Strongly agree* on a five-point scale as reactions to the following statements:

- My browsing has slowed down.
- I have been annoyed by the presence of the program.
- My computer has slowed down in general.
- The program has interfered with my browsing.
- The program has interfered with my use of the laptop in general.
- I have not noticed interaction between the program and my browser.
- Use of my laptop has been as enjoyable as before.
- The program has prevented me from doing something I wanted to do.
- The browser has crashed more often than usual.
- Other programs have crashed more often than usual.

We will refer to this as the *general questionnaire* A.3. By comparing the results from the two weeks we can see whether the application had an effect on user experience. Each user's answer to the week with the fake application can be treated as a baseline for their general level of annoyance and (mistaken) judgements of the computer slowing down when they know something novel has been installed. Any significant deviance from this when the real application was present is thus potential evidence for its negative effects.

There was further space in the questionnaire for freely written qualitative answers, which were requested if the user agreed with a unfavourable statement (i.e. *I have been annoyed by the presence of the program*) or disagreed with a favourable statement (i.e. *Use of my laptop has been as enjoyable as before*). This is important, since it reveals whether the experienced discomfort was due to something the application could have caused or was only coincidentally present during the week, and further, if the problems were caused by the application, whether they were more psychological or resulted from actual technical issues.

At the end of the study, a second questionnaire A.4 was filled, asking whether the users remembered having been exposed to any of the banners during the last two weeks. We will denote it as the *final questionnaire*. If they had seen the banner(s), they answered the same questions that were asked in the lab test questionnaire. Finally, their prior familiarity with the concept of phishing was inquired, as well as whether they had received any phishing warnings during the last two weeks.

There were 17 users altogether, ten female and seven male. Ten users were 20–24 years old, a few were 25–29 or 30–34, and one over 50. All were either university students or had a university degree. All used the laptop daily for surfing the Internet, most c.a. 3–5 hours daily. In both laptop use and Internet surfing there was variation between 1–3 hours and over 7 hours among the users, with an approximate normal distribution. Eight users had Windows as their main operating system, five used OSX and four Linux. Studying, work and personal use were approximately tied in popularity as the main reasons for laptop use. Seven users had Firefox as their main browser, and ten used Chrome.

6.5.2 General questionnaire

All users filled out the general questionnaire both after using the fake and the real version of *Off the Hook*. Of course, if the user was exposed to the loading screen and/or either of the banners during the two weeks, there was an impact on her laptop use, possibly generating annoyance or interference with various tasks. However, *Off the Hook* was not supposed to cause the browser to slow down significantly, or to increase the chance of the browser or other programs crashing. Thus, our null hypotheses were:

- Among people who did not see either banner, no difference should exist between any answers to the general questionnaire of the real app and of the fake app.
- There is no difference between the real and the fake app with respect to browsing speed.

- There is no difference between the real and the fake app with respect to programs crashing.
- *Off the Hook* does not negatively impact the enjoyability of laptop use.

The answers to the general questionnaires give information about the extent to which *Off the Hook* can be estimated to cause interference with the user's regular laptop use.

The null hypothesis was, for each question individually, that user's answers would be the same after having used the fake app and the real app, respectively. The statistical test that we perform, *Wilcoxon signed ranks test*, gives a *p-value*, which is the probability of getting a result which deviates from this null hypothesis as much or more as the actual result. Standardly, the p-value of 0,05 is used as the benchmark of statistical significance. Thus, if the probability of receiving a result that deviates from the null hypothesis as much as or more than the actual result is less than 0,05 (i.e. 1 out of 20), the null hypothesis is rejected.

In interpreting the results it is important to remember that many users reported *Off the Hook* having given the phishing warning on legitimate sites, such as on-line banks or email servers. Such false warnings were the main source of annoyance reported in the freely written qualitative section in the general questionnaire.

There was no significant change between the real and the fake app in how often the browser or other programs were reported to have crashed.In contrast, a difference was noticeable in the reported slowing down of the browser. When the fake app was installed, only one user agreed with the claim *My browsing has slowed down* and one also agreed with *My computer has slowed down in general*. In contrast, seven agreed with *My browsing has slowed down* after having the real *Off the Hook* installed, and three with *My computer has slowed down in general*. The first of these yielded a statistically significant result ($p < 0{,}05$), although the second did not ($p \approx 0{,}12$). The results indicate that there is a real possibility that *Off the Hook* did in fact slow down browsing, although the same cannot be said of the speed of the laptop itself.

Answers between the two groups for both *The program has interfered with my browsing* and *The program has interfered with my use of the laptop in general* differed significantly ($p < 0{,}05$), but only among those who had been exposed to the red warning. No comparable effect was found among those who had not seen the warning. This result is unsurprising.

