Phishing Detection: A Literature Survey

Article *in* IEEE Communications Surveys & Tutorials · April 2013

DOI: 10.1109/SURV.2013.032213.00009

CITATIONS
134

READS
6,837

3 authors:

Mahmoud Khonji

13 PUBLICATIONS **306** CITATIONS

SEE PROFILE

Youssef Iraqi

Khalifa University

124 PUBLICATIONS **1,608** CITATIONS

SEE PROFILE

Andy Jones

University of Hertfordshire

185 PUBLICATIONS **854** CITATIONS

SEE PROFILE

Phishing Detection: A Literature Survey

Mahmoud Khonji, Youssef Iraqi, *Senior Member, IEEE,* and Andrew Jones

***Abstract*—This article surveys the literature on the detection of phishing attacks. Phishing attacks target vulnerabilities that exist in systems due to the human factor. Many cyber attacks are spread via mechanisms that exploit weaknesses found in end-users, which makes users the weakest element in the security chain. The phishing problem is broad and no single silver-bullet solution exists to mitigate all the vulnerabilities effectively, thus multiple techniques are often implemented to mitigate specific attacks. This paper aims at surveying many of the recently proposed phishing mitigation techniques. A high-level overview of various categories of phishing mitigation techniques is also presented, such as: detection, offensive defense, correction, and prevention, which we belief is critical to present where the phishing detection techniques fit in the overall mitigation process.**

***Index Terms*—phishing; social engineering; phishing detection; security; email classification;**

I. Introduction

Phishing is a social engineering attack that aims at exploiting the weakness found in system processes as caused by system users. For example, a system can be technically secure enough against password theft, however unaware end users may leak their passwords if an attacker asked them to update their passwords via a given Hypertext Transfer Protocol (HTTP) link, which ultimately threatens the overall security of the system.

Moreover, technical vulnerabilities (e.g. Domain Name System (DNS) cache poisoning) can be used by attackers to construct far more persuading socially-engineered messages (i.e. use of legitimate, but spoofed, domain names can be far more persuading than using different domain names). This makes phishing attacks a layered problem, and an effective mitigation would require addressing issues at the technical and human layers.

Since phishing attacks aim at exploiting weaknesses found in humans (i.e. system end-users), it is difficult to mitigate them. For example, as evaluated in [1], end-users failed to detect 29% of phishing attacks even when trained with the best performing user awareness program. On the other hand, software phishing detection techniques are evaluated against bulk phishing attacks, which makes their performance practically unknown with regards to targeted forms of phishing attacks. These limitations in phishing mitigation techniques have practically resulted in security breaches against several organizations including leading information security providers [2], [3].

Due to the broad nature of the phishing problem, this phishing detection survey begins by:

- Defining the phishing problem. It is important to note that the phishing definition in the literature is not consistent, and thus a comparison of a number of definitions is presented.
- Categorizing anti-phishing solutions from the perspective of phishing campaign life-cycle. This presents the various anti-phishing solution categories such as *detection*. It is important to view the overall anti-phishing picture from a high-level perspective before diving into a particular technique, namely: phishing detection techniques (which is the scope of this survey).
- Presenting evaluation metrics that are commonly used in the phishing domain to evaluate the performance of phishing detection techniques. This facilitates the comparison between the various phishing detection techniques.
- Presenting a literature survey of anti-phishing detection techniques, which incorporates software detection techniques as well as user-awareness techniques that enhance the detection process of phishing attacks.
- Presenting a comparison of the various proposed phishing detection techniques in the literature.

This survey begins by defining the phishing problem in Section II, presenting background and related works in Section III, and an overview of phishing mitigation approaches in Section IV which also presents the taxonomy of various mitigation techniques, such as: detection, prevention, and correction techniques. Subsequent sections in this survey will then focus on phishing detection techniques, which include detection techniques through user awareness, as presented in Section VI, and a number of software techniques. The software-based detection techniques are: blacklists in Section VII, heuristic detection techniques in Section VIII, visual similarity detection techniques in Section IX, and data mining detection techniques in Section X. The evaluations of the surveyed detection techniques are presented in Section XII, followed by the lessons that we have learned in Section XIII. The conclusion is drawn in Section XIV.

II. Definition

The definition of phishing attacks is not consistent in the literature, which is due to the fact that the phishing problem is broad and incorporates varying scenarios. For example, according to PhishTank[1]:

> "Phishing is a fraudulent attempt, usually made through email, to steal your personal information"

PhishTank's definition holds true in a number of scenarios which, roughly, cover the majority of phishing attacks (although no accurate studies have been made to reliably quantify

M. Khonji, Y. Iraqi, and A. Jones are with Khalifa University, UAE. Email: {mkhonji, youssef.iraqi, andrew.jones}@ku.ac.ae

[1]A community-driven project that facilitates individuals to submit, verify, track and share phishing URLs. The definition of phishing attacks by PhishTank is available in http://www.phishtank.com/what_is_phishing.php.

this). However, the definition limits phishing attacks to stealing personal information, which is not always the case.

For example, a socially engineered message can lure the victim to install a Man in the Browser (MITB) malware (e.g. in forms of web browser ActiveX components, plugins or email attachments) which would in turn transfer money to the attacker's bank account, whenever the victim logs in to perform his/her banking tasks, without the need to steal the victim's personal information. Thus we consider that PhishTank's definition is not broad enough to encompass the whole phishing problem.

