Off-the-Hook: An Efficient and Usable Client-Side Phishing Prevention Application

Samuel Marchal, *Member, IEEE*, Giovanni Armano, Tommi Gröndahl, Kalle Saari, Nidhi Singh and N. Asokan, *Fellow, IEEE*

Abstract—Phishing is a major problem on the Web. Despite the significant attention it has received over the years, there has been no definitive solution. While the state-of-the-art solutions have reasonably good performance, they suffer from several drawbacks including potential to compromise user privacy, difficulty of detecting phishing websites whose content change dynamically, and reliance on features that are too dependent on the training data.

To address these limitations we present a new approach for detecting phishing webpages in real-time as they are visited by a browser. It relies on modeling inherent phisher limitations stemming from the constraints they face while building a webpage. Consequently, the implementation of our approach, *Off-the-Hook*, exhibits several notable properties including high accuracy, brand-independence and good language-independence, speed of decision, resilience to dynamic phish and resilience to evolution in phishing techniques.

Off-the-Hook is implemented as a fully-client-side browser add-on, which preserves user privacy. In addition, *Off-the-Hook* identifies the target website that a phishing webpage is attempting to mimic and includes this target in its warning. We evaluated *Off-the-Hook* in two different user studies. Our results show that users prefer *Off-the-Hook* warnings to Firefox warnings.

Index Terms—Phishing webpage detection, phishing prevention, phishing target identification, machine learning, web security, browser add-on.

1 Introduction

Phishing webpages ("phishs") lure unsuspecting web surfers into revealing sensitive information. It is a major security concern on the web, Many solutions have been proposed to detect and avoid phishs. Nevertheless, phishing detection remains an arms race with no definitive solution. Automated phish detection systems [1], [2] with acceptable accuracy (>99%) and achieving very low rates of misclassifying legitimate webpages (<0.1%) are computationally expensive and slow. Thus, they are typically used in a centralized architecture (e.g. Google Safe Browsing [3], PhishTank [4]) where a blacklist of phishing sites is constructed based on offline analysis of websites. This raises major issues including *several days of delay* in phish identification [5] and *vulnerability to dynamic phishing* where a phishing website serves different content depending on who the client is. In addition, users must share their browsing history with these centralized services thereby *compromising their privacy*. These concerns are partially addressed by real-time client-side solutions, but existing client-side solutions typically have *low detection accuracy* [6].

Most techniques [1], [2], [3] primarily use a bag-of-words approach and are thus *language and brand-dependent*. While they can be effective at detecting phishs against known target "brand" (like "paypal"), they are not effective against phishs masquerading as brands that were not known targets. Use of static words as features in a phish detection model makes it more vulnerable to circumvention by including specific words that can increase the chance of a phish being misclassified as legitimate [7]. Finally, phishing warnings in today's web browsers (e.g. Chromium, Firefox) have two drawbacks. First, users are only told that the website they are trying to access is a phish. We argue that a more useful guidance would be to point the user towards the legitimate website that they intended to visit in the first place. Second, warning messages typically use technical jargon which make them difficult to understand [8].

In this paper, we introduce a new phish detection tool, *Off-the-Hook*. It is implemented as a browser add-on that can decide in real time if a visited webpage is a phish. On encountering a phish, *Off-the-Hook* identifies the *target brand* mimicked by the phish. *Off-the-Hook* implementation is fully-client-side and the decision process relies solely on information extracted from the web browser while loading a webpage. Thus it *preserves users' privacy*, provides *real-time protection* and is *resilient to dynamic phish* since the content actually loaded in the browser is analyzed to render a decision.

The core of *Off-the-Hook* lies in modeling **inherent phisher limitations** evident in the composition of phishing webpages. For instance, external hyperlinks and external content sources on a phish point to domains that are typically *outside the control* of the phisher. Moreover, while phishers can freely change most of the phishing page, the latter part of its domain name is *constrained* as it is limited to those domains that are generally controlled by phishers. By measuring differences in the composition and consistency of term usage in *constrained/unconstrained* and *controlled/uncontrolled* sources, we improve the effectiveness of phish detection. Modeling phishers' limitations that typically remain constant over time makes our detection model hard to circumvent and *resilient to evolution in phishing techniques*. The

- *S. Marchal, T. Gröndahl, K. Saari, N. Asokan are with the Secure Systems Group of Aalto University, Finland. E-mail: {samuel.marchal,tommi.grondahl}@aalto.fi, asokan@acm.org, kalle.saari.9675@gmail.com*
- *G. Armano was with the Secure Systems Group of Aalto University, Finland. He is now with Portaltech Reply, United Kingdom. E-mail: giovanniarmano70713@gmail.com*
- *N. Singh is with McAfee Gmbh, Germany. E-mail: nidhi.singh@McAfee.com*

use of a small number of non-static features (210) allows *fast decision* (<0.5 second), high accuracy (99.9%) and low rate of mislabeling legitimate websites (<0.1%), with very little labeled training data. By eschewing the bag-of-words approach *Off-the-Hook* is not limited to specific languages or targeted brands.

In case of a phish, *Off-the-Hook* uses simple language to formulate the warning to users and points them to the likely target of the phish. We evaluated *Off-the-Hook* in two user studies showing that it is likely to be acceptable to user.

We claim the following contributions :

- the design and implementation of a **client-side-only** phish detection tool: *Off-the-Hook* (Sect. 3). It offers (a) **better privacy**, (b) **real-time protection**, (c) **resilience to dynamic phish** and (d) effective warnings.
- a new set of features to detect phishs (Sect. 4.3) and a classifier, using these features. Unlike previous work, our approach is **brand-independent**, provides **good language-independence** and learns a generalized model for detecting phishs making it **resilient to evolution in phishing techniques**.
- a fast target identification technique (Sect. 5) with accuracy (90-97%) comparable to previously reported techniques. Identified likely **targets are included as redirection** options in phishing warnings.
- a comprehensive evaluation of the application, showing that its accuracy (>99.9%) and misclassification rate (<0.1%) are comparable to state-of-the-art while being **client-side-only** (Sect. 6) .
- two usability studies of *Off-the-Hook* (Sect. 7) showing that it is usable and its **warnings are preferred to Firefox warnings**.

2 Background

2.1 Phishing

Phishing refers to the class of attacks where a victim is lured to a fake webpage masquerading as a target website and is deceived into disclosing personal data or credentials. Phishing campaigns are typically conducted using spam emails to drive users to fake websites [9]. Impersonation techniques range from technical subterfuges (email spoofing, DNS spoofing, etc.) to social engineering. The former is used by technically skilled phishers while unskilled phishers resort to the latter. Phishs mimic the look and feel of their target websites [10]. In order to make the phishs believable, phishers may embed some content (HTML code, images, etc.) taken directly from the target website and use relatively little content that they themselves host [11]. This includes outgoing links pointing to the target website. They also use keywords referring to the target in different elements of the phishs (title, text, images, links) [9], [11], [12], [13].

2.2 URL Structure

A webpage is typically addressed by a uniform resource locator (URL), which typically has the structure as shown in Fig. 1. It begins with the *protocol* used to access the page. The fully qualified domain name (*FQDN)* identifies the server hosting the webpage. The *FQDN* in turn, consists of a registered domain name (*RDN*) and prefix which we refer to as *subdomains*. A phisher has full control over the *subdomains* portion and can set it to any value. The *RDN* portion is constrained since it has to be registered with a domain name registrar. *RDN* is composed of a *public suffix* (*ps*) preceded by a *main level domain* (*mld*). The URL may also have a *path* and *query* components which, too, can be changed by the phisher at will. We use the term *FreeURL* to refer to those parts of the URL that are fully controllable by the phisher.

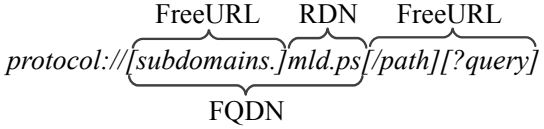


Fig. 1: Structure of a URL

Consider an example URL:
https://www.amazon.co.uk/ap/signin?_encoding=UTF8
We can identify the following components:

- *protocol = https*
- *FQDN = www.amazon.co.uk*
- *RDN = amazon.co.uk*
- *mld = amazon*
- *FreeURL* = {*www*, */ap/signin?_encoding=UTF8*}

2.3 Data Sources

Our analysis of phishs (that are loaded in a web browser) yields the following data sources that can be useful in detecting phishs:

- *Starting URL*: the URL provided to the user to access the website. It can be distributed in emails, instant messages, websites, documents, etc.
- *Landing URL*: the final URL pointing to the actual content presented to the user in his web browser (this is the URL shown in the browser address bar when the page is completely loaded).
- *Redirection chain*: the set of URLs requested to go from the starting URL to the landing URL (including both).
- *Logged links*: the set of URLs logged by the browser while loading the page. They generally point to sources from which embedded content (code, images, etc.) in the webpage are loaded.
- *HTML*: the HTML source code of the webpage and iFrames included in the page. We consider four elements extracted from this source code:
  - *Text*: text contained between <*body*> HTML tags (actually rendered on user's display).
  - *Title*: text contained between <*title*> HTML tags (appears in the browser tab title).
  - *HREF links*: the set of URLs representing outgoing links in the webpage.
  - *Copyright*: the copyright notice, if any, in Text.

3 *Off-the-Hook* Design

3.1 Design Goals

We designed *Off-the-Hook* to satisfy a number of requirements:

- *Accuracy (R1)*: misidentification of legitimate webpages as phishs (false positives) must be minimal to avoid harming usability. Detection rate of phishs (recall) must be maximal to provide good protection.
- *Context-independent detection (R2)*: the detection model must not rely on features specific to a particular target or language.

- *Temporal resilience (R3)*: detection accuracy must not degrade over time as phishers adapt their techniques.
- *Resilience to dynamic phishs (R4)*: the phish or not-phish decision must be based on the actual webpage content depicted in the browser.
- *User privacy (R5)*: phish detection must not require users having to disclose their browsing history to any outside party.
- *Effective protection (R6)*: to protect users before they disclose passwords or other sensitive information to a phisher, detection must take place quickly (<1 second). Warning messages must be clear and easily understandable. They must provide both relevant information about the threat and relevant continuation options.

3.2 Design Choices

To meet the aforementioned requirements we adopted the following design choices:

- *Client-side implementation*: *Off-the-Hook* is fully implemented on the client-side. It computes its decision based on the webapge content actually displayed in the browser. It is thus privacy preserving (*R5*) and is not vulnerable to dynamic phishs (*R4*).
- *Model phisher limitations*: the detection model relies on inherent phisher limitations (Sect. 4.1). Thus, we expect these limitations to be present in any phish, which will make the detection accurate (*R1*). Those limitations will also remain over time and are difficult to circumvent, thus allowing the detection technique to be temporally resilient (*R3*).
- *Use few but non-static features*: the features used for the model are *non-static*: they represent phisher limitations rather than static words found in training data. The detection is thus context-independent (*R2*). Using a small number of features allows the decision to be fast (*R6*).