Concerning general enjoyability of laptop use, the relevant questions here were *I have been annoyed by the presence of the program* and *Use of my laptop has been as enjoyable as before*. The Wilcoxon signed ranks test gives a significant result for both ($p < 0{,}05$), resulting in the rejection of the null hypothesis: *Off the Hook* seemed to have a negative impact on the enjoyability of laptop use. In the latter of these questions there is even a difference between those who were not exposed to the warning banner at all, although the numbers are two small here to make claims of statistical significance. Due to the relatively large amount of interference that users experienced, especially on legitimate sites, the result is understandable. Further, the loading screen and/or the green banner may create annoyance simply by taking time, and not being transparent enough to make the user understand what is happening.

Despite the large number of users who had experienced some interference from *Off the Hook*, only one user agreed with the statement *The program has prevented me from doing something I wanted to do*, and no significant difference was found between the real and fake app in this regard. Furthermore, the user who agreed with the statement reported that this effect had been psychological, i.e. the user had chosen not to engage in an activity due to the presence of the app rather than the app actually preventing anything. Unsurprisingly, the effect was the same in both the real and the fake app. Thus, the users were able to disregard the banner if needed, and no interference was fatal to other endeavours, apart from possible psychological effects.

Off the Hook had no significant impact on the reported frequency of crashing, either of the browser or of other programs.

In the qualitative section of the questionnaire the most frequent comments involved the phishing warning being presented on sites the user knew to be legitimate, generating annoyance.Some slowing down or crashing was also reported, and for a few users *Off the Hook* itself crashed or (reportedly) interfered with the start-up of other programs.

In summary, of the initial null hypotheses the only one that was not rejected is that *Off the Hook* had no impact on crashing, either of the browser or of other programs. In contrast, reported slowing down of browsing (although not the laptop in general) as well as increased discomfort were both attested. While interference by *Off the Hook* was frequent, its presence was not a problem for normal on-line interactions.

6.5.3 Final questionnaire

In the final questionnaire A.4, all users were asked the following questions:

- During the last two weeks have you seen this banner?
  - The red warning.
  - The loading screen.
  - The green banner.
- Prior to the two week study, were you familiar with the concept of *phishing*?
  - If yes, explain briefly how you understand what *phishing* means.
- During the two-week study:
  - Have you received e-mail you have thought to be phishing?
  - Have you visited websites you have thought to be phishing?
  - Have you received a warning about phishing?
- If you have received a warning about phishing during the last two weeks, which of the following was it?

- The *Off the Hook* warning (referred by figure number in the questionnaire)
- Another warning
- I have received both the *Off the Hook* warning and another warning.

Three users were unfamiliar with the concept of phishing. The rest knew what it means, and could explain it (approximately) correctly. No user claimed to know what phishing is but explained it mistakenly.

Half of the users (9) had seen the phishing warning. No user had seen only another phishing warning instead of *Off the Hook*, although one had seen another warning in addition to *Off the Hook*. All users who had seen either the warning or the green banner also reported having seen the loading screen, as expected due to its large size.

The users who were exposed to the red warning banner answered to a questionnaire concerning their opinions about the banners' content and look. The questionnaire also inquired how they proceeded after seeing the banner. Importantly, a majority of users had chosen *I understand the risks but I want to proceed to this website.* This either means that the banner had mistakenly been displayed on a site the user knew was safe, or that the user did not find the banner trustworthy or important. The first issue has to do with its computational success, and the second with the trust users give to the program. Both raise concerns about *Off the Hook*'s usability.

A significant majority of users understood that the banner warned of a scam, although a few chose the false alternative *Warning of a website that is illegal to visit* instead. While this choice was rare, its presence nevertheless indicates of the possibility of a mistaken interpretation. It is, of course, likely that it would lead to effectively the same outcome as the correct interpretation: considering the website illegitimate and not wanting to enter it.

On average, the banner received a generally positive rating on text content and visual appearance. A majority of users agreed with all of the following the claims: *The text was easy to understand*, *The text was quick to read*, *It was immediately clear what the banner notified me of*, *The colour of the banner is appropriate*, and *The banner looks professional*, although less so with the last claim than the others. On the other hand, there was less positive consensus about the usefulness of the application, as users were on average neutral on both whether they found the application useful, and whether they would uninstall it if it was preinstalled on their computer.

All the continuation links were considered useful on average, although the least so for *More info.* Interestingly, the link judged to be most useful was *Do not display this message for this website in the future*, closely followed by *I understand the risks but I want to proceed to this website.* Both indicate of the frequency of instances when users wanted to get rid of the banner and proceed to the website that brought it up. This is in line with the result that *I understand the risks[..]* was the most frequently chosen link. A significant amount of such instances were, based on other answers, due to the banner having falsely appeared on legitimate sites.