Another definition is provided by Colin Whittaker et. al. [4]:

> "We define a phishing page as any web page that, without permission, alleges to act on behalf of a third party with the intention of confusing viewers into performing an action with which the viewer would only trust a true agent of the third party"

The definition by Colin Whittaker et. al. aims to be broader than PhishTank's definition in a sense that attackers goals are no longer restricted to stealing personal information from victims. On the other hand, the definition still restricts phishing attacks to ones that act on behalf of third parties, which is not always true.

For example phishing attacks may communicate socially engineered messages to lure victims into installing MITB malware by attracting the victims to websites that are supposed to deliver safe content (e.g. video streaming). Once the malware (or crimeware as often named by Anti-Phishing Working Group (APWG)[2]) is installed, it may log the victim's keystrokes to steal their passwords. Note that the attacker in this scenario did not claim the identity of any third party in the phishing process, but merely communicated messages with links (or attachments) to lure victims to view videos or multimedia content.

In order to address the limitations of the previous definitions above, we consider phishing attacks as semantic attacks which use electronic communication channels (such as E-Mails, HTTP, SMS, VoIP, etc...) to communicate socially engineered messages to persuade victims to perform certain actions (without restricting the actions) for an attacker's benefit (without restricting the benefits). See Definition 1.

Definition 1: Phishing is a type of computer attack that communicates socially engineered messages to humans via electronic communication channels in order to persuade them to perform certain actions for the attacker's benefit.

For example, the performed action (which the attacker persuades the victim to perform it) for a PayPal user is submitting his/her login credentials to a fake website that looks similar to PayPal. As a perquisite, this also implies that the attack should create a need for the end-user to perform such action, such as informing him that his/her account would be suspended unless he logs in to update certain pieces of information [5].

[2]A non-profit global pan-industrial and law enforcement association focused on eliminating the fraud, crime and identity theft that result from phishing, pharming, malware and email spoofing.

III. Background

A. History

According to APWG, the term *phishing* was coined in 1996 due to social engineering attacks against America On-line (AOL) accounts by online scammers.

The term *phishing* comes from *fishing* in a sense that fishers (i.e. attackers) use a bait (i.e. socially-engineered messages) to fish (e.g. steal personal information of victims). However, it should be noted that the theft of personal information is mentioned here as an example, and that attackers are not restricted by that as previously defined in Section II.

The origins of the *ph* replacement of the character *f* in *fishing* is due to the fact that one of the earliest forms of hacking was against telephone networks, which was named *Phone Phreaking*. As a result, *ph* became a common hacking character replacement of *f*.

According to APWG, stolen accounts via phishing attacks were also used as a currency between hackers by 1997 to trade hacking software in exchange of the stolen accounts.

Phishing attacks were historically started by stealing AOL accounts, and over the years moved into attacking more profitable targets, such as on-line banking and e-commerce services.

Currently, phishing attacks do not only target system end-users, but also technical employees at service providers, and may deploy sophisticated techniques such as MITB attacks.

B. Phishing Motives

According to Weider D. et. al. [6], the primary motives behind phishing attacks, from an attacker's perspective, are:

- *Financial gain:* phishers can use stolen banking credentials to their financial benefits.
- *Identity hiding:* instead of using stolen identities directly, phishers might sell the identities to others whom might be criminals seeking ways to hide their identities and activities (e.g. purchase of goods).
- *Fame and notoriety:* phishers might attack victims for the sake of peer recognition.

C. Importance

According to APWG, phishing attacks were in a raise till August, 2009 when the all-time high of 40,621 unique[3] phishing reports were submitted to APWG. The total number of submitted unique phishing websites that were associated with the 40,621 submitted reports in August, 2009 was 56,362.

As justified by APWG, the drop in phishing campaign reports in the years 2010 and 2011 compared to that of the year 2009 was due to the disappearance of the Avalanche gang[4] which, according to APWG's 2^{nd} half of 2010 report, was responsible for 66.6% of world-wide phishing attacks in the 2^{nd} half of 2009 [7]. In the 1^{st} half of the year 2011, the total number of submitted phishing reports to APWG

[3]A single phishing campaign might be reported multiple times, thus it is important to not consider the redundant reports.

[4]A phishing gang that is held responsible for many phishing campaigns as reported by APWG.

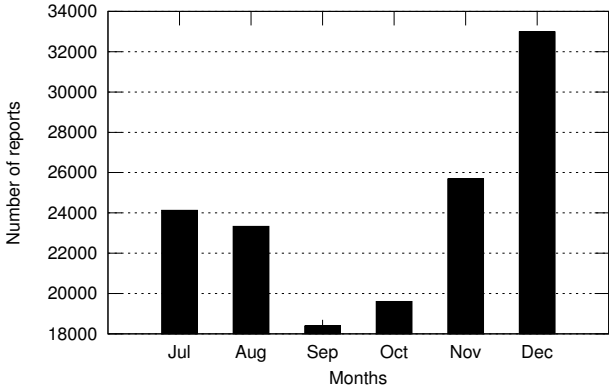

Fig. 1. Total number of submitted unique phishing reports in the second half of 2011 to APWG. Source: [9]

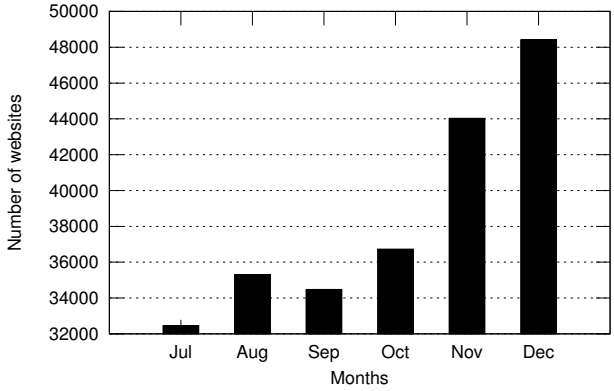

Fig. 2. Total number of known unique phishing websites in the second half of 2011 according to APWG. Source: [9]

was 26,402, which is 35% lower than that of the peak in the year 2009 [8]. However, according to APWG, the drop in phishing attacks was due to the switch in the activities of the Avalanche gang from traditional phishing campaigns into malware-based phishing campaigns. In other words, the Avalanche gang did not stop phishing campaigns but rather switched their tactics toward malware-based phishing attacks (which still requires electronic communication channels and social engineering techniques to deliver malware).