3.3 Decision Flow

Off-the-Hook's overall decision process, depicted in Fig. 2 involves two main components: a *phish detector* and a *target identifier*. The phish detector is a classifier that identifies phishs based on a set of new features (Sect. 4). The target identifier finds the likely target of a phish based on "keyterms" in a webpage (Sect. 5). The system is complemented by a local *whitelist* that can preempt any webpage from being analyzed. It is composed from previous corrective decisions.

When the browser visits a URL, the data sources of the corresponding webpage are extracted. If the *landing URL* belongs to the whitelist, the webpage is considered legitimate and no further analysis is performed. Otherwise, the extracted data sources are fed to the phish detector that classifies the page as "phish" or "not-phish". If the decision is "phish", the target identifier infers the list of likely targets. If one of the target matches the *landing URL*, the tentative decision of the phish detector is overruled by the target identifier and the page is deemed legitimate. If not, the page is confirmed as phish and its target is identified. The results are communicated to the user via color-coded icons and messages as shown in Fig. 2.

3.4 Architecture and Implementation

Off-the-Hook is implemented as a web browser add-on for Google Chrome and Mozilla Firefox, and distributed for Windows ($\geq$ 8),

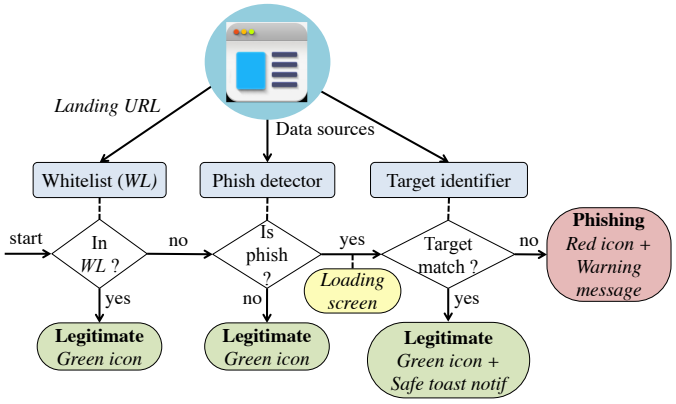


Fig. 2: *Off-the-Hook* decision process and warning messages

Ubuntu ($\geq$ 12.04) and OS X ($\geq$ 10.8). A beta version is publicly available for download [14]. Its implementation architecture is depicted in Fig. 3.

Add-on is developed in Javascript and interacts with the web browser. It collects the required data sources (Section 2.3) and displays the phishing decision indicator and any needed message in the browser window. The add-on is composed of the *content script* and the *background script*.

Background processes are developed in Python and executed in separated processes to handle the analysis of data extracted from the webpages. They consist of the *phish detector* (Sect. 4) and the *target identifier* (Sect. 5). The *dispatcher* handles the data exchanged between these processes and the add-on.

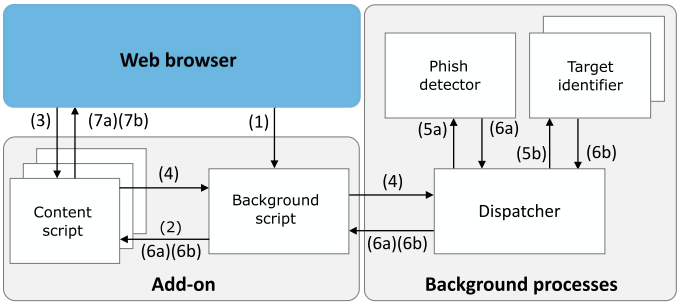


Fig. 3: Software implementation architecture

Background script runs concurrently with the browser to collect the *redirection chain* and the *logged links* every time a new webpage is loaded (1). It sends the collected data to the corresponding instance of the *content script* (2). When all data sources are extracted, the *background script* forwards them to the *dispatcher* (4). It later sends the computed result of phish detection and target identification back to the *content script* (6a)(6b).

Content script is executed at every page load to combine the data collected by the *background script* with the information extracted from the webpage (3). It gathers *title, text, HREF links* and *HTML source* and combine them with *redirection chain* and *logged links* to generate a json file containing all the data sources. It sends this file to the background processes for analysis (4). When it receives the result from the *phish detector* (6a), in case of a phish, it blocks users' interaction with the webpage (7a) until results from the *target identifier* are returned (6b). Then, it either displays a warning or a confirmation of the webpage legitimacy depending on

target identification results (7b). There is one instance of *content script* running per opened browser tab.

Dispatcher waits for incoming json files from the *content scripts* (4). It checks the *landing URL* against the whitelist to determinate if the page has to be analyzed. If the URL is not found in the whitelist, the data is forwarded to the *phish detector* (5a) and to one instance of the *target identifier* (5b) for processing. Results of phish detection are forwarded to the *content script* (6a). Received target domains (6b) are checked against the *landing URL*. If the *RDN* of any target corresponds to the *RDN* of the *landing URL*, the webpage is considered as legitimate. The URL is added to the local whitelist and the decision is transmitted to the *content script*. Alternatively, the list of targets is sent (6b). The local whitelist is supplied with *landing URLs* from which the target identifier confirms the legitimacy of the webpage while the phish detector predicted it as phish. A second source of whitelisted URLs is when the user overrides the result of phish detection.

Phish detector extracts a feature vector (Sect. 4.3) from the data sources received from the *content script* and performs the classification of the website (5a). It sends the result of the classification to the *content script* through the *dispatcher*.

Target identifier infers the potential targets of the phish (5b) based on the information contained in multiple data sources (Sect. 5.2). Once the targets are identified, it sends them to the *dispatcher* (6b). Since this task is relatively more time consuming (cf. Sect. 6.5), two instances of target identifiers run concurrently.

We designed the software architecture of *Off-the-Hook* to be independent from any system and hardware. We developed it in high level programming languages, i.e. Python and Javascript, to be portable with minimal changes to the largest class of devices and operating systems. Nevertheless, browsers expose different functions for extracting data sources and our *add-on* thus has browser-specific Javascript artifacts. One way to eliminate such browser-dependencies is to use cross-browser extension frameworks such as KangoExtensions [15] or BestToolBars [16]. They enable the use of a single code base for all browsers addressed by the framework e.g. Firefox, Chrome, Safari, etc. However, popular browser platforms like Google Chrome have recently started to block non-self-contained add-ons (which make use of external libraries) from being published in their market places.

We developed background processes in Python because they rely on the availability of machine learning libraries that are not yet available in Javascript. Since these background processes are external programs running concurrently with the browser, they require operating system specific packaging.

An optimal implementation of Off-the-Hook would be native integration in a browser, similarly to existing phishing prevention systems like Google Safe Browsing [3]. Such a solution would permit a native implementation of the whole system in a single language, e.g. C, increasing performance and reducing the communication overhead between different programs.

3.5 User Interface

The user interface is designed to meet a trade-off between salience and ease of understanding. Phishing warnings are implemented as active warnings that interrupt users' task and move their attention to the message, which guarantees their efficiency [8], [17] We used a cognitive walkthrough [18] during the development of *Off-the-Hook* to refine the layout and content of various notifications and warnings. The resulting interface has minimal jargon and textual content: the phishing warning shows a minimal message along with options for continuation (Fig. 4). The user interface displays the four following elements:

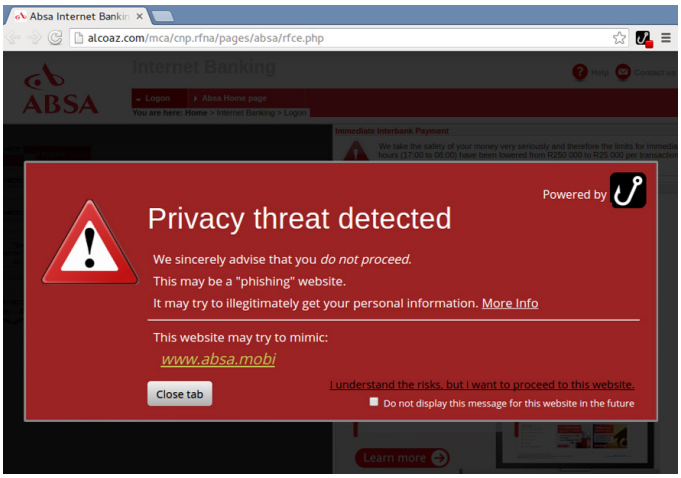


Fig. 4: *Off-the-Hook* warning message (Google Chrome extension)

Navigation bar icon is composed of *Off-the-Hook*'s logo combined with a colored badge. A green badge indicates a legitimate webpage and a red badge indicates a phish (Fig. 4: top-right).

Loading screen is a layer that prevents users' interaction with the webpage. It is composed of a semi-transparent layer that covers the webpage entirely. An animated *Off-the-Hook* icon occupies the center of the loading screen. This loading screen appears when the phish detector predicts a page as a phish, and remains until the target identifier provides its result.

Warning message pops up in the center of the window and is surrounded by a semi-transparent layer (Fig. 4). It contains information understandable by ordinary, non-expert users. The warning message provides continuation options that allow users to proceed to the website, with an option to remember the decision having the effect of adding the *landing URL* to the local whitelist. They can close the browser's tab. In contrast with existing warnings [8], we leverage our target identifier to propose links redirecting to the possible target of the phish.

Safe toast notification consists of a green logo placed on the top-right of the browser window. It is displayed for five seconds after the *loading screen* to indicate that a webpage is legitimate, when the target identifier overrides the phish detector.

The *navigation bar icon* is is always present. The other elements are displayed depending on phish detection and target identification results for a certain webpage as observed in Fig. 2.

4 Phish Detection

4.1 Modeling Phisher Limitations

We recall that even on systems they control, phishers are constrained from freely constructing URLs to pages they host (Sect. 2.2). Similarly, in order to maximize the believability of their phishs, phishers include content from URLs outside their control (Sect. 2.1). Thus, we divide the data sources from Sect. 2.3 into subcategories according to the level of *control* phishers may have on them and the *constraints* on phishers.

Control: URLs from *logged links* and *HREF links* are subdivided into *internal* and *external* according to their *RDN*. The set of *RDN*s

extracted from URLs involved in the redirection chain are assumed to be under the control of the webpage owner. Any URLs that include these *RDN*s are marked *internal*. Other *RDN*s are assumed to be possibly outside the control of the webpage owner. URLs containing such *RDN*s are marked *external*.