Users who remembered seeing the green banner were also asked the same questions of it as in the other usability test. All such users had seen it easily, and most did not consider it annoying. Most users felt more safe about the website due to the green

banner, but two agreed with the claim *The banner made me doubt the safety of the website*. However, opinions on its ease of understanding varied, and some considered it to be difficult to interpret. This was most likely due to its lack of information content, which is supported by a majority of users preferring more information on it. This is an interesting result, especially in comparison to the opposite result from the lab test, where most users saw the green banner as redundant. Likely, this may partly be because, unlike in the lab test, not all users had been exposed to the red banner prior to the green banner, making the latter difficult to interpret as related to phishing. Another clear difference to the lab test was that a majority would have preferred more information on the green banner. However, a slight majority of users still found the banner unnecessary.

In conclusion, half of the users had seen the red warning, which overall received a positive rating in terms of both content and appearance. In this respect the results are similar to those of the lab test. However, while in the lab test every user closed the tab, here most users had chosen to ignore the warning, probably due to it appearing on a legitimate site. There was also a corresponding difference in how useful the different continuation links were judged to be, ignoring the warning having a priority over the others. The green banner had mostly made users feel safer about the website, but its understandability was deemed much lower than in the lab test, likely due to it having appeared to users who had never been exposed to the red warning. These conclusion, impacts Requirements 4, 5 and 6.

7 Conclusion

This document presented the main contributions in the implementation of a client-side software. We summarize the contributions and draw some paths for possible future work.

7.1 Summary of contributions

We presented *Off the Hook*, a client-side privacy friendly software that performs real-time analysis of webpages while the user is browsing thanks to its integration in the browser through an add-on. We reached a high level of deployability thanks to its compatibility with more than 10 operating systems and two web browsers that cover almost 90% of Internet users. It is implemented entirely on client-side and preserves users' privacy. Although, with the explicit consent of the user, it can be used to collect precious information on phishing websites and users' feedback. The classification based on the information displayed in the webpage prevents the risk of adaptive attacks and gives the user the same level of protection as blacklist approaches while overcoming their delay problem. It reduces the memory consumption of the techniques that are at its basis by almost 50%, preventing high resource drain. It has the merit of reducing the time needed for the extraction and feature computation from the 14 seconds of Marchal et al. to an overall median time of 0,14 seconds. While the time has drastically decreased, the data collected is virtually the same, considering that no instance has been misclassified during experiments.

The qualitative results gathered from the usability study verify that our user interface is well designed and helps to understand the phishing threat, even for first-time users. Furthermore, the presence of a continuation link to the target website has the dual advantage of helping users to trust the warning message and providing a useful alternative to closing the tab or proceeding to the webpage. The second user study attested the improvement of the performance of the *background processes*. In fact, there was no statistically significant effect on general slowing down in laptop usage. However, there was a significant portion of users that reported the slowing down of browsing. This aspect was not noticed in the development process. It can be due to the interference free environment in which the add-on was tested or the higher performance of the systems used during development.

7.2 Limitations

The implementation of a client-side solution, together with the advantages related to the protection of privacy, brings a limitation in its deployment for devices with a reduced amount of resources such as smartphones. For such devices, the lower performance of the processors might prevent an efficient execution of the code. Furthermore, for Android devices, the phishing detector and the target identifier should be developed in Java and executed as services (processes that run in the background) reintroducing the problem of the reliability of decision. It might also lead, during browsing, to a large battery consumption and a reduction of the memory

available for other processes.

A second limitation is related to the incorrect identification as a phish of some payment or webmail systems. Such webpages are often used under the domain of certain e-commerces or companies the include services such as those offered by Mastercard or Outlook. The anatomy of such websites is similar to that encountered in phishing websites. Their content is loaded from an external website, and in the page several references point to it (i.e. are powered by *brand name*). If the webpage has no references to the domain which hosts it, it might be identified as a phish.

A third limitation is related to the way in which the data is collected from the webpage, which differs between the add-on and the scraper. Although we showed that none of the tested instances were misclassified, it is something that we cannot assert for sure. Misclassification may also occur between different browsers and depend on the plug-in installed in the browser. If they download or make requests while performing their activity, the classification might be biased.

7.3 Future work

The initial requirements of this work were mainly met. However, the possible problems raised by the user studies can be subjects of future work.

The influence on the browsing experience should be decreased until it is close or equal to zero. This can be done through the collection of the data in a background thread. This solution was not taken into account before, because, the user experience perceived in the testing machines were not impacted by this part of the computation.

A second aspect of future investigation is the development of the application on mobile devices. The number of smartphones and tablets that are sold every year is increasing, and for this reason it would be interesting to include it for these devices. This introduces further challenges in performance optimization, especially concerning memory usage and battery lifetime. In this scenario, a solution based on a server-side computation might be the best choice. This server should receive the features vector computed in the smartphone, perform the classification and return the result. However, this approach might arise privacy concerns because it force the user to share the data regarding the visited websites. It will also increase the delay in the computation of the result and the Internet usage for sending the features vector and receive back the result.