Among the various types of malware that are used in phishing attacks, Trojan horses software seem to be in a raise, and are the most popular type of malware deployed by phishing attacks. According to APWG, Trojans software contributed 72% of the total malware detected in the 1^{st} half of 2011, from the previous value of 55% in the 2^{nd} half of 2010.

It is also important to note that although the number of phishing attack reports dropped since the peak in 2009, the number of phishing attack reports are still high ,compared to that of the 2^{nd} half of 2008 which faced an average of 28,916 unique reports, and ranged between 22,000 and 26,000 of unique reports each month in the $1st$ half of 2011.

On the other hand, the 2^{nd} half of 2011 saw a raise in phishing reports and websites, which seems to be correlated with holidays season [9] as depicted in Figures 1 and 2. Which is further amplified when knowing that each phishing campaign can be sent to thousands or even millions of users via electronic communication channels.

The year 2011 saw a number of notable spear phishing attacks against well known security firms such as RSA [10] and HB Gary [2], which resulted in further hacks against their clients such as RSA's client Lockheed Martin [3]. This shows that the dangers of phishing attacks, or security vulnerabilities due to the human factor, are not limited to the naivety of end-users since technical engineers can also be victims.

Minimizing the impact of phishing attacks is extremely important and adds great value to the overall security of an organization.

D. Challenges

Because the phishing problem takes advantage of human ignorance or naivety with regards to their interaction with electronic communication channels (e.g. E-Mail, HTTP, etc…), it is not an easy problem to permanently solve. All of the proposed solutions attempt to minimize the impact of phishing attacks.

From a high-level perspective, there are generally two commonly suggested solutions to mitigate phishing attacks:

- User education; the human is educated in an attempt to enhance his/her classification accuracy to correctly identify phishing messages, and then apply proper actions on the correctly classified phishing messages, such as reporting attacks to system administrators.
- Software enhancement; the software is improved to better classify phishing messages on behalf of the human, or provide information in a more obvious way so that the human would have less chance to ignore it.

The challenges with both of the approaches are:

- Non-technical people resist learning, and if they learn they do not retain their knowledge permanently, and thus training should be made continuous. Although some researchers agree that user education is helpful [1], [11], [12], a number of other researchers disagree [13], [14]. Stefan Gorling [13] says that:

  > "this is not only a question of knowledge, but of utilizing this knowledge to regulate behavior. And that the regulation of behavior is dependent on many more aspects other than simply the amount of education we have given to the user"

- Some software solutions, such as authentication and security warnings, are still dependent on user behavior. If users ignore security warnings, the solution can be rendered useless.
- Phishing is a semantic attack that uses electronic communication channels to deliver content with natural languages (e.g. Arabic, English, French, etc…) to persuade victims to perform certain actions. The challenge here is that computers have extreme difficulty in accurately understanding the semantics of natural languages. A notable attempt is E-mail-Based Intrusion Detection System (EBIDS) [15], which uses Natural Language Processing (NLP) techniques to detect phishing attacks, however its performance evaluation showed a phishing detection rate

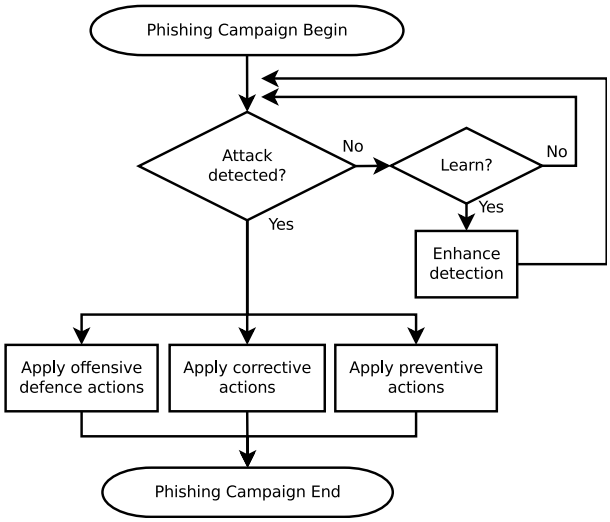

Fig. 3. The life-cycle of phishing campaigns from the perspective of anti-phishing techniques.

of only 75%. In our opinion, this justifies why most well-performing phishing classifiers do not rely on NLP techniques.

IV. Mitigation of Phishing Attacks: An Overview

Due to the broad nature of the phishing problem, we find important to visualize the life-cycle of the phishing attacks, and based on that categorize anti-phishing solutions.

Based on our review of the literature, we depict a flowchart describing the life-cycle of phishing campaigns from the perspective of anti-phishing techniques, which is intended to be the most comprehensive phishing solutions flowchart. See Figure 3.

When a phishing campaign is started (e.g. by sending phishing emails to users), the first protection line is detecting the campaign. The detection techniques are broad and could incorporate techniques used by service providers to detect the attacks, end-user client software classification, and user awareness programs. More details are in Section IV-A.