Constraints: Within a URL, we distinguish between *RDN*, which cannot be freely defined by the webpage owner, and *FreeURL*, which can be. *RDN* is thus *constrained* by DNS registration policies and *FreeURL* is *unconstrained*.

4.2 Extracting Term Distributions

While facing the aforementionned limitations, the primary technique of a phisher is essentially social engineering: fooling a victim into believing that the phish is the target [19]. Thus, it is plausible that lexical analysis of the data sources will help in identifying phishs: we conjecture that legitimate webpages and phishs differ in the way terms are used in *different locations* in those pages. To incorporate measurements of such term usage consistency, we first define what "terms" are and how they are extracted from a webpage. Let A be the set of the 26 lowercase English letters: $A = \{a, b, c, ..., x, y, z\}$. We extract terms from a data source as follows:

- canonicalize letter characters by mapping upper case characters, accented characters and special characters to a matching letter in A; e.g., $\{$ *B, β, b̀, b̂* $\} \rightarrow b$.
- split the input into substrings whenever a character $c \notin A$ is encountered.
- throw away any substring whose length is less than 3.

Let $T = A^n | n \geq 3$ be the set of all possible terms. Suppose $T_S = \{t_{i \in \{1;m\}} \in T\}$ was extracted from a data source S and t_i occurs with probability p_i. The set of m pairs $(t_i, p_i) \in T \times]0, 1]\,, i \in \{1; m\}$ represents the *term distribution* D_S of *S*.

TABLE 1: Term distributions with indicator showing if they are controlled or uncontrolled / constrained or unconstrained

Distribution	Data source	Cont./Constr.
D_{text}	Text	yes / no
D_{title}	Title	yes / no
$D_{copyright}$	Copyright notice	yes / no
D_{start}	Starting URL – *FreeURL*	yes / no
D_{land}	Landing URL – *FreeURL*	yes / no
D_{intlog}	Internal logged links – *FreeURL*	yes / no
$D_{intlink}$	Internal HREF links – *FreeURL*	yes / no
$D_{startrdn}$	Starting URL – *RDN*	yes / yes
$D_{landrdn}$	Landing URL – *RDN*	yes / yes
D_{intrdn}	Internal links (HREF & logged) – *RDN*	yes / yes
D_{extrdn}	External logged links – *RDN*	no / yes
D_{extlog}	External logged links – *FreeURL*	no / no
$D_{extlink}$	External HREF links – *FreeURL*	no / no

Tab. 1 defines the term distributions we consider. The external sources $extrdn$, $extlog$, $extlink$ are those assumed to be outside the control of the webpage owner. *RDN* data sources $startrdn$, $landrdn$, $intrdn$, $extrdn$ are constrained by DNS registration. The rest is controlled by the webpage owner without constraints.

4.3 Computing Features

We now introduce a set of 210 features and motivate their selection. We intend to capture the constraints and degree of control discussed earlier (Sect. 4.1) as well as consistency checking of term usage (Sect. 4.2). We group features into the following five categories (Tab. 2).

TABLE 2: Feature sets used for phish detection

Name	Count	Type
f_1	92	URL
f_2	68	Term usage consistency
f_3	32	Usage of starting and landing *mld*
f_4	13	*RDN* usage
f_5	5	Webpage content
f_{all}	210	Entire feature set

URL: First we define eight statistical features related to the lexical composition of URLs (Tab. 3). Feature 2 is meant to identify strings in *path* and *query* that look like domain names. Phishing URL and domain name obfuscation techniques [12] tend to produce long URLs composed of many terms. This is the rationale for features 3-8.

All eight features are extracted from the starting URL (8) and landing URL (8). The mean, median and standard deviation values are computed for features 3-8 on the following sets of URLs: internal logged links, external logged links, internal HREF links and external HREF links ($4*6*3$). Feature 1 is computed on these sets as a ratio of URLs using *https* over the total count of URLs for each set ($4*1$). Feature 2 is computed only for the starting and landing URLs. Thus, the complete URL-based feature set (f_1) consists of 92 features: $8+8+4*(6*3+1) = 92$.

TABLE 3: URL features (f_1)

#	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*

Term usage consistency: The second set of features (f_2) captures the consistency of term usage between different types (controlled vs. uncontrolled; constrained vs. unconstrained) of data sources in the page. Using 12 term distributions (we discard $D_{copyright}$) defined in Sect. 4.2, we create 68 features ($12 * 11/2 + 2$) as follows. 66 features depict the similarity of pairs of sources by computing pairwise Hellinger Distance between their distributions ($12 * 11/2$). The Hellinger Distance [20] is a metric used to quantify the dissimilarity between two probabilistic distributions P and Q. It is an instance of f-divergence that is symmetric and bounded in $[0, 1]$. The value 1 represents complete dissimilarity ($P \cap Q = \emptyset$) and the value 0 means that P and Q are the same probabilistic distribution. We stress the importance of two distributions by defining two additional binary features set to 1 if any word from D_{intrdn}, or D_{extrdn} respectively, is present in D_{title}.

Usage of starting and landing *mld*: Legitimate websites are likely to register a domain name reflecting the brand or the service they represent. However, phishers often use domain names having no relation with their target [10]. Hence, we expect the starting *mld* and/or the landing *mld* to appear in several sources extracted from a legitimate webpage while phishs should not have this characteristic. We define 32 features (f_3) inferring the usage of the starting and landing *mld* in the text, the title and *FreeURL* of the logged links and HREF links. 12 binary features are set to 1 if the starting/landing *mld* appear in D_{text}, D_{title}, D_{intlog}, D_{extlog}, $D_{intlink}$ or $D_{extlink}$ (6*2). 20 features are the sum of probability from terms of D_{title}, D_{intlog}, D_{extlog}, $D_{intlink}$ and $D_{extlink}$ that are substrings of starting/landing *mld* (5*2) or that are substrings of starting/landing *FreeURL*, but not substrings of starting/landing *mld* (5*2). The latter 10 features

outline obfuscation techniques that use keywords of the target in *FreeURL*. D_{text} is not considered since it is often composed of many short irrelevant terms that match several parts of a *mld*.

***RDN* usage**: We define 13 features (f_4) related to *RDN* usage consistency. We compute statistics related to the use of similar and different *RDN*s in starting URL, landing URL, redirection chain, loaded content (logged links) and HREF links. We expect legitimate webpages to use more *internal RDN*s and less redirection than phishs [21].

Webpage content: Finally, five features (f_5) count the number of terms in the text and the title (2), and the number of input fields, images and IFrames (3) in the page. Phishs tend to have minimal text to circumvent text-based detection techniques [22] and use more images and HTML content loaded from other sources. In addition, since phishing attacks seek to steal user data, phishs often contain several input fields [10].

Recalling the requirements formulated in Sect 3.1, it is worth noting that while we use terms to compute our feature set, it is not based on any observed language or term usage knowledge. The computation relies solely on the information gathered through a web browser (*R4, R5*). Hence, the feature set is context-independent (*R2*) and the use of non-static features mitigates the risk for adversarial attacks on the machine learning model (*R3*). Furthermore, since the feature set is small (210 features), we expect it to be fast to compute once the data sources are available (*R6*).

4.4 Phish Detection Model

To use our feature set for discriminating phishs from legitimate ones, we use a supervised machine learning approach. In supervised machine learning, a classification model is learned from observations over a set of data labeled with several classes. The learned model is used to predict the class of unlabeled instances. We use Gradient Boosting [23] to build the classification model because (a) of its strong ability to select and weight the most relevant features and (b) boosting algorithms are known to be fairly robust to overfitting, enabling the resulting model to have good generalization capabilities [24].

Gradient Boosting predicts the class of an unknown instance by computing values defined in $[0, 1]$ that gives the confidence of the instance to belong to a given class. In the case of predicting only two classes, the confidence value v_1 for one class is equal to $1 - v_2$, where v_2 is the confidence value for the other class. A *discrimination threshold* predicts, according to the computed confidence values, the class of an instance. By tuning this threshold, we can favor the prediction of one class over the other. The variation of the discrimination threshold over $[0, 1]$ is used to evaluate the accuracy of a given model by examining how false positive rate varies with true positive rate (ROC) or precision varies with recall.

5 Target Identification

The identification of the target of a phish relies on a set of "keyterms" in that webpage related to a brand or service. Rather than leveraging any brand-specific knowledge or text corpus to infer these keyterms [25], [26], we introduce a new technique that uses only the information extracted from the webpage.

5.1 Keyterms Extraction

A keyterm is one that appears in several data sources (e.g., title, text and landing URL) on a page. We use terms from five data sources introduced in Sect. 2.3 that contain user-visible data rendered by the browser:

- Starting and landing URLs: $T_{start} \cup T_{startrdn} \cup T_{land} \cup T_{landrdn}$
- Title: T_{title}
- Text: T_{text}
- Copyright: $T_{copyright}$
- HREF links: $T_{intlink} \cup T_{extlink}$

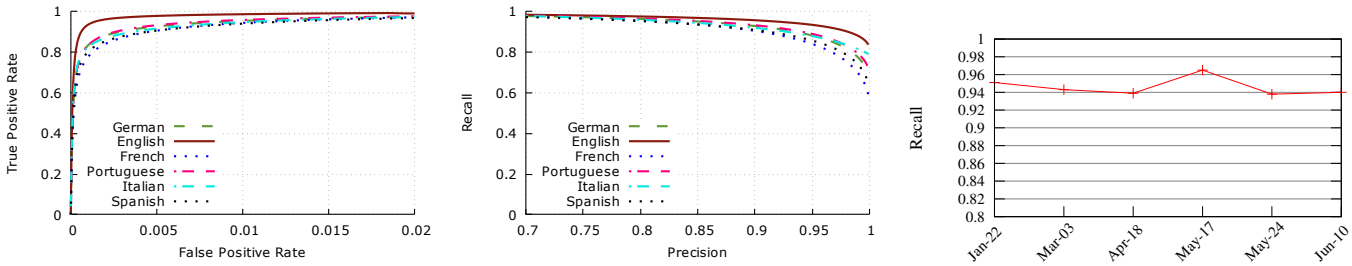
We use two different techniques to identify keyterms. They are used in sequence, depending on the information available in different data sources and the success of each technique (Sect. 5.2). The first technique considers the result of pairwise intersection between the five sets of terms as potential keyterms. Each term appearing in at least two data sources is added to a list and ranked in descending order according to a term's overall frequency in the visible parts of the website. The top-*N* terms in the ordered list are selected as keyterms. (We use *N=5* in our model as it was proved to be a sufficient number to represent a webpage [27].) These *N* keyterms are called *boosted prominent terms*. The second technique considers the same data sources but discards the intersection between text and HREF links ($T_{text} \cap (T_{intlink} \cup T_{extlink})$). In certain scenarios, the text and links of a webpage contain the same terms because the name of the links and the corresponding URL can be the same. This is common practice in news websites. In such cases, the intersection terms may be dominated by terms that are irrelevant for target identification, which may introduce some noise in the keyterms inference. The *N* extracted keyterms using this technique are called *prominent terms*.