Finally, to overcome the limitation related to the misclassification of some websites, a global whitelist that contains verified legitimate websites can be included to the software. This list might be pushed directly to the users every time a new entry is added. This list might also be composed from users' feedback through a function that allows users to signal the legitimacy of a webpage. This advertised URL can go through further controls and, if its legitimacy is attested, can be added to the global whitelist. However, the abuse of the whitelist system might lead phishers to target the websites that are included in the list.

References

[AF13] Devdatta Akhawe and Adrienne Porter Felt. Alice in warningland: A large-scale field study of browser security warning effectiveness. In *Proceedings of the 22nd USENIX Conference on Security*, pages 257–272, 2013.

[AJPN15] Pieter Agten, Wouter Joosen, Frank Piessens, and Nick Nikiforakis. Seven months' worth of mistakes: A longitudinal study of typosquatting abuse. In *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*. Internet Society, 2015.

[AMA16] Giovanni Armano, Samuel Marchal, and N. Asokan. Real-time client-side phishing prevention add-on. In *Proceedings of the IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, 2016.

[APW06] APWG. Apwg. phishing activity trends report june 2006. "http://docs.apwg.org/reports/apwg_report_june_2006.pdf", 2006.

[APW14] APWG. Apwg. global phishing survey: Trends and domain name use in 2h2014. "http://docs.apwg.org/reports/APWG_Global_Phishing_Report_1H_2014.pdf", 2014.

[ARK12] Anupama Aggarwal, Ashwin Rajadesingan, and Ponnurangam Kumaraguru. Phishari: Automatic realtime phishing detection on twitter. In *eCrime Researchers Summit (eCrime), 2012*, pages 1–12. IEEE, 2012.

[AS] Wombat Security: Security Awareness and Training Software. State of the phish 2016. http://info.wombatsecurity.com/hubfs/WombatThreatSim-StateofPhish2016_final_web.pdf.

[BFOS84] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

[bra] brandprotect. Phishing attacks: The truth about average takedown times. http://info.brandprotect.com/Blog/bid/78782/Phishing-Attacks-The-Truth-about-Average-Takedown-Times.

[Bre01] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[BWSW10] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. Lexical feature based phishing url detection using online learning. In *Proceedings of the 3rd ACM workshop on Artificial intelligence and security*, pages 54–60. ACM, 2010.

[CLTM04] Neil Chou, Robert Ledesma, Yuka Teraguchi, and John C. Mitchell. Client-side defense against web-based identity theft. In *11th Annual Network and Distributed System Security Symposium*, 2004.

[CMM83] Jaime G. Carbonell, Ryszard S. Michalski, and Tom M. Mitchell. *An Overview of Machine Learning*, pages 3–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.

[Cor] Mozilla Corporation. Webextensions. https://wiki.mozilla.org/WebExtensions.

[cre] Create undetectable facebook phishing site. http://www.picateshackz.com/2015/12/undetectable-facebook-phishing-page.html.

[Cro] Crossrider. Crossrider - build cross browser extensions with javascript. http://crossrider.com/developers.

[CSDM14] Teh-Chung Chen, Torin Stepan, Scott Dick, and James Miller. An anti-phishing system employing diffused information. *ACM Transactions on Information and System Security (TISSEC)*, 16(4):16, 2014.

[CV05] Rudi Cilibrasi and Paul MB Vitányi. Clustering by compression. *IEEE Transactions on Information theory*, 51(4):1523–1545, 2005.

[CY00] L.L. Cam and G.L. Yang. *Asymptotics in Statistics: Some Basic Concepts*. Springer Series in Statistics. Springer New York, 2000.

[DT05] Rachna Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. In *Proceedings of the 2005 Symposium on Usable Privacy and Security*, SOUPS '05, pages 77–88, New York, NY, USA, 2005. ACM.

[DTH06] Rachna Dhamija, J. D. Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 581–590, New York, NY, USA, 2006. ACM.

[ECH08] Serge Egelman, Lorrie Faith Cranor, and Jason Hong. You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1065–1074, 2008.

[EMC+10] Serge Egelman, David Molnar, Nicolas Christin, Alessandro Acquisti, Cormac Herley, and Shriram Krishnamurthi. Please continue to hold an empirical study on user tolerance of security delays, 2010.

[Fla06] David Flanagan. *JavaScript: the definitive guide.* " O'Reilly Media, Inc.", 2006.

[Fou] Outercurve Foundation. Wix toolset. http://wixtoolset.org/.

[FRGBMR+13] Óscar Fontenla-Romero, Bertha Guijarro-Berdiñas, David Martinez-Rego, Beatriz Pérez-Sánchez, and Diego Peteiro-Barral. Online machine learning. *Efficiency and Scalability Methods for Computational Intellect*, pages 27–54, 2013.

[Fri02] Jerome H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, February 2002.

[GG05] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European Conference on Information Retrieval*, pages 345–359. Springer, 2005.