The ability to detect phishing campaigns can be enhanced whenever a phishing campaign is detected by learning from such experience. For example, by *learning* from previous phishing campaigns, it is possible to *enhance the detection* of future phishing campaigns. Such learning can be performed by a human observer, or software (i.e. via a machine learning algorithm).

Once the phishing attack is detected, a number of actions could be applied against the campaign. According to our review of the literature, the following categories of approaches exist:

- Offensive defense — these approaches aim to attack phishing campaigns to render them less effective. This approach is particularly useful to protect users that have submitted their personal details to attackers. More details are in Section IV-B.
- Correction — correction approaches mainly focus on taking down the phishing campaign. In case of phishing websites, this is achieved by suspending the hosting account or removing phishing files. More details are in Section IV-C.
- Prevention — phishing prevention methods are defined differently in the literature depending on the context. In this survey, the context is attempting to prevent attackers from starting phishing campaigns in the future. More details are in Section IV-D.

However, if the phishing campaign is not detected (let it be detected by a human or a software classifier), then none of these actions can be applied. This emphasizes on the importance of the detection phase.

A. Detection Approaches

In this survey, we consider any anti-phishing solution that aims to identify or classify phishing attacks as detection solutions. This includes:

- User training approaches — end-users can be educated to better understand the nature of phishing attacks, which ultimately leads them into correctly identifying phishing and non-phishing messages. This is contrary to the categorization in [16] where user training was considered a preventative approach. However, user training approaches aim at enhancing the ability of end-users to detect phishing attacks, and thus we categorize them under "detection". Further discussions on the human factor are presented in Section VI.
- Software classification approaches — these mitigation approaches aim at classifying phishing and legitimate messages on behalf of the user in an attempt to bridge the gap that is left due to the human error or ignorance. This is an important gap to bridge as user-training is more expensive than automated software classifiers, and user-training may not be feasible in some scenarios (such as when the user base is huge, e.g. PayPal, eBay, etc...). Further discussions on software classification approaches are presented in Sections VII, VIII, IX and X.

The performance of detection approaches can be enhanced during the learning phase of a classifier (whether the classifier is human or software). In the case of end-users, their classification ability can be enhanced by improving their knowledge of phishing attacks by learning individually through their online experience, or by external training programs. In the case of software classifiers, this can be achieved during the learning phase of a Machine Learning-based classifier, or the enhancement of detection rules in a rule-based system.

Detection techniques not only help in *directly* protecting end-users from falling victims to phishing campaigns, but can also help in enhancing phishing honeypots[5] to isolate phishing spam from non-phishing spam.

It is also important to note that the detection of phishing attacks is the starting point of the mitigation of phishing attacks. As depicted in Figure 3, if a phishing campaign is not detected, none of the other mitigation approaches can be

[5]Monitored network decoys that can be used to distract attackers from attacking valuable resources, provide early warnings about new attack trends, or enable in-depth analysis of the performed attacks.

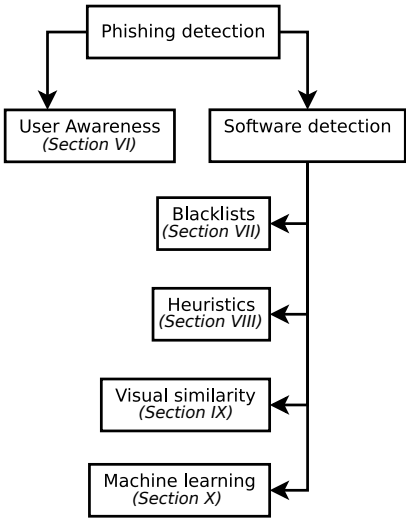

Fig. 4. An overview of phishing detection approaches.

applicable. For example, all of the mitigation techniques, such as *correction*, *prevention* and *offensive defense* depend on a functional and accurate *detection* phase.

The primary focus of this survey is covering the detection phase of phishing attacks. Figure 4 depicts an overview of phishing detection techniques that are covered in the subsequent sections of this survey.

B. Offensive Defense Approaches

Offensive defense solutions aim to render phishing campaigns useless for the attackers by disrupting the phishing campaigns. This is often achieved by flooding phishing websites with fake credentials so that the attacker would have a difficult time to find the real credentials.

Two notable examples are:

- BogusBiter [17] — A browser toolbar that submits fake information in HTML forms whenever a phishing website is encountered. According to BogusBiter, the detection of phishing websites is done by other tools. In other words, instead of simply showing a warning message to the end-user whenever a phishing website is visited, BogusBiter also submits fake data into HTML forms of the visited phishing website. Submitting fake data into the HTML forms is intended to disrupt the corresponding phishing campaigns, with the hope that such fake data may make the attackers task of finding correct data (among the fake data) more difficult. This is an attempt to save the stolen credentials of other users that have been captured by the phishing campaign by contaminating the captured results with bogus data. However, the limitations are:
  - Toolbars need to be installed on a wide enough user base to render this effective.
  - If the user base is wide enough, BogusBiter may cause Denial of Service (DOS) floods against servers that host legitimate shared hosted websites as well, simply because one of the shared web-hosts may have a phishing content.
  - Increased bandwidth demand.
  - Non-standard HTML forms are not detected by BogusBiter.
  - The empirical effectiveness of this solution is not accurately measured.
- Humboldt [18] — Similar to BogusBiter, except that BogusBiter relies on submissions from end-user clients, while Humboldt relies on distributed and dedicated clients over the Internet instead of end-user toolbars that may visit phishing sites, in addition to a mechanism to avoid causing DOS floods against servers. This can make Humboldt more effective against phishing websites due to the more frequent submission of data to phishing pages. The limitations are:
  - Increased bandwidth demand.
  - Non-standard HTML forms are not detected by Humboldt.
  - The empirical effectiveness of this solution is not accurately measured.