5.2 Identification Process

We now discuss how the extracted keyterms lists (*boosted prominent terms* and *prominent terms*) are used to infer the target of a phish.

Step 1: Extract *boosted prominent terms*, and try to "guess" the target *FQDN*. The *mld*s from the starting and landing URLs, from the logged links and HREF links are collected. Thereafter, every collected *mld* is checked to figure out if it can be composed based on the keyterms part of *boosted prominent terms* possibly separated by a dash '-' or a string of digits. For each guessed *FQDN* (typically 2-3), a search engine query is performed and the returned *RDN*s are stored. If the *RDN*s of the suspected phish (starting and landing URL) appear in the results of the search engine query, we declare this site legitimate and stop the process. Otherwise we go to step 2. This decision is based on the assumption that a search engine would not return a phishing site as a top hit because (a) a new phishing site (only a few hours old) would not have been indexed by a search engine yet and (b) an old phishing site would have been already detected and ended up in a blacklist.

Step 2: The set of *N prominent terms* is queried against a search engine. If the suspected *RDN* appears in the set of *RDN*s returned by the search engine, it is declared legitimate and we stop the process. If some *mld*s resulting form the search engine query appear in a controlled data source of the webpage, we record them and go to step 4. These *mld*s represent the candidate targets. Alternatively, we continue to step 3.



(a) ROC evaluation results for six languages (b) Precision vs recall evaluation for six languages (c) Recall evolution over time

Fig. 5: Accuracy metrics evolution for the phish detection system

Step 3: Repeat Step 2 but instead of using *prominent terms*, use *boosted prominent terms*. If the webpage is not confirmed as legitimate, go to step 4.

Step 4 (target selection): For each *mld* candidate target, we count how many times it appears in the data sources of the webpage and rank it in a list according to this criteria. *Off-the-Hook* depicts the three most frequent (*i.e.*, top-3) potential targets in its warning message. If a single target is required, we can return top-1.

6 Performance Evaluation

This section presents the performance evaluation of the phish detection system and the target identification method presented in Sect. 4 and 5 respectively. The accuracy evaluation was performed offline with an instrumented version of *Off-the-Hook* on datasets previously captured.

6.1 Evaluation Datasets

We obtained URLs from two sources in order to gather ground truth data of phishs and legitimate webpages (Tab. 4). Neither dataset contains personal data and both datasets are available on request for research use.

The set of phishing URLs (**Phish**) was obtained through the community website PhishTank [4]. We conducted three different collection "campaigns". The first campaign resulted in *phishTrain* which was used for training the phish detection classifier. The second campaign, perfromed at a later point in time (4-10 months), resulted in *phishTest* which was used as the test set. *PhishTest* consists of webpages in a variety of languages. *PhishTest1K* is a random sample of *phishTest* used for testing the classification model with a realistic phishing to legitimate webpage distribution ($\approx 1/100$) [2], [27]. We manually determined the language of each webpage in *phishTest1K*. It is composed of 84% English webpages, 6% French, 4% Japanese, 2% German and 4% other languages (including Spanish and Portuguese). The last campaign resulted in *phishBrand* that was used for evaluating our target identification scheme (Sect. 6.3). *PhishBrand* consists of 600 phishs for each of which we manually identified the target, resulting in a total of 126 different targets. Each campaign consisted of checking for new entries in PhishTank every hour and scraping the webpages (in several languages) for those URLs. The datasets were further manually sanitized to remove any legitimate or unavailable websites and parked domain names. Tab. 4 provides a detailed description of these datasets including the collection period and the count of URLs before and after sanitization.

The legitimate URLs (**Leg**) were provided by Intel Security. An English training set (*legTrain*) is composed of 8,500 legitimate webpages. Six larger test sets of webpages in different languages (English, French, German, Portuguese, Italian and Spanish) were gathered. A detailed description of these sets is provided in Tab. 4 with the percentage of webpage in each set having a *RDN* listed in Alexa top 1M, depicting the diversity and popularity of the URLs.

TABLE 4: Datasets description

Set	Name	Collection Period	Initial	Clean	Alexa
Phish	*phishTrain*	15-Jul-23/Sep-9	2,766	1,500	–
	phishTest	16-Jan-22/Jun-10	21,531	13,309	–
	phishTest1K	16-Jan-22/Jun-10	–	1,000	–
	phishBrand	15-Sep-22/Sep-28	600	600	–
Leg	*legTrain*	15-Jul-22/Aug-01	8,500	–	56%
	English	15-Aug-17/Sep-23	100,000	–	61%
	French	16-Jul-29/Aug-17	20,000	–	18%
	German	16-Jul-6/Jul-29	20,000	–	5%
	Italian	16-Jul-5/Jul-29	20,000	–	11%
	Portuguese	16-Jul-15/Aug-18	20,000	–	12%
	Spanish	16- Jul-29/Aug-26	20,000	–	22%

6.2 Phish Classification

6.2.1 Accuracy and Language-Independence

We now present detailed evaluation of our phish detection method across six different languages so as to demonstrate its language-independence characteristics. The model was trained using *legTrain* (8,500) and *phishTrain* (1,500). The testing was done using *phishTest1K* (1,000) and each language specific dataset, *e.g.*, *English* (100,000), *French* (20,000), *German* (20,000), etc.

To begin this evaluation, we first compute *Receiver Operating Characteristic (ROC)* and corresponding Area Under the Curve (AUC), which shows the change of false positive rate with respect to true positive rate while varying the discrimination threshold of the classifier. The evaluation results for legitimate dataset of six languages are shown in Fig. 5a, wherein we see that, at a significantly high true positive rate of 0.9, the false positive rate for all languages is very low ($<$0.008). As the true positive rate increases to around 0.95, the false positive rate does not increase much. Even at true positive rate of 0.98, the false positive rate stays substantially low at 0.02. In line with these results, the AUC is around 0.999 for all languages, as shown in Tab. 5. Note that these results are consistent across all languages, which is very desirable in a multi-lingual phish detection scenario.

The detailed evaluation results for precision, recall and false positive rate are shown in Tab. 5. These values were obtained by setting the discrimination threshold of Gradient Boosting to 0.7, which favors the prediction of legitimate webpages ($[0, 0.7[$) over phishs ($[0.7, 1]$). In this table, we see that our method

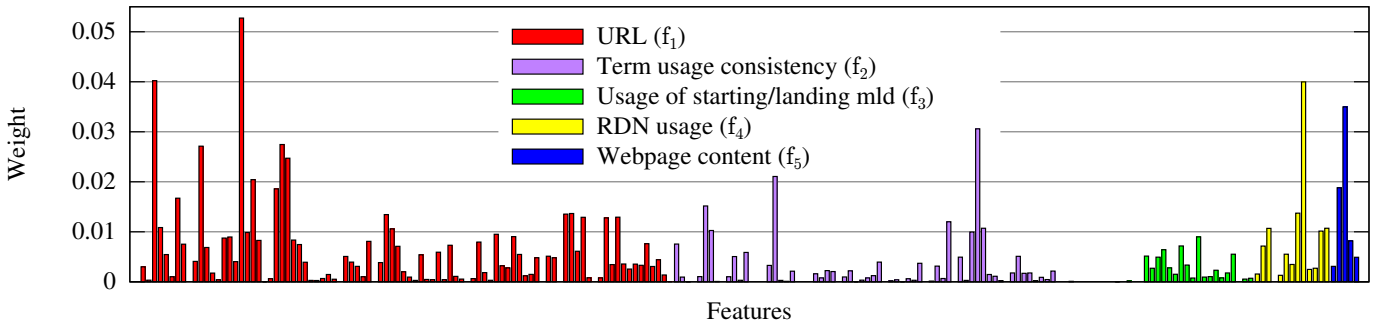


Fig. 6: Weight of features in gradient boosting classification model according to the set f_n they belong to.

achieves significantly high precision (0.92–0.98) and high recall (around 0.95) for all languages. Hence, the F_1-score, which is the harmonic mean of precision and recall, is also significantly high (0.93–0.97). The false positive rate is significantly low, i.e., in the range of 0.0005–0.004, across all languages.

TABLE 5: Detailed accuracy evaluation of the phish detector alone for six languages

Language	Precision	Recall	F_1-score	FP Rate	AUC
English	0.975	0.952	0.964	0.0002	0.999
French	0.937	0.952	0.944	0.0032	0.998
German	0.944	0.952	0.947	0.0028	0.997
Portuguese	0.949	0.952	0.950	0.0025	0.998
Italian	0.928	0.952	0.940	0.0037	0.998
Spanish	0.919	0.952	0.935	0.0042	0.998

In many large-scale, real-world scenarios (especially in web security domain), a machine learning model is considered usable only if it achieves high precision (e.g., 0.9 or 0.95) with significant recall (e.g., 0.5 or 0.6) [28]. In order to test our method against this criterion, we evaluated how recall of the proposed method changes with precision by varying the discrimination threshold from 0 to 1. The result is shown in Fig. 5b where we see that when the precision is higher than 0.9, the recall for all languages is significantly high and is always in the range of 0.60–0.96.

Obtaining high accuracy values in classification at a given moment in time does not guarantee this accuracy will remain high over time. Phishing techniques evolves and characteristics captured by a classification model may not be valid forever. To see how the detection capability (recall) of the model degrades with time, we split *phishTest* in five subsets of equivalent size ($\approx 2,500$) according to the time they were captured. Fig. 5c depicts the evolution of recall value computed on these subsets (note that the latest entry in *phishTrain* was gathered on September 9, 2015). In this figure, we see that recall remains high (≈ 0.95) across the time period of January 22 to June 10, 2016. This implies that the model does not have to be retrained or updated frequently, making *Off-the-Hook* suitable for real-life deployments. The global recall on the 13,309 phishs of *phishTest* is 0.951 which is consistent with the recall obtained with its subset *phishTest1K* (0.952) in Tab. 5.