[goo] Google safe browsing. https://developers.google.com/safe-browsing/.

[Hal99] Mark A Hall. *Correlation-based feature selection for machine learning.* PhD thesis, The University of Waikato, 1999.

[Inc] Alexa Internet Inc. Alexa top sites list. http://www.alexa.com/topsites.

[JM06] Markus Jakobsson and Steven Myers. *Phishing and countermeasures: understanding the increasing problem of electronic identity theft.* John Wiley & Sons, 2006.

[LMF10] Anh Le, Athina Markopoulou, and Michalis Faloutsos. Phishdef: URL names say it all. *CoRR*, abs/1009.2275, 2010.

[LMF11] Anh Le, Athina Markopoulou, and Michalis Faloutsos. Phishdef: Url names say it all. In *INFOCOM, 2011 Proceedings IEEE*, pages 191–195. IEEE, 2011.

[LXP+11] Gang Liu, Guang Xiang, Bryan A Pendleton, Jason I Hong, and Wenyin Liu. Smartening the crowds: computational techniques for improving human verification to fight phishing scams. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, page 8. ACM, 2011.

[Mar15] Samuel Marchal. *DNS and Semantic Analysis for Phishing Detection.* PhD thesis, University of Luxembourg, 2015.

[MF15] Alexey Melnikov and Ian Fette. The WebSocket Protocol. RFC 6455, October 2015.

[MFSE12] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. Proactive discovery of phishing related domain names. In *International Workshop on Recent Advances in Intrusion Detection (RAID)*, pages 190–209. Springer Berlin Heidelberg, 2012.

[MFSE14] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. Phishstorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11(4):458–471, 2014.

[MG08] D Kevin McGrath and Minaxi Gupta. Behind phishing: An examination of phisher modi operandi. *LEET*, 8:4, 2008.

[mic] Microsoft smart screen. https://support.microsoft.com/en-us/help/17443/windows-internet-explorer-smartscreen-filter-faq.

[MKK08] Eric Medvet, Engin Kirda, and Christopher Kruegel. Visual-similarity-based phishing detection. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, page 22. ACM, 2008.

[MSSA16] Samuel Marchal, Kalle Saari, Nidhi Singh, and N. Asokan. Know your phish: Novel techniques for detecting phishing sites and their targets. In *Proceedings of the IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, 2016.

[MSSV09] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254. ACM, 2009.

[Net] Netcraft toolbar. http://toolbar.netcraft.com/.

[NRSH15] Ajaya Neupane, Md. Lutfor Rahman, Nitesh Saxena, and Leanne Hirshfield. A multi-modal neuro-physiological study of phishing detection and malware warnings. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 479–491, 2015.

[PD06] Y. Pan and X. Ding. Anomaly based web phishing page detection. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 381–392, Dec 2006.

[Phi] PhishTank. Phishtank. https://www.phishtank.com/.

[PyI] PyInstaller. Pyinstaller. http://www.pyinstaller.org/.

[Rek11] Koceilah Rekouche. Early phishing. *arXiv preprint arXiv:1106.4692*, 2011.

[RT14] Akhlaqur Rahman and Sumaira Tasnim. Ensemble classifiers and their applications: A review. *CoRR*, abs/1404.4088, 2014.

[RW13] Venkatesh Ramanathan and Harry Wechsler. Phishing detection and impersonated entity discovery using conditional random field and latent dirichlet allocation. *Comput. Secur.*, 34:123–139, May 2013.

[SBJ08] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shades of grey: On the effectiveness of reputation-based "blacklists". In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 57–64. IEEE, 2008.

[Sch99] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[SHK+10] Steve Sheng, Mandy Holbrook, Ponnurangam Kumaraguru, Lorrie Faith Cranor, and Julie Downs. Who falls for phish?: A demographic analysis of phishing susceptibility and effectiveness of interventions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 373–382, 2010.

[SM86] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[SSM+12] Nidhi Singh, Harsimrat Sandhawalia, Nicolas Monet, Herve Poirier, and Jean-Marc Coursimault. Large scale url-based classification using online incremental learning. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 402–409. IEEE, 2012.

[Str12] Javelin Strategy. Research "2010 identity fraud survey report,", 2012.

[SWW+09] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Faith Cranor, Jason Hong, and Chengshan Zhang. An empirical analysis of phishing blacklists. In *Proceedings of Sixth Conference on Email and Anti-Spam (CEAS)*, 2009.

[Tea15] RSA Research Team. 2014 cybercrime roundup. "https://www.emc.com/collateral/fraud-report/h13929-rsa-fraud-report-jan-2015.pdf", 2015.

[TGM+11] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and evaluation of a real-time url spam filtering service. In *2011 IEEE Symposium on Security and Privacy*, pages 447–462. IEEE, 2011.

[VRD03] Guido Van Rossum and Fred L Drake. *Python language reference manual*. Network Theory, 2003.