Although offensive defense approaches can theoretically make the attackers task more difficult in finding a victim's personal information, it is not known how difficult it really becomes. For example, a phisher might simply set up a script to test the credentials in a loop, and by using anonymous web surfing techniques attackers sessions will be difficult to track by the target web server. In other words, the actual returned security value of offensive defense approaches are not accurately evaluated and can be questioned.

C. Correction Approaches

Once a phishing campaign is detected, the correction process can begin. In the case of phishing attacks, correction is the act of taking the phishing resources down. This is often achieved by reporting attacks to Service Providers.

Phishing campaigns often rely on resources, such as:

- Websites — could be a shared web host owned by the phisher, a legitimate website with phishing content uploaded to it, or a number of infected end-user workstations in a botnet[6].
- E-mail messages — could be sent from a variety of sources, such as: free E-mail Service Provider (ESP) (e.g. Gmail, Hotmail, etc...), open Simple Mail Transfer Protocol (SMTP) relays or infected end-user machines that are part of a botnet.
- Social Networking services — web 2.0 services, such as Facebook and Twitter, can be used to deliver socially engineered messages to persuade victims to reveal their passwords.
- Public Switched Telephone Network (PSTN) and Voice over IP (VoIP) — similar to other forms of phishing attacks, attackers attempt to persuade victims to perform actions. However, the difference is that attackers attempt to exploit spoken dialogues in order to collect data (as opposed to clicking on links). Moreover, due to the way VoIP protocols (e.g. Session Initiation Protocol (SIP))

[6]A botnet is a number of infected computers controlled by attackers for malicious use.

function, and the way many VoIP provider systems are configured, spoofing Caller IDs are used by attackers as tools to increase their persuasion [19].

In order to correct such behavior, responsible parties (e.g. service providers) attempt to take the resources down. For example:

- Removal of phishing content from websites, or suspension of hosting services.
- Suspension of email accounts, SMTP relays, VoIP services
- Trace back and shutdown of botnets.

This also extends to the shutdown of firms that frequently provide services to phishing attackers.

The shutdown process can be initiated by organizations that provide brand protection services to their clients, which may include banking and financial companies that are possible victims of phishing attacks. When phishing campaigns are identified, they can be reported to their hosting Internet and web hosting service providers for immediate shutdown. Depending on the country where phishers and phishing campaigns exist, the penalties and procedures can differ.

D. Prevention Approaches

The "prevention" of phishing attacks can be confusing, as it can mean different things depending on its context:

- Prevention of users from falling victim — in this case, phishing detection techniques will also be considered prevention techniques. However, this is not the context we refer to when "prevention" is mentioned in this survey.
- Prevention of attackers from starting phishing campaigns — in this case, law suits and penalties against attackers by Law Enforcement Agencies (LEAs) are considered as prevention techniques.

In this survey, whenever the keyword "prevention" is mentioned, it refers to the second previous item which is minimizing the possibility of attackers starting phishing campaigns via LEA.

Usually, LEA may take a number of weeks to complete their investigation and response procedures. Thus, it is common to apply prevention techniques after all other mitigation techniques, which is due to the expensive nature of LEA investigations that makes them consume a relatively large period of time.

Once the sources of the phishing attacks are traced, LEA can then file law suits which in turn may issue penalties such as: imprisonment, fines and forfeiture of equipments used to convey the attacks.

V. Evaluation Metrics

Since the subsequent sections in this survey will compare a number of detection techniques, we find it useful to introduce the evaluation metrics used in the phishing literature.

In any binary classification problem, where the goal is to detect phishing instances in a dataset with a mixture of phishing and legitimate instances, only four classification possibilities exist. See the confusion matrix presented in Table I for details, where $N_{P\to P}$ is the number of *phishing* instances that are correctly classified as *phishing*, $N_{L\to P}$ is the number of *legitimate* instances that are incorrectly classified as *phishing*, $N_{P\to L}$ is the number of *phishing* instances that are incorrectly classified as *legitimate*, and $N_{L\to L}$ is the number of *legitimate* instances that are correctly classified as *legitimate*.

TABLE I
Classification confusion matrix

	Classified as phishing	**Classified as legitimate**
Is phishing	$N_{P\to P}$	$N_{P\to L}$
Is legitimate	$N_{L\to P}$	$N_{L\to L}$

Based on our review of the literature, the following are the most commonly used evaluation metrics:

- True Positive (TP) rate — measures the rate of correctly detected phishing attacks in relation to all existing phishing attacks. See Equation (1) for details.
- False Positive (FP) rate — measures the rate of legitimate instances that are incorrectly detected as phishing attacks in relation to all existing legitimate instances. See Equation (2) for details.
- True Negative (TN) rate — measures the rate of correctly detected legitimate instances in relation to all existing legitimate instances. See Equation (3) for details.
- False Negative (FN) rate — measures the rate of phishing attacks that are incorrectly detected as legitimate in relation to all existing phishing attacks. See Equation (4) for details.
- Precision (P) — measures the rate of correctly detected phishing attacks in relation to all instances that were detected as phishing. See Equation (5) for details.
- Recall (R) — equivalent to TP. See Equation (6) for details.
- f_1 score — Is the harmonic mean between P and R. See Equation (7) for details.
- Accuracy (ACC) — measures the overall rate of correctly detected phishing and legitimate instances in relation to all instances. See Equation (8) for details.
- Weighted Error (W_{Err}) — measures the overall weighted rate of incorrectly detected phishing and legitimate instances in relation to all instances. See Equation (9) for details.