6.2.2 Feature Analysis

We now analyze the impact of each feature set and individual features from Sect. 4.3 on phish classification. Fig. 6 shows the weight of the 210 features. While we see that most features are relevant, features related to the usage of starting and landing *mld* (f_3) have a small impact on classification with a weight always lower than 0.01. The same observation holds for most term usage consistency features (f_2), apart from a few exceptions having significant weight (> 0.01). This is also shown in Tab. 6 where we see that features from f_2 and f_3 have the lowest average weight. On the other hand, features form f_4 and f_5 have the highest impact on classification with an average weight of 0.0084 and 0.0139 per feature respectively. One feature from each of these sets has a strong weight (> 0.03) as can be observed in Tab. 7, which shows the ten most important features of the classification model. URL features from f_1 predominantly impact the classification with a global weight of 0.578 for this feature set and most features having a significant weight as it can be seen in Fig. 6. Moreover, f_1 has six features in the ten most important of the model including the two top features (Tab. 7).

Looking at the most significant features in Tab. 7, we see that they come from different sets (f_1, f_2, f_4, f_5) confirming the relevance of the feature set definition process. However, we also notice that these most relevant features rely predominantly on the data source *internal logged links* (features ranked $1, 5, 6, 8, 9$ and 10). Internal logged links correspond to loaded resources controlled by the page owner and unconstrained. These are parts where phishers use obfuscation techniques by embedding keywords in *FreeURL* to lure their victims.

TABLE 6: Feature set weight.

Set	Weight	Avg weight per feature
f_1	0.578	0.0062
f_2	0.185	0.0027
f_3	0.058	0.0018
f_4	0.109	0.0084
f_5	0.070	0.0139
f_{all}	1.0	0.0047

TABLE 7: Top ten features of the classification model.

Rank	Set	Feature	Weight
1	f_1	mean # of terms in internal logged links	0.053
2	f_1	length of starting URL	0.040
3	f_4	count of external HREF links	0.040
4	f_5	count of input fields	0.035
5	f_2	$Hellinger(D_{intlog}, D_{start})$	0.031
6	f_1	median length of internal logged links	0.027
7	f_1	length of landing URL	0.027
8	f_1	SD length of internal logged links	0.025
9	f_2	$Hellinger(D_{intlog}, D_{text})$	0.021
10	f_1	SD # of terms in internal logged links	0.020

6.3 Target Identification

To assess the performance of target identification, we used *phishBrand* dataset. Since target identification can provide up to three

candidate targets for each phish, we evaluate the likelihood of the correct target being in top-1, top-2 and top-3 results provided by our method. Tab. 8 presents the count of correctly identified targets, unknown targets and missed targets considering these three sets. The last column gives the success rate of each method. The 17 pages with unknown target correspond to phishs that contained only some input fields and no hint about the target. Hence, we were not able to infer the target for these phishs with manual analysis. Yet, we include these phishs with unknown targets for the computation of the success rate. In Tab. 8, we see that the accuracy of identifying the correct target (top-1) is 90.5%, and if the criteria for identifying the correct target is expanded to top-3 then the accuracy increases to 97.3%. These results are comparable to the best state-of-the-art method for target identification [26] that achieves a success rate of 92.1%. It is of note that 311 phishs had only one identified potential target (top-1) and no alternative targets. Hence, more than half of the phishing warnings raised by *Off-the-Hook* would provide one redirection link to a unique target.

TABLE 8: Target identification results

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%

6.4 Overall System Accuracy

To see how the target identification system complements the phish detection system in *Off-the-Hook*, we fed the former with misclassified legitimate webpages identified in Sect. 6.2.1. As presented in Sect. 3.4, if an identified target *RDN* matches the *RDN* of the *landing URL*, the website is considered as legitimate. *Off-the-Hook* considers this criteria to avoid displaying warning messages for legitimate webpages. Tab. 9 compares false positives and precision of the standalone phish detector and of *Off-the-Hook*. Accuracy, Recall and AUC are not compared since they were not drastically improved (0.999/0.952/0.999). We can see that overall the system combination reduces false positives by over 50%, missclassifying only 166 legitmate webpages as phish out of 200,000. This gives a final *FP rate* of 0.0008 and a precision of 0.975 to *Off-the-Hook*, which is comparable to the best state-of-the-art phish detection system [2].

TABLE 9: Accuracy improvement using target identification.

Language	Phish Detector			*Off-the-Hook*		
	FP	FP Rate	Pre.	FP	FP Rate	Pre.
English	24	0.0002	0.975	19	0.0002	0.980
French	64	0.0032	0.937	30	0.0015	0.969
German	57	0.0028	0.944	33	0.0016	0.966
Portuguese	51	0.0025	0.949	35	0.0017	0.964
Italian	74	0.0037	0.928	25	0.0012	0.974
Spanish	84	0.0042	0.919	24	0.0012	0.975
Overall	354	0.0018	0.955	**166**	**0.0008**	**0.975**

6.5 Computational System Requirements

To assess the time required for the computation steps presented in Sect. 3.4 we report them as column headers in Tab. 10. It depicts the mean, median and standard deviation of the time for the full process of phish detection (1-7a) and target identification (1-7b). This table also includes the performances excluding the time needed for the data collection (1-4): (5-7a) and (5-7b). The statistics regarding 6099 legitimate websites were gathered over a 6 weeks period of normal web browsing while 154 phishs were visited on purpose. The laptop on which *Off-the-Hook* Firefox version was installed has a 2.7GHz Intel Core i5 processor and 16GB memory. We can see that the time needed to classify websites is under 250 ms. In case the *phish detector* makes a false positive decision, thanks to the *target identifier*, we are able to remove the *loading screen* in around 1.5 seconds (1772 - 220 = 1552 ms).

TABLE 10: Processing time in milliseconds for legitimate webpages and phishs (column headers refer to steps in Fig. 3).

		(1-4)	(5-7b)	(5-7a)	(1-7b)	(1-7a)
Legitimate	**Mean**	164	2,334	328	2,498	492
	Median	56	1,655	154	**1,772**	**220**
	Stdev	497	2,369	571	2,482	829
Phishing	**Mean**	72	1,986	150	2,058	222
	Median	36	1,848	90	**1,877**	**127**
	Stdev	158	1,615	297	1,642	297

TABLE 11: Memory footprint of *Off-the-Hook* components.

	Dispatcher	Phish detector	Target identifier	Overall
Average	21.93 MB	120.62 MB	152.50 MB	295,05 MB
Stdev	0.22 MB	2.42 MB	4.36 MB	4.76 MB

Tab. 11 presents the average memory usage of the three main components of *Off-the-Hook* system. In this table it can be seen that while the *dispatcher* has a low (and stable) memory consumption of 22 MB, the modules required for classification increase the memory consumption of the *phish detector* to 120 MB. The two instances of the *target identifier* required 76 MB each, which leads to an overall memory consumption of 295 MB for the proposed *Off-the-Hook* system.

7 Usability Study

To test first-time users' perception of *Off-the-Hook* from a usability perspective, we conducted two user studies. The *lab test* investigated its intuitive understandability and users' behavioral responses in a lab setting. The *field test* involved installing it on the participant's own laptop to study potential interference with general laptop use and produce genuine reactions in a naturalistic setting. This section describes both tests and their main results. All the questionnaires used in the tests are publicly available [29].

7.1 Test settings

7.1.1 Lab test

The purpose of this study was to display *Off-the-Hook* to a naive user, inquiring whether they understand the warning and safe toast notification's contents, how they would continue after seeing the warning banner, and whether they prefer *Off-the-Hook* to its well-established alternative, Firefox's warning of phishing (called "web forgery" there). The lab test is a qualitative study that makes no statistical claims.

Participants: Ten participants were recruited for the lab test, most via emails sent to student mailing lists and some by contacting them in person. Results from the background questionnaire are provided in Tab. 12.

TABLE 12: Results from background questionnaire

		Lab test	Field test
Number of participants		10	18
Age	20–24	4	10
	25–29	6	4
	30–34	0	3
	> 50	0	1
Gender	Female	3	11
	Male	7	7
Education	High school	0	5
	Bachelor's degree	4	7
	Master's degree	6	4
	Doctoral degree	0	2
Main purpose of laptop use	Work	3	5
	Study	5	7
	Personal	2	6
Internet use (hours per day)	1–3	3	7
	3–5	4	6
	5–7	2	4
	> 7	1	1
Browser	Mozilla Firefox	4	7
	Google Chrome	5	11
	Safari	1	0
Knowledge about phishing	Knowledgeable	10	15
	Uninformed	0	3

Materials: The test involved the participant opening three links provided in fake emails on a test laptop with Windows 10, Google Chrome and *Off-the-Hook* installed. The links were verified before each test session to trigger the appropriate banners. An image of Firefox's web forgery warning was additionally shown to the user.
Procedure: We advised the participant to open three email links consecutively. The first link led to a safe website, indicated by the green badge on the navigation bar. The second link brought up the warning message. We asked the participant various questions concerning the banner's content and appearance. Next, we showed the participant a picture of Firefox's web forgery warning and asked them to compare the two banners. Finally, a third link was opened, first leading to the loading screen and then to the safe toast notification. We gathered the participant's assessments of it. At the end of the test the participant filled out an additional questionnaire containing similar questions as asked during the test, with pre-provided answer options on a Likert scale.
Ethical considerations No personal information apart from the explicit discussion and questionnaires was gathered.

7.1.2 Field test

The purpose of this test was twofold: (a) to see whether the presence of the application has a negative impact on user experience, and (b) to test how users react to the banners during normal laptop use.
Participants: 18 people took part in the test. Most were recruited via emails sent to student mailing lists, and some by contacting them in person. Results from the background questionnaire are provided in Tab. 12. It is worth noting that our test subjects were all university students or had graduated from a university. Their educational backgrounds range across all levels of higher education and their time usage of Internet is equally distributed though, which provides a fair representativity.
Materials: The study was conducted on each participant's own laptop by installing *Off-the-Hook* on the user's own laptop for one week and a fake version for another week. The fake version displays the same icon on the navigation bar as *Off-the-Hook*, but without performing any other processes.
Procedure: The within-subjects approach was used in this study. Each participant used both the real and fake version of *Off-the-Hook* for one week, and the order of the real and fake application was evenly distributed among the participants. The only prior information given to the participants was that we are installing a browser add-on. The same questionnaire was filled in after both weeks, asking the user to choose between *Strongly disagree* and *Strongly agree* on a five-point Likert scale as reactions to the statements displayed in Tab. 13. There was further space in the questionnaire for freely written qualitative answers. After the two weeks, the participants were asked whether they remembered having been exposed to any of the banners during the test period. Those who did answered the same questionnaire as in the lab test, inquiring about the understandability, appearance and usability of each banner.
Ethical considerations: No personal information apart from the explicit questionnaires was gathered. We did not record the participants' browsing history.