[W3s] W3schools. Browser statistics and trends. http://www.w3schools.com/browsers/browsers_stats.asp.

[WBJF92] Cathleen Wharton, Janice Bradford, Robin Jeffries, and Marita Franzke. Applying cognitive walkthroughs to more complex user interfaces: experiences, issues, and recommendations. In Penny Bauersfeld, John Bennett, and Gene Lynch, editors, *Conference on Human Factors in Computing Systems, CHI 1992, Monterey, CA, USA, May 3-7, 1992, Proceedings*, pages 381–388. ACM, 1992.

[Wes08] Ryan West. The psychology of security. *Commun. ACM*, 51(4):34–40, April 2008.

[Wil04] Phil Wilson. *The Definitive Guide to Windows Installer*. Apress, 2004.

[WMG06] Min Wu, Robert C. Miller, and Simson L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 601–610, New York, NY, USA, 2006. ACM.

[WML06] Min Wu, Robert C. Miller, and Greg Little. Web wallet: Preventing phishing attacks by revealing user intentions. In *Proceedings of the Second Symposium on Usable Privacy and Security*, SOUPS '06, pages 102–113, New York, NY, USA, 2006. ACM.

[WRLP94] Cathleen Wharton, John Rieman, Clayton Lewis, and Peter Polson. The cognitive walkthrough: A practitioner's guide. In Jakob Nielsen and Robert L. Mack, editors, *Usability inspections methods*, pages 105–140. John Wiley and Sons, Inc., New York, 1994.

[WRN10] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *Proceedings of the 2010 Network and Distributed System Security (NDSS) Symposium*, 2010.

[XH09] Guang Xiang and Jason I. Hong. A hybrid phish detection approach by identity discovery and keywords retrieval. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 571–580, New York, NY, USA, 2009. ACM.

[XYA+08] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten, and Ivan Osipkov. Spamming botnets: signatures and characteristics. *ACM SIGCOMM Computer Communication Review*, 38(4):171–182, 2008.

[YW10] Chuan Yue and Haining Wang. Bogusbiter: A transparent protection against phishing attacks. *ACM Trans. Internet Technol.*, 10(2):6:1–6:31, June 2010.

[ZECH06] Yue Zhang, Serge Egelman, Lorrie Cranor, and Jason Hong. Phinding phish: Evaluating anti-phishing tools. 2006.

[ZHC07] Yue Zhang, Jason I Hong, and Lorrie F Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648. ACM, 2007.

Appendices

A Questionnaires

A.1 Background questionnaire

Background questionnaire for the usability study

Test (filled by the conductors):

- ❑ Test A
- ❑ Test B

User group (filled by the conductors):

- ❑ Group 1
- ❑ Group 2

Age:

- ❑ < 20
- ❑ 20 – 24
- ❑ 25 – 29
- ❑ 30 – 34
- ❑ 35 – 39
- ❑ > 40

Gender:

- ❑ Male
- ❑ Female
- ❑ Other / do not want to answer

Education (only mark the highest; if you have two degrees of the same level, mark both)

- ❑ Primary school
- ❑ High school
- ❑ Vocational school, major: _____
- ❑ Bachelor's degree, major: _____
- ❑ Master's degree, major: _____
- ❑ Doctoral degree: major: _____

How often do you use your laptop daily?

- ❑ < 1 h
- ❑ 1 – 3 h
- ❑ 3 – 5 h
- ❑ 5 – 7 h
- ❑ > 7 h

What is you main operating system?

- ❑ Windows
- ❑ OSX
- ❑ Linux
- ❑ Other

  If other, what? _____

What is your main purpose of laptop use?

- ❑ Studying
- ❑ Work
- ❑ Personal use
- ❑ Playing games

How often do you surf the Internet daily?

- ❑ < 1 h
- ❑ 1 – 3 h
- ❑ 3 – 5 h
- ❑ 5 – 7 h
- ❑ > 7 h

What is your main browser?

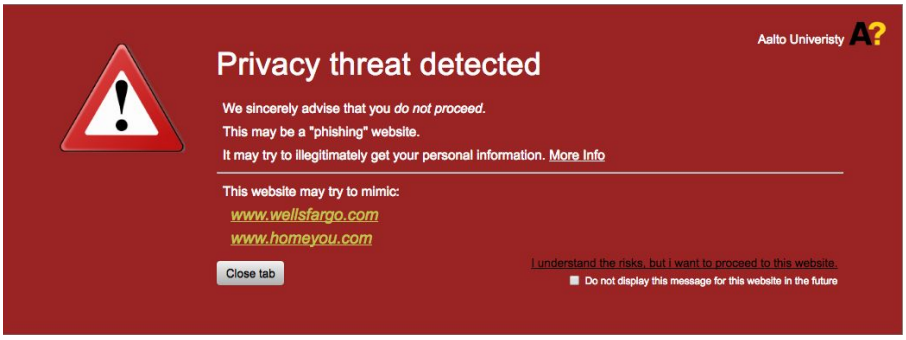
- ❑ Firefox
- ❑ Chrome
- ❑ Other

  If other, what? _____

A.2 Add-on usability questionnaire

Part 1

One of the links brought up this banner (Banner 1):