$$TP = \frac{N_{P\to P}}{N_{P\to P} + N_{P\to L}} \tag{1}$$

$$FP = \frac{N_{L\to P}}{N_{L\to L} + N_{L\to P}} \tag{2}$$

$$TN = \frac{N_{L\to L}}{N_{L\to L} + N_{L\to P}} \tag{3}$$

$$FN = \frac{N_{P\to L}}{N_{P\to P} + N_{P\to L}} \tag{4}$$

$$P = \frac{N_{P\to P}}{N_{L\to P} + N_{P\to P}} \tag{5}$$

$$R = TP \tag{6}$$

$$f_1 = \frac{2PR}{P+R} \tag{7}$$

$$ACC = \frac{N_{L\to L} + N_{P\to P}}{N_{L\to L} + N_{L\to P} + N_{P\to L} + N_{P\to P}} \tag{8}$$

$$W_{Err} = 1 - \frac{\lambda \cdot N_{L\to L} + N_{P\to P}}{\lambda \cdot N_{L\to L} + \lambda \cdot N_{L\to P} + N_{P\to L} + N_{P\to P}} \tag{9}$$

where λ weights the importance of legitimate instances as used previously in [16]. For example, if $\lambda = 9$, then W_{Err} penalizes the misclassification of legitimate instances 9 times more than the misclassification of phishing instances.

VI. Detection of Phishing Attacks: The Human Factor

Since phishing attacks attempt to take advantage of the inexperienced users, an obvious solution is educating the users, which would in turn reduce their susceptibility to falling victims of phishing attacks. A number of user training approaches have been proposed throughout the past years.

The human factor is broad. Simply educating end-users alone does not necessarily regulate their behavior [13]. This section will present and discuss some of the work contributed in the field of user training in relation to phishing attacks.

A. Phishing Victims

Julie S. Downs et al. [20] surveyed 232 computer users to study what are the criteria that can predict the susceptibility of a user to fall victims for phishing emails. The survey was formed in a role play where each user was expected to analyze emails as well as answering a number of questions. The outcome of the study was that those who had a good knowledge about the definition of "phishing" were significantly less likely to fall for phishing emails, while knowledge about other areas, such as cookies, spyware and viruses did not help in reducing vulnerability to phishing emails. Interestingly, the survey showed that knowledge about negative consequences (e.g. credit card theft) did not help in reducing vulnerability to phishing emails. The study concluded that user educational messages should focus on educating users about phishing attacks rather than warning them about the dangers of negative consequences.

Another study that confirms the study in [20] was made by Huajun Huang et. al. [21], which concluded that the primary reasons that lead technology users to fall as victims for phishing attacks are:

- Users ignore passive warnings (e.g. toolbar indicators).
- A large number of users cannot differentiate between phishing and legitimate sites, even if they are told that their ability is being tested.

A demographic study made by Steve Shen et. al. [1] shows a number of indirect characteristics that correlate between victims and their susceptibility to phishing attacks. According to their study, gender and age strongly correlate with phishing susceptibility. They conclude that:

- Females tend to click on email links more often than males.
- People between 18 and 25 years old were much more likely to fall victim to phishing attacks than other age groups.

This was justified to be caused by a lack of sufficient technical knowledge and experience, which further confirms [20] and [21].

B. User-Phishing Interaction Model

Xun Don et. al [5] described the first visual user-phishing interaction model. The model describes user interaction from the decision making point of view; which starts the moment a user sees phishing content, and ends when all user actions are completed (see Figure 5). The goal is assisting the process of mitigating phishing attacks by firstly understanding how users interact with phishing content.

Inputs to the decision making process are:

- External information: could be anything learned through the User Interface (UI) (Web/mail client and their content), or expert advice. The phisher only has control over what is presented by the UI. Usually, the user does not ask for expert advice unless he is in doubt (i.e. if a user is convinced that a phishing site is legitimate, he might not ask for expert advice in the first place).
- Knowledge and context: the user's current understanding of the world, which is built over time (e.g. news, past experience).
- Expectation: users have expectations based on their understanding and the outcome of their actions.

During the decision making process, two types of decisions can be made, which are:

- Planning a series of actions to be taken.
- Deciding on the next action in sequence to be taken. This is influenced by the outcome resulting from the previous action.

The first action is often done consciously, while the subsequent actions are done sub-consciously. As a result, the outcome of the first action affects user's ability in detecting phishing attacks in subsequent actions.

A phishing example that takes advantage of the above behavior is, an email that presents a non-existent Uniform Resource Locator (URL) pointing to a legitimate website, followed by a backup URL pointing to a phishing site. The first action a victim could take is clicking on the primary legitimate URL to view (say) an e-card (as claimed by the phisher), which obviously would result in a *404 page not found* error. The second action would be clicking the backup link pointing to a phishing site. Since the phishing site was visited as a second action by the victim user, he would have a lower probability to detect inconsistencies in the backup URL.