7.2 Results

7.2.1 Lab test

All users understood the purpose of the warning banner, although this took more time for some. There were no major misunderstandings of textual content. The continuation options were recognized fast and evaluated as useful by the majority of participants. The link to the target site was noticed and understood without complications. When the banner appeared, every participant chose to close the tab, the majority (6) from the browser and the rest (4) from the banner. Seven out of ten users preferred *Off-the-Hook* to Firefox's web forgery warning. Firefox's was deemed more difficult to understand due to technical jargon. Further, Firefox's banner contains no link to the target website, and many users preferred *Off-the-Hook* due to this. On the negative side, *Off-the-Hook* was considered by some to be less "professional" in appearance than Firefox's banner. Participants were divided on the safe toast notification, six out of ten finding it useful. Some considered it redundant given that the navigation bar icon already displays the relevant information of the website being safe. Problematically, no user correctly guessed the reason why some safe sites brought up the safe toast notification but others did not.

7.2.2 Field test

During the two-week test, 12 of the 18 participants had seen the loading screen, nine had seen the warning banner, and nine had seen the safe toast notification. One participant had seen another phishing warning instead of *Off-the-Hook*. Results from the questionnaire filled by all participants after having first the fake and then the real *Off-the-Hook* installed, or vice versa, are provided in Tab. 13.

A difference between the real and fake *Off-the-Hook* was found 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*, whereas seven agreed after having the real *Off-the-Hook* installed, giving a statistically significant result with both The Wilcoxon signed ranks test and the sign test. However, a similar slowing effect was not found in general laptop functioning. Since the most likely reason why *Off-the-Hook* would affect browsing speed would be that it slows down the whole laptop, it is likely that many participants simply did not notice this

TABLE 13: Results of Wilcoxon signed ranks test (two-tailed) and sign test (two-tailed) between answers to fake and real *Off-the-Hook*. Last column indicates whether the real software elicited more agreement (Agree) or disagreement (Disagree) than the fake one.

Claim	Wilcoxon signed ranks test					Sign test			Direction
	Ties	Z	p	W / Crit.	Signif.	Z	p	Signif.	
My browsing has slowed down	8	-2.80	0.005	0 / 8	Yes	3	0.003	Yes	Agree
I have been annoyed by the presence of the program	6	-2.82	0.005	3 / 13	Yes	2.83	0.005	Yes	Agree
My computer has slowed down in general	6	-1.37	0.170	21.5 / 13	No	1	0.317	No	-
The program has interfered with my browsing	4	-2.82	0.005	7.5 / 21	Yes	2.89	0.004	Yes	Agree
The program has interfered with my use of the laptop in general	9	-2.67	n/a	0 / 5	Yes	2.24	0.025	Yes	Agree
Use of my laptop has been as enjoyable as before	5	-2.90	0.004	4 / 17	Yes	2.64	0.008	Yes	Disagree
The program has prevented me from doing something I wanted	10	-1.40	n/a	8 / 3	No	1.41	0.157	No	-
The browser has crashed more often than usual	11	-0.25	n/a	12.5 / 2	No	0	1	No	-
Other programs have crashed more often than usual	12	-0.94	n/a	6 / 0	No	1	0.317	No	-

for other aspects of their laptop use. Possibly, the loading screen may also have been judged by some as slowing down browsing.

Off-the-Hook also seemed to have a negative impact on laptop use experience, with ca. half of the users agreeing with the statement *I have been annoyed by the presence of the program* for the real but not the fake app. Most of these users had witnessed the warning banner mistakenly appearing on safe sites, which is likely a significant cause for this negative result. As demonstrated in Sect. 6.4, the overall false positive rate of *Off-the-Hook* is low. However, among safe sites that brought up the warning were a university email site and an online bank, both of which were popular among test participants. Even if the false positive rate is low in principle, only one instance of it suffices to negatively impact user experience. We discuss this issue further in Sect. 7.3.

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. Results concerning the proposition "The program has prevented me from doing something I wanted" further indicate that *Off-the-Hook* did not stop the users from browsing as they wished, in case they wanted to disregard it. There was one exception to this, where a user reported the loading screen blocking access to a website. Presumably this was due to the slow speed of the participant's laptop, since the participant reported seeing the loading screen but none of the banners. This raises the issue of whether an exit link should be incorporated to the loading screen to allow the user to close the tab during the loading. The majority of participants understood the warning banner correctly, and it received a generally positive rating on text content and visual appearance. All the continuation links were considered useful on average. Nine users had seen the safe toast notification, five of whom it had made feel safer about the website. Some considered it difficult to interpret, and most would have preferred more information on it. Nevertheless, the majority found it unnecessary.

It is worth noting that the field test featured a moderate number of 18 participants. The Wilcoxon signed ranks test considers the number of data points in calculating the critical value of the W statistic. Nevertheless, it rendered strongly significant results for all statements in Table 13 except one: *The program has interfered with my use of the laptop in general*.

7.3 Discussion

Both tests indicate that users generally interpreted the warning banner correctly, and were thus able to make an informed decision on how to proceed. Its textual understandability and visual appearance were also deemed appropriate on average. Most users found all the continuation options useful, and no user advocated the removal of any of them. The ease of understanding of the textual content and the variety of continuation options were both appreciated in comparison to Firefox' phishing warning. The negative impacts on user experience were largely due to false positives, and can be remedied by establishing a global whitelist containing popular websites that are falsely treated as phish by *Off-the-Hook*.

The interpretation and appreciation of the safe toast notification were somewhat less clear, as some considered it redundant but others found it helpful. No user correctly guessed why it appears on some sites and not others, which is potentially problematic. Removing the safe toast notification would be a possible remedy for this, given that the same information is also present at the navigation bar. The main usability problem with this would be that the loading screen may result in confusion or suspicion, since it currently has no information. Adding a simple text, such as "checking" along with a progress bar would remedy this issue.

In summary, the following modification suggestions emerged from the usability studies:

- establishing a global whitelist to exclude common false positives
- removing the safe toast notification
- adding a small explanatory text to the loading screen
- adding an exit link to the loading screen

8 Related Work

Phish detection: Analysis of the content [10], [27] and code execution (*e.g.* the use of javascript, pop-up windows, etc.) [38] of a webpage provides relevant information to identify phishs. Some detection methods rely on URL lexical obfuscation characteristics [12], [33] and webpage hosting related features [30], [39] to decide if a webpage is a phish. The visual similarity of a phish with its target was also exploited to detect phishs [31], [40]. Visual similarity analysis presupposes that a potential target is known a priori though, limiting its applicability. In contrast, *Off-the-Hook* identifies phishs and *discovers* their targets.

Multi-criteria methods [1], [2] have been proved the most efficient to detect phishs. These techniques use a combination of webpage features (HTML terms, links, frame, etc.), connection features (HTML header, redirection, etc.) and host based features (DNS, IP, ASN, geolocation, etc.) to infer webpage legitimacy. The identification method uses machine learning techniques fed with hundreds of thousands of these features, which are mostly static and learned from training sets containing data such as IP address, Autonomous System Number (ASN), bag-of-words for different data sources (webpage, URL, etc.). This limits the generalizability of the approach that is context-dependant and it requires large training datasets, numbering hundreds of thousand

TABLE 14: Phish detection systems evaluation and accuracy comparison

Technique	Testing set Legitimate	Testing set Phish	Legitimate set	Train /Test	Leg /Phish	Evaluation	FPR	Precision	Recall	Accuracy
Ma *et al.* [30]	15,000	20,500	DMOZ	1/1	3/4	cross-validation	0.001	0.998	0.924	0.955
Whittaker *et al.* [2]	1,499,109	16,967	several	6/1	90/1	old/new	0.0001	0.989	0.915	0.999
Thomas *et al.* [1]	500,000	500,000	several	4/1	1/1	cross-validation	0.003	0.961	0.734	0.866
Chen *et al.* [31]	404	1,945	top Alexa	9/1	1/5	cross-validation	0.007	0.992	1	0.994
Cantina [27]	2,100	19	English	-	110/1	no learning	0.03	0.212	0.89	0.969
Cantina+ [32]	1,868	940	several	1/4	2/1	old/new	0.013	0.964	0.955	0.97
Xiang *et al.* [10]	7,906	3,543	several	-	2/1	no learning	0.019	0.957	0.9	0.955
Ramesh *et al.* [25]	1,200	3,374	top Alexa	-	1/3	no learning	0.005	0.998	0.996	0.996
PhishStorm [33]	48,009	48,009	DMOZ	9/1	1/1	cross-validation	0.014	0.984	0.913	0.949
Kausar *et al.* [34]	71	89	unknown	81/1	1/1	unknown	0.112	0.888	0.887	0.875
PhishShield [35]	250	1,600	PhishTank	-	1/6	no learning	0.0004	0.999	0.971	0.966
Jain *et al.* [36]	405	1,120	top Alexa	-	1/1	no learning	0.015	0.993	0.861	0.894
Varshney *et al.* [37]	2,500	500	top Alexa	-	5/1	no learning	0.076	0.723	0.995	0.936
Off-the-Hook	100,000	1,000	English	1/10	100/1	old/new	0.0002	0.980	0.952	0.999
Off-the-Hook	200,000	2,000	several	1/20	100/1	old/new	0.0008	0.975	0.951	0.999

of webpages [2]. Finally, their computation is reportedly costly [1] and use [2] some proprietary features preventing usage on end-user devices, in contrast to our fully client-side approach.

Client-side techniques: Alternative light-weight and simplified implementations of multi-criteria methods [3] do not consider that many features and are used on the client-side, but are less efficient than implementing the same technique server-side [2]. In a similar manner, detection techniques relying on fixed heuristics (Alexa ranking, Google pagerank, etc.) are deployed as client-side phishing protection systems [34], [35], [36] but also exhibit high rate of missclassification [34], [36] or are not able to render decisions for all webpages [35]. Furthermore, static heuristics in client-side solutions can be easily discovered by an attacker and circumvented. Obfuscating the detection model which uses static features (e.g. bag-of-words) [3] does not prevent such attackers to perform adversarial machine learning [7]. In contrast, our detection model considers phishers' limitations, which provides better accuracy and resilience to this class of attack.