1. How did you initially interpret the banner?
   - ❑ Warning of a detected virus
   - ❑ Warning of a scam
   - ❑ Warning of a website that is illegal to visit
   - ❑ Warning of hardware malfunction
   - ❑ Warning of browser malfunction
   - ❑ Other

     If other, what? _____

3. Did you choose “More info”?
   - ❑ Yes
   - ❑ No

3. Which continuation method did you choose?
   - ❑ “This website may try to mimic...”
   - ❑ “Close tab”
   - ❑ “I understand the risks, but I want to proceed to this website”
   - ❑ Other

     If other, what? _____

4. Did you choose “Do not display this message for this website in the future”?
   - ❑ Yes
   - ❑ No

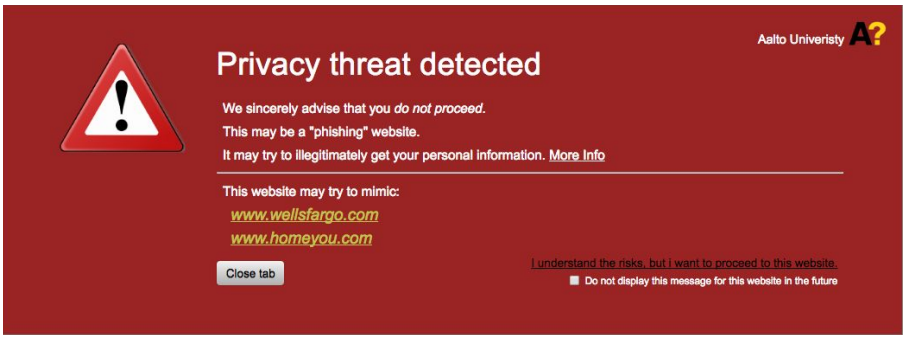
5. Mark the evaluations that most closely correspond to your reactions to the following statements.

	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly agree**
The text was easy to understand.					
The text was quick to read.					
It was immediately clear what the banner notified me of.					
The colour of the banner is appropriate.					
The banner looks professional.					
I would find the program displaying the banner useful on my computer.					
If the program was pre-installed on my computer, I would uninstall it.					

6. How useful did you find the following possible choices?

	Very useful	**Somewhat useful**	**Neutral**	**Somewhat useless**	**Completely useless**
More Info					
This website may try to mimic...					
Close tab					
I understand the risks...					
Do not display this message...					

7. If there were any parts of the banner which you did not understand, please circle them to the picture below:

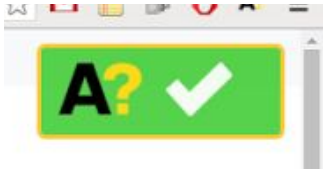


8. Compare the banner above (Banner 1) to the banner below (Banner 2), and mark which you prefer on each property.

	Banner 1	**Banner 2**
Clarity of text		
Colour		
Image		
Overall appearance		
Ease of understanding		
The understandability of the continuation links		
The usefulness of the continuation links		
Overall preference		

Part 2

One of the links brought up this banner:



Mark the evaluations that most closely correspond to your reactions to the following statements.

	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly agree**
The banner was easy to notice.					
The presence of the banner was annoying.					
The banner was easy to understand.					
The placement of the banner was appropriate.					
The banner was visible for too long.					
The banner was visible too briefly.					
The banner was unnecessary.					
The banner made me feel safe about the website.					
The banner made me doubt the safety of the website.					
I would prefer more information on the banner.					

A.3 Two weeks study general questionnaire

Mark the answer that most closely corresponds to your reaction to the following statements. All statements concern only the laptop the program was installed on.

1.	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
My browsing has slowed down.					
I have been annoyed by the presence of the program.					
My computer has slowed down in general.					

2.	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
The program has interfered with my browsing.					

If yes, how?

3.	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
The program has interfered with my use of the laptop in general.					

If yes, how?

4.	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
I have not noticed interaction between the program and my browser.					
Use of my laptop has been as enjoyable as before.					

5.	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
The program has prevented me from doing something I wanted to do.					

If yes, what?

6.	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
The browser has crashed more often than usual.					
Other programs have crashed more often than usual.					

If something has crashed, do you remember what were you doing?