Each of the two types of decisions mentioned above, follow the following steps:

- Construction of perception: constructed through the context where the user reads (say) an email message. Such

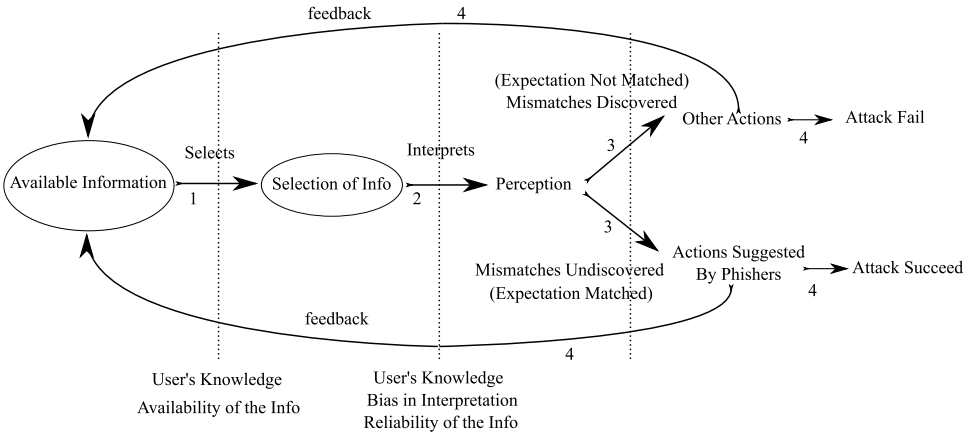

Fig. 5. Overview of the interaction between end-users and phishing content. Source: [5].

as, senders/recipients, conversation cause, or suggested actions by the email. In legitimate messages, there are no inconsistencies between the reality and message claims (e.g. senders are the real senders whom they claim to be, and suggested actions by email content does what it says). However, in phishing messages there are inconsistencies (e.g. if the sender's ID is spoofed, or the message's content claims to fix a problem while attempting, in reality, to obtain personal information). If the end-user discovers inconsistencies in a given phishing message, the phishing attack would then fail to persuade the end-user.

- Generation of possible solutions: users usually find solutions through available resources. However, with phishing emails, the user is not requested to generate a possible solution in the first place, as the phisher already suggests a solution to the user. For example, if the phishing email content presents a problem, such as account expiry, it will also present a solution, such as activating the account through logging in a URL from which expiry is prevented.
- Generation of assessment criteria: different users have different criteria that reflects how they view the world, their emotional state, personal preferences, etc.... As the paper claims, most phishing attempts do not take into account such details, but rely on generic common-sense criteria instead; for example: an attacker might place a tick box labeled "Secure login" to meet a security criteria most users require. Phishing attacks aim to match user criteria as much as possible.

As stated earlier, phishers can only modify the decision process of users through providing *external information* through the UI. The user interface provides two data sets (see Figure 6):

- Meta data: such as URL presented in web browser address bars, or email addresses.
- Content data: such as site or email content.

Phishing attacks succeed if a phishing attack convinces the user that both *meta data* and *content data* are legitimate.

Users may use *meta data* to decide whether an email message is legitimate. Phishers may also spoof *meta data* in order to further trick the users. As stated in [5], the solution to *meta-data* integrity problem is *not* through user education or awareness as it is very difficult for users to validate whether the source IP address is legitimate in case the domain name was spoofed. Users should not be expected to validate the *meta-data* as it is rather a system design or implementation problem.

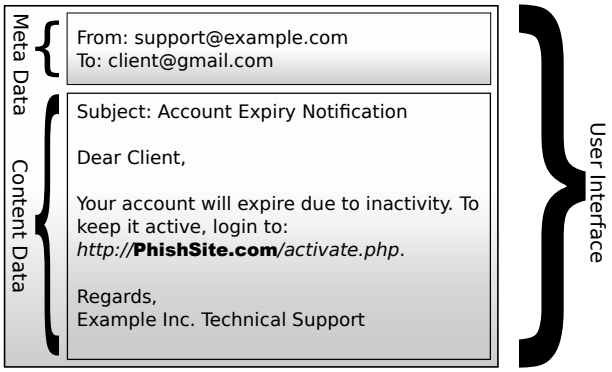

Fig. 6. User Interface (UI) components..

On the other hand, through social engineering, phishers create convincing and legitimate-looking *content data*. A common solution to this is user awareness.

C. Service Policies

Although different methods have been developed to deliver warnings and notifications to end-users, *them* having a knowledge of phishing is required to render warnings or notifications useful.

Users should be made aware of various types of social engineering attacks, and a low level of awareness would result in exposing user's credentials irrespective of software or hardware protection layers. A relatively common solution is educating users via periodic messages (Emails, SMS, etc...).

Organizations should have strict policies against the distribution of confidential data over email, Short Message Service (SMS), or VoIP, coupled with user awareness programs (e.g. periodic educational messages) to ensure that users are aware of the policies. Users that are made aware of this have a higher

chance of detecting inconsistencies within a phishing message if the *content data*, for example, asked for confidential information.

Service providers should also strictly enforce their policies against illicit use of their services. Many hosting providers take services down in cases where they are abused. As researched by T. Moore, and R. Clayton in [22], service take-down is increasingly common in the way security problems are handled.

Users should also be aware of the environment that they interact with. e.g. mail and web clients display *meta data*, which users need to use to validate *content data*; e.g. Thunderbird displays notifications if a mail signature is not valid. Ignoring warnings, such as X.509 certificate verification failure messages, could lead users into permanently trusting illicit certificates. If the trusted certificate was caused by a Man in the Middle (MITM) attack, then such behavior could break the trust model of Public Key Infrastructure (PKI), making Hypertext Transfer Protocol Secure (HTTPS) or Secure/Multipurpose Internet Mail Extensions (S/MIME) the same as their insecure counterparts.

User education should be coupled with clear IT policies as well as applied practices. Otherwise, the policy could reverse the educational message. For example, an IT policy that warns about the dangers of HTTPS sites with invalid X.509 certificates, while publishing local sites with self-signed certificates (such as the default setup of MS Outlook Web Access), could lead into an implied message that the security policies and warnings are not of great importance. Ultimately, this leads to the spread of habits opposing the actual intentions of security policies.