Other methods focused, as we do, on the study of terms that compose the data sources of a webpage [12], [33] and admit client-side implementation. Cantina [27], [32] was among the first systems to propose a lexical analysis of terms that comprise a webpage. In Cantina [27] key terms are selected using TF-IDF to provide a unique signature of a webpage. Using this signature in a search engine, Cantina infers the legitimacy of a webpage. A similar method [10], based on TF-IDF and Google search, checks for inconsistency between a webpage identity and the identity it impersonates to identify phish. The main difference between these methods and ours is language independence since these methods rely on TF-IDF computation to infer their keyterms. An alternative to TF-IDF computation is to use domain name and page title to perform the search query and confirm a webpage as legitimate if its domain name appear in the top-6 results of Google's query [37]. Nevertheless, all these techniques [10], [27], [32], [33], [37] need to perform search queries for each page to analyze which is time and resource consuming. In contrast, *Off-the-Hook* resorts to search queries in a few occasions when the phishing detector first identifies a webpage as a phish and target identification is thus required. The analysis of webpages identified as legitimate by the phishing detector do not trigger any search query. Those represent the vast majority (>99%) of visited webpages.

Accuracy comparison: Tab. 14 presents comparative accuracy results of *Off-the-Hook* to the most relevant state-of-the-art systems. Techniques at the top only admit centralized deployment, while others admit a client-side implementation as *Off-the-Hook*. We present the size of the testing sets used to evaluate each system and the provenance of the legitimate set, showing how representative the set is. For example, using popular websites (such as top Alexa sites) [25], [31] as the legitimate set is not representative. The ratio of training to testing instances (*Train/Test*) indicates the scalability of the method and the ratio of legitimate to phishing instances (*Leg/Phish*) shows the extent to which the experiments represent a real world distribution ($\approx 100/1$) [2], [27]. We also identify the evaluation method (e.g., cross-validation vs. training with old data and testing with new data). Finally, we present several metrics for assessing the classification performance. If data for any of the columns were missing from the original paper describing the system, we estimated them. For comparison purposes, if several experimental setups were proposed in a paper, we selected the most relevant to assess their practical efficacy using the following ordered criteria:

1) learning and testing instances are different,
2) the ratio of legitimate webpage to phish in the testing set is representative of real world observations ($\approx 100/1$),
3) the learning set is older than the testing set,
4) the false positive rate (FPR) is minimized.

A low false positive rate is paramount for a phish detection technique, since this relates to the proportion of legitimate webpages to which a user will be incorrectly denied. We can see that among the most relevant state-of-the-art techniques, only three [2], [30], [35] have comparable false positive rates to ours (≤ 0.001). Two of them [2], [30] do not admit client-side implementation, thus raising concerns like user privacy and delay in phish identification [5] that we previously identified.

The technique proposed by Ma *et al.* [30] has a lower accuracy than in our system ($0.955 < 0.999$). In addition, they use a testing set that does not represent real world distribution (legitimate to phish ratio of 3:4). Further the parameters they used for cross-validation (1:1 ratio for learning to testing instances) do not assess the scalability of the approach. Whittaker *et al.* [2] report results similar to us in several metrics. However, they use a *huge training set (>9M instances)* and their test set is actually *smaller* than the training set (a sixth, at 1.5M)! Scalability and language-/brand-independence are likely to be poor since they use 100,000 mostly static features (bag-of-words). PhishShield [35] is a client-side solution that has high precision and low false positive rate. It was evaluated on a very small testing dataset of 1,850 instances containing only 250 legitimate webpages, which raises concerns

about how representative the results are. Nevertheless, this system has lower accuracy (0.966) than *Off-the-Hook* and is not able to render a decision for every analyzed webpage, i.e. more than 2% of the pages evaluated did not get a decision.

In contrast to the state-of-the-art in phish detection, *Off-the-Hook* is a client-side application that is context-independent, resilient to phishing attacks evolution and does not rely on access to external sources, while performing better than or as well as the state-of-the-art.

Target identification: One proposal [25] was to use a similar technique as Cantina with keywords retrieval and Google search to discover a list of potential target as the top results of the search, but the authors do not report accuracy figures for target identification. Similarly, extracting the logo from a webpage screenshot and using it in Google Image Search leads to identify the target of 87.0% of phishs in [41]. HREF links have been used to build community graphs of webpages. By counting the mutual links between two webpages and further performing visual similarity analysis between suspicious webpages, Wenyin *et al.* [26] identify the target of a given phish with an accuracy of 92.1%. However, this technique is slow because of the need to crawl many additional websites to build the community graph. Conditional Random Fields and Latent Dirichlet Allocation (LDA) have been applied to phishing email content to identify their target [22] with a success rate of 88.1%. The technique we propose, in contrast to previous techniques is language-independent for keyterms inference. It is as efficient as any state-of-the-art solutions achieving a success rate of 90.5-97.3%.

9 Summary and Conclusion

9.1 Meeting Design Goals

Accuracy (R1): Sect. 6.2 shows that our feature set yielded results that outperform most previous work. The main reason is the new separation scheme applied to data sources related to their level of control and constraints (Sect. 4.1). This is evident from the weight in classification model of features from the set f_1 that comprises URL features separated accordingly to constraint and control considerations. Accuracy is improved by the target identifier, which helps reducing false positives by over 50% without impacting negatively other accuracy measures (Sect. 6.4). This makes *Off-the-Hook* comparable to the best existing technique [2] in term of accuracy (FP Rate < 0.001 and recall > 0.95) while relying on fewer features and less training data.

Context-independent detection (R2): As presented in Sect. 4.3, none of our 210 features rely on static observations (e.g word, IP address, ASN). It makes phish detection context-independent and very accurate regardless of language or targeted brand.

Temporal resilience (R3): The separation of controlled and constrained data sources models inherent limitations in webpage composition, that will remain over time. It explains why the phish detection rate (recall) of *Off-the-Hook* remains high and constant several months after the detection model was trained, as highlighted in Sect. 6.2.1. The detection model is also more robust to adversarial machine learning attacks since, while knowing features used for classification, phishers cannot modify constrained and uncontrolled part of their phishs. Hence, they cannot easily circumvent detection.

Resilience to dynamic phishing (R4): It is guaranteed by design since *Off-the-Hook* analyses the actual webpage content depicted in the browser to render its decision.

User privacy (R5): With the fully-client-side implementation, all information the phish detector needs is available locally and no browsing information is shared with any third party. The target identification process requires some requests to search engine, which can leak some browsing information. However, this process is triggered only when a page is first identified as phish, which occurs rarely.

Effective protection (R6): The fast decision making of *Off-the-Hook*, with a median time lower than 0.2 seconds to block interaction and warning in less than 2 seconds, prevents users from disclosing any sensitive information to a phisher (Sect. 6.5). The several banners and their global design are deemed appropriate for phishing protection as highlighted in the usability studies (Sect. 7). *Off-the-Hook* warning is preferred over that of Firefox. Similarly, the new continuation option to the target of the phish received positive feedback from participants who would appreciate such a feature in warnings from other protection software.

9.2 False Warnings

We now discuss possible causes for false warnings.

The term extraction technique that guarantees language independence raised some issues. We chose to split strings at any non-alphabetic character and to only consider terms composed of at least three characters to discard stop words and recurrent short terms having no significance. This negatively impacted term distribution comparisons. Long subdomains such as *theinstantexchange* or *insuranceservicenow* were considered as single term. In contrast, short domain name string corresponding to brand and composed of separating characters (digit, hyphen, etc.) such as *dl4a*, *s2mr* or *e-go* were split and the resulting terms were discarded as too short. The inconsistent usage of abbreviations or acronyms like *intl* for *international* also had a negative impact. Most legitimate webpages misclassified by the phish detector had one of these characteristics although they were largely confirmed as legitimate by the target identifier, preventing to raise a warning.

Other misidentified webpages were hosting content not related to their domain name and mostly very little and generic content. One common example was login webpage for private webmail service using popular webmail provider (e.g. Outlook, Gmail). These webpages were identical to the regular login page of the provider but hosted on a different domain name e.g. Outlook login interface hosted on *mail.aalto.fi*. This misled the phish detector and the target identifier, which identified the website of the service provider as the target i.e. *live.com* for Outlook. From a system and human perspectives these webpages are similar to phishing webpages. The only means to assert the trustworthiness of these websites is using prior knowledge about the private webmail service being hosted on a given domain name, since in most cases these websites did not use SSL/TLS certificate. Having this knowledge, one can add an exception in the local whitelist to prevent warnings from appearing again. Alternatively, to avoid disturbance to first time users, a global whitelist could be implemented. as pointed in Sect. 7.3. This can be fed with crowdsourced feedback from *Off-the-Hook* users related to their choice to *proceed* and *do not display the warning again* for a website raising a warning. Such a solution would need though further verification and protection mechanism to prevent phishers from manipulating this global whitelisting system in order to bypass detection.

Some empty/unavailable webpages and parked domain names were also identified as phishs. The former is explained by the

lack of information contained in empty/unavailable webpages. Several parked domain names use similar composition schemes and obfuscation techniques as phishing domains [42] and may have been previously used in such a malicious activity [43]. From the point of view of our classification system, some parked pages have the same characteristics as phishs. This misclassification of unavailable and parked domain names is not of major concern though since, for the former no content access is prevented since the link point empty resources. For the latter, domain parking is considered as spam by major Internet actors (*e.g.* Google) and some efficient state-of-the-art techniques [42] can be applied to discard these webpages from phishing identification.

9.3 Evasion Techniques

One way to evade detection is to use IP-based URLs. A lower detection rate was observed for such phishs. However, relying on IP address rather than domain names deprives phishers from the flexibility brought by the DNS to change the hosting location of their phishing content while keeping the same link. Moreover, IP blacklisting is widely used to prevent access to malicious hosting infrastructure.

Another evasion technique is to limit the text content available in a webpage: use few external links, do not load external content and build short URLs. We observed some of these techniques actually being used individually in webpages used for evaluation. They did not impact classifier performance because even though they prevent some features from being computed, others, such as those based on title, starting/landing URL and logged links could still lead to effective detection of phishs. Simultaneous use of multiple evasion techniques may impact classifier performance. However, using such subterfuges would degrade the quality of the phish and reduce the number of victims.

Off-the-Hook is resilient to dynamic phishing in which phishers serve different content depending on who the client is. However, delayed generation of DOM elements or dynamic Javascript on a webpage may affect the feature extraction process. Dynamically generated elements may not be present just after a webpage is loaded and thus not captured in our data sources. However, this would not have a severe impact on classification accuracy. Some features related to HTML source code may not be computed accurately but features related to starting URL, landing URL, redirection chain and logged links will be accurately computed anyway and have shown to be efficient at detecting phishs (Section 6.2). Furthermore, dynamic generation of webpage elements may tip-off potential victims noticing changes taking place on the rendered webpage after long delays. Techniques to detect modifications in webpages after loading have been proposed [44] and could be used to trigger a new data source extraction for *Off-the-Hook* to rerun the decision process.