A.4 Two weeks study final questionnaire

Part 1

1. During the last two weeks, have you seen this banner?
   - ❑ Yes
   - ❑ No

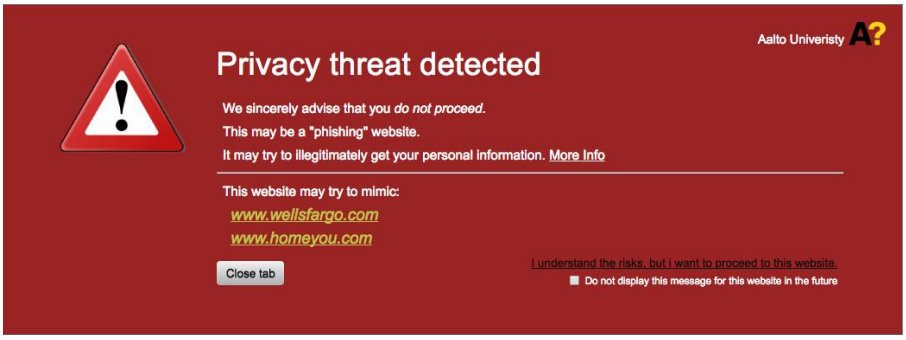


Figure 1

If you answered “No”, skip to **Part 2**.
If you answered “Yes”, continue to the questions below.

2. If there were any parts of the banner which you did not understand, please circle them to Figure 1.

3. How did you initially interpret the banner?
   - ❑ Warning of a detected virus
   - ❑ Warning of a scam
   - ❑ Warning of a website that is illegal to visit
   - ❑ Warning of hardware malfunction
   - ❑ Warning of browser malfunction
   - ❑ Other, what? _____

3. Which options displayed on the banner did you choose?
   - ❑ “More info “
   - ❑ “This website may try to mimic...”
   - ❑ “Close tab”
   - ❑ “I understand the risks, but I want to proceed to this website”
   - ❑ “Do not display this message for this website in the future”
   - ❑ None

If none, what did you do instead?

6. Mark the evaluations that most closely correspond to your reactions to the following statements about Figure 1.

	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly agree**
The text was easy to understand.					
The text was quick to read.					
It was immediately clear what the banner notified me of.					
The colour of the banner is appropriate.					
The banner looks professional.					
I would find the program displaying the banner useful on my computer.					
If the program was pre-installed on my computer, I would uninstall it.					

7. How useful did you find the following possible choices displayed on the banner?

	Very useful	**Somewhat useful**	**Neutral**	**Somewhat useless**	**Completely useless**
More Info					
This website may try to mimic...					
Close tab					
I understand the risks...					
Do not display this message...					

Part 2

1. During the last two weeks, have you seen this banner?
   - ❏ Yes
   - ❏ No

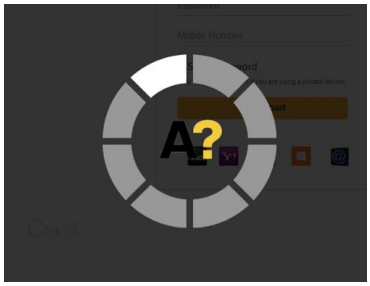


Figure 2

2. During the last two weeks, have you seen this banner?
   - ❏ Yes
   - ❏ No

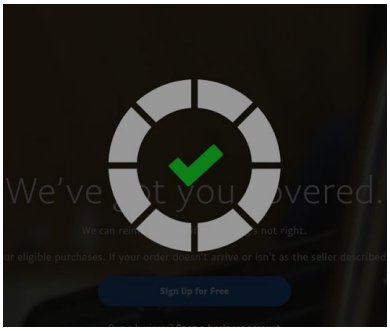


Figure 3

3. During the last two weeks, have you seen this banner?
   - ❑ Yes
   - ❑ No

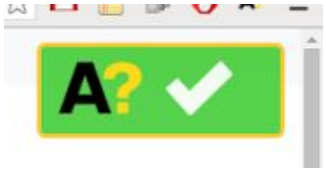


Figure 4

If no, skip to **Part 3**.
If yes, mark the evaluations that most closely correspond to your reactions to the following statements about Figure 4.

	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly agree**
The banner was easy to notice.					
The presence of the banner was annoying.					
The banner was easy to understand.					
The placement of the banner was appropriate.					
The banner was visible for too long.					
The banner was visible too briefly.					
The banner was unnecessary.					
The banner made me feel safe about the website.					
The banner made me doubt the safety of the website.					
I would prefer more information on the banner.					

Part 3

1. Prior to the two-week study, were you familiar with the concept of “phishing”?
   - ❑ Yes
   - ❑ No

If yes, explain briefly how you understand what “phishing” means?

2. During the two-week study:

	Yes	**No**
Have you received e-mail you have thought to be phishing?		
Have you visited websites you have thought to be phishing?		
Have you received a warning about phishing?		

If you have received a warning about phishing during the last two weeks, was it Figure 1 (see the first page) or another warning?

- ❑ Figure 1
- ❑ Another warning
- ❑ I have received both Figure 1 and another warning

Thank you for participating!