We consider *RDN*s that are not in the redirection chain as being outside the control of an attacker: *external*. However, knowing our scheme a phisher can include links to domain names he registered in a phish, while not using them for redirection. The phisher can thus freely define URLs that our system assumes to be uncontrolled. This can decrease the recall of our method and ease circumvention. Retraining the model once this attack starts to be used would cope with this issue. *External links* features would be given lower weight in the classification model since they would start to be less discriminative.

A final probable evasion technique is to use typosquatting domains and misspelled terms in the different data sources we analyze. When different but similar terms like *paypal*, *paypaI* or *paipal* are used in different sources, our distributions comparison metric would not infer any similarity. The classifier would thus probably conclude that the webpage is legitimate. However, the presence of reference links to the target would disclose the real target. In addition, misspellings may tip-off potential victims.

Acknowledgments

This work was supported in part by the Academy of Finland (grant 274951) and Intel Collaborative Research Center for Secure Computing (ICRI-SC).

References

[1] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time url spam filtering service," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2011, pp. 447–462.

[2] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in *Proceedings of the 2010 Network and Distributed System Security (NDSS) Symposium*, 2010.

[3] Google, "Safe browsing." [Online]. Available: https://www.chromium.org/developers/design-documents/safebrowsing

[4] Phishtank, "Out of the Net, into the Tank." [Online]. Available: https://www.phishtank.com/

[5] X. Han, N. Kheir, and D. Balzarotti, "Phisheye: Live monitoring of sandboxed phishing kits," in *ACM CCS*, 2016, pp. 1402–1413.

[6] M. Al-Daeef, N. Basir, and M. Saudi, "A review of client-side toolbars as a user-oriented anti-phishing solution," in *Advanced Computer and Communication Engineering Technology*, 2016, pp. 427–437.

[7] B. Liang, M. Su, W. You, W. Shi, and G. Yang, "Cracking classifiers for evasion: A case study on the google's phishing pages filter," in *International Conference on World Wide Web*, 2016, pp. 345–356.

[8] D. Akhawe and A. P. Felt, "Alice in warningland: A large-scale field study of browser security warning effectiveness," in *Proceedings of the 22nd USENIX Conference on Security*, 2013, pp. 257–272.

[9] APWG, "Phishing Activity Trends Report," APWG, Tech. Rep. 3Q2016, 2016.

[10] G. Xiang and J. I. Hong, "A hybrid phish detection approach by identity discovery and keywords retrieval," in *Proceedings of the 18th International Conference on World Wide Web*, 2009, pp. 571–580.

[11] Y. Pan and X. Ding, "Anomaly based web phishing page detection," in *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC)*, 2006, pp. 381–392.

[12] A. Le, A. Markopoulou, and M. Faloutsos, "PhishDef: URL names say it all," in *Proceedings of IEEE INFOCOM*, 2011, pp. 191–195.

[13] S. Marchal, J. François, R. State, and T. Engel, "Proactive discovery of phishing related domain names," in *Research in Attacks, Intrusions, and Defenses*, 2012.

[14] SSG@Aalto, "Off-the-Hook - A phishing prevention system." [Online]. Available: https://ssg.aalto.fi/projects/phishing/add-on.html

[15] KangoExtensions, "Cross-browser extension framework." [Online]. Available: http://kangoextensions.com/

[16] BestToolBars, "Build addons and toolbars for all browsers." [Online]. Available: http://www.besttoolbars.net/products/cross-browser-extensions-framework/

[17] S. Egelman, L. F. Cranor, and J. 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*, 2008, pp. 1065–1074.

[18] C. Wharton, J. Rieman, C. Lewis, and P. Polson, "The cognitive walkthrough method: A practitioner's guide," in *Usability Inspection Methods*, J. Nielsen and R. L. Mack, Eds. John Wiley & Sons, Inc., 1994, pp. 105–140.

[19] S. Hardy, M. Crete-Nishihata, K. Kleemola, A. Senft, B. Sonne, G. Wiseman, P. Gill, and R. J. Deibert, "Targeted threat index: Characterizing and quantifying politically-motivated targeted malware," in *23rd USENIX Security Symposium*, 2014, pp. 527–541.

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

[21] Z. Li, S. Alrwais, X. Wang, and E. Alowaisheq, "Hunting the red fox online: Understanding and detection of mass redirect-script injections," in *IEEE Symposium on Security and Privacy (SP)*, 2014, pp. 3–18.

[22] V. Ramanathan and H. Wechsler, "Phishing detection and impersonated entity discovery using conditional random field and latent dirichlet allocation," *Computers & Security*, vol. 34, pp. 123–139, 2013.

[23] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[24] P. Buhlmann and T. Hothorn, "Boosting algorithms: Regularization, prediction and model fitting," *Statistical Science*, vol. 22, no. 4, pp. 477–505, 2007.

[25] G. Ramesh, I. Krishnamurthi, and K. S. S. Kumar, "An efficacious method for detecting phishing webpages through target domain identification," *Decision Support Systems*, vol. 61, pp. 12–22, 2014.

[26] L. Wenyin, G. Liu, B. Qiu, and X. Quan, "Antiphishing through phishing target discovery," *IEEE Internet Computing*, vol. 16, no. 2, 2012.

[27] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: A content-based approach to detecting phishing web sites," in *Proceedings of the 16th International Conference on World Wide Web*, 2007, pp. 639–648.

[28] N. Singh, H. Sandhawalia, N. Monet, H. Poirier, and J. Coursimault, "Large scale url-based classification using online incremental learning," in *Proceedings of the 11th International Conference on Machine Learning and Applications (ICMLA)*, 2012, pp. 402–409.

[29] SSG@Aalto, "Off-the-Hook usability study questionnaires." [Online]. Available: https://ssg.aalto.fi/projects/phishing/usability-study.html

[30] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD*, 2009, pp. 1245–1254.

[31] T.-C. Chen, T. Stepan, S. Dick, and J. Miller, "An anti-phishing system employing diffused information," *ACM Transactions on Information and System Security*, vol. 16, no. 4, pp. 16:1–16:31, 2014.

[32] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "Cantina+: A feature-rich machine learning framework for detecting phishing web sites," *ACM Trans. Inf. Syst. Sec.*, vol. 14, no. 2, pp. 21:1–21:28, 2011.

[33] S. Marchal, J. Francois, R. State, and T. Engel, "Phishstorm: Detecting phishing with streaming analytics," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458–471, 2014.

[34] F. Kausar, B. Al-Otaibi, A. Al-Qadi, and N. Al-Dossari, "Hybrid client side phishing websites detection approach," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 7, pp. 132–140, 2014.

[35] R. S. Rao and S. T. Ali, "Phishshield: A desktop application to detect phishing webpages through heuristic approach," *Procedia Computer Science*, vol. 54, pp. 147 – 156, 2015.

[36] A. K. Jain and B. B. Gupta, "A novel approach to protect against phishing attacks at client side using auto-updated white-list," *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 9, 2016.

[37] G. Varshney, M. Misra, and P. K. Atrey, "A phish detector using lightweight search features," *Computers & Security*, vol. 62, pp. 213 – 228, 2016.

[38] R. M. Mohammad, F. A. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Computing and Applications*, vol. 25, no. 2, pp. 443–458, 2014.

[39] M. N. Feroz and S. Mengel, "Examination of data, rule generation and detection of phishing url using online logistic regression," in *Proceedings of the IEEE Conference on Big Data*, 2014, pp. 241–250.

[40] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks*, 2008, pp. 22:1–22:6.

[41] E. H. Chang, K. L. Chiew, S. N. Sze, and W. K. Tiong, "Phishing detection via identification of website identity," in *International Conference on IT Convergence and Security*, 2013, pp. 1–4.

[42] T. Vissers, W. Joosen, and N. Nikiforakis, "Parking sensors: Analyzing and detecting parked domains," in *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS)*, 2015, pp. 1–14.

[43] Z. Li, S. Alrwais, Y. Xie, F. Yu, and X. Wang, "Finding the linchpins of the dark web: A study on topologically dedicated hosts on malicious web infrastructures," in *IEEE Symposium on Security and Privacy*, 2013, pp. 112–126.

[44] P. De Ryck, N. Nikiforakis, L. Desmet, and W. Joosen, "TabShots: Client-side detection of tabnabbing attacks," in *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '13. ACM, 2013, pp. 447–456.

Samuel Marchal received the engineering degree and M.Sc. degree in computer science in 2011 from TELECOM Nancy, France. He received the Ph.D. degree jointly from the University of Luxembourg, Luxembourg and the University of Lorraine, France, in 2015. He conducted his doctoral research at the Interdisciplinary Centre for Security, Reliability and Trust (SnT) in Luxembourg. He is currently a Post-Doctoral Researcher at Aalto University, Finland, in the Secure Systems Research Group. His interests lie in system security, network security and machine learning.

Giovanni Armano received the B.E. and the M.Eng. degrees in computer engineering from Polytechnic University of Turin, Italy, in 2014 and 2016, respectively. He spent the second year of his M.Sc. studies at Aalto University during which he wrote his M.Sc. Thesis "Real-time Automated Phishing Detection" under the supervision of Prof. N. Asokan and Dr. Samuel Marchal. In December 2016 he joined PortalTech Reply in London as a back-end developer.

Tommi Gröndahl received his MA degree in cognitive science from the University of Helsinki in 2016. He is currently pursuing PhD studies in computer science at Aalto University and in cognitive science at the University of Helsinki. Prior to this, he has worked in Adage Oy as a junior user experience specialist.

Kalle Saari received the M.Sc. and Ph.D. degrees in mathematics and computer science from University of Turku, Finland, in 2003 and 2008, respectively. He is currently a M.Sc. student in machine learning at Aalto University, Finland. He is working on his M.Sc. Thesis in the Secure Systems Research Group under the supervision of Prof. N. Asokan. His interests include machine learning, data mining and data science.

Nidhi Singh received her Ph.D. in computer science from the International Institute of Information Technology - Bangalore, India. She is presently research lead at McAfee Gmbh, Germany, and before that she worked in Xerox Research Center Europe and IBM Labs for several years. Her research interests include application of machine learning algorithms for large-scale text classification, real-time anomaly detection, and energy-aware computing ("green IT").

N. Asokan is a Professor of Computer Science at Aalto University where he co-leads the secure systems research group and directs Helsinki-Aalto Center for Information Security – HAIC. Before joining academia, he spent 17 years in industry research labs with IBM Research and Nokia Research Center. Asokan received his formal education from University of Waterloo, Syracuse University, and Indian Institute of Technology, Kharagpur. More information on his research at http://asokan.org/asokan/.