Another challenge is the widespread use of computers in environments other than the workplace, where proper IT policies do not exist. Frequent misuse of technology could lead end users into considering ignoring security warnings as the norm, which might gradually be taken into their workplace as well.

The psychological aspect of the phishing problem is broad, and plays a major role on the end user behavior. Due to the generic nature of the challenge, a highly predictable solution remains difficult to achieve, if not impossible by today's scientific advancements in understanding human mental functions and behavior.

D. Passive and Active Warnings

User interfaces can show security warnings based on triggered actions, such as viewing phishing web pages, as commonly deployed by many web browsers. There are generally two ways of presenting the warnings to the end user:

- Passive warnings — the warning does not block the content-area and enables the user to view both the content and the warning as in the snapshot depicted in Figure 7.
- Active warnings — the warning blocks the content-data, which prohibits the user from viewing the content-data while the warning is displayed as in the snapshot depicted in Figure 8.

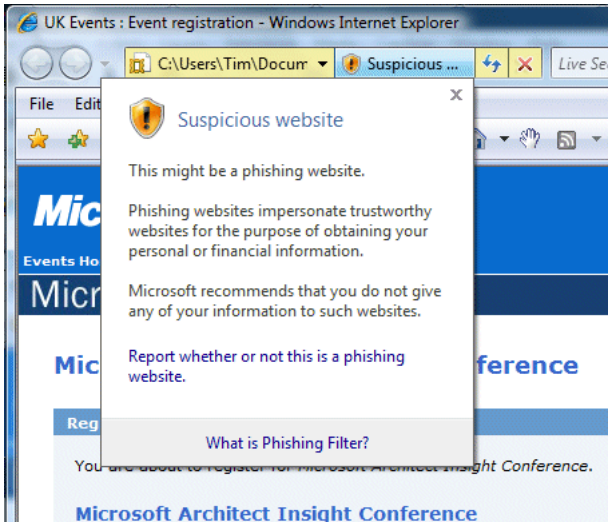

Fig. 7. IE displaying a passive warning page against a suspected unsafe site. Source: http://www.itwriting.com/.

Fig. 8. FireFox displaying an active warning page against a suspected unsafe site.

According to a study conducted by Egelman et. al. [23], passive warnings are ineffective. Only 13% of the participants heeded to passive browser warnings, while 79% of the participants heeded for active warnings.

Similarly, toolbars have mostly followed passive warnings expressed by notification icons, which justifies the failure of security toolbars to properly notify end-users of phishing risks. According to a study conducted by Min Wu et. al [24], users failed to heed toolbar passive warnings.

Both of the studies, [23] and [24], concluded that active warnings are superior to passive ones. Users do not heed warnings unless they are forced to by blocking the content-data portion of the UI to display a clear warning message.

E. Educational Notices

E-services, such as e-banking, often communicate periodic educational messages to warn their clients of potential phishing threats. The messages are often delivered via SMS's and

e-mails.

According to a study conducted by Kumaraguru et. al. [11], periodic security notices are ineffective and, although they may improve the knowledge of end-users, the periodic notices fail to change their behavior.

Alternatively, Kumaraguru et. al. [11] proposes and evaluates the design of an alternative periodic method to send educational notices, that are embedded into the daily activity of the end-users. The study shows that embedded training systems are more effective than periodic security notices (89% of participants who were trained by the periodic notices were victims for testing phishing emails, while 30% of participants who were trained by comics embedded training messages were victims of testing phishing emails). The evaluation in [11] also compared multiple approaches in educating users, such as text, annotated figure and comics. The study concluded that embedded comics were the most effective approach.

The proposed system functions as follows:

- The e-mail administrator prepares a number of fake phishing emails.
- The phishing emails are communicated to the victims. No warnings are shown at this stage.
- Once the victim interacts with a phishing email, such as by clicking on a phishing link, the user is then shown a security warning teaching him the risks of phishing attacks.

The advantage of this embedded training method is that it teaches the end-user in the most receptive moment — which is when he falls as a victim for a fake phishing attack.

Kumaraguru et. al. [11] then concluded with a number of design principles that should be followed to enhance user educational notices:

- Training messages to be embedded into the daily activity of the user, without the need to read external sources (e.g. other websites or SMS).
- The warning message should clearly and concisely explain the causes; warning messages can fail if they have too much of textual data.
- The warning message should clearly and concisely explain proper actions to be taken by the end-user to enhance his/her security.
- The warning should not be delayed, but be shown immediately following the moment when the user falls as victim and clicks on an email link.
- The fake phishing messages used for training purposes should mimic closely phishing messages in the wild.
- Enhancing the security warning text with story-based comics enhances readability.

However, the drawbacks of the proposed embedded training system are:

- The system requires a human administrator to craft the messages. This adds delay and increases the maintenance cost of the solution.
- The crafted phishing emails by the administrator are limited by the administrator's understanding of phishing attacks. If the administrator is unaware of latest trends in phishing emails, his/her crafted phishing attacks might be less effective.

In other words, a highly important parameter to the proposed method is the administrator who is tasked to create phishing-like messages, and the proposed solution's performance is heavily based on the performance of the administrator whom in turn is a variable as not all administrators are equivalent.

VII. Phishing Detection by Blacklists

Blacklists are frequently updated lists of previously detected phishing URLs, Internet Protocol (IP) addresses or keywords. Whitelists, on the other hand, are the opposite, and could be used to reduce FP rates. Blacklists do not provide protection against zero-hour phishing attacks as a site needs to be previously detected first in order to be blacklisted. However, blacklists generally have lower FP rates than heuristics [25].

As studied in [25], blacklists are found to be ineffective against zero-hour phishing attacks, and were able to detect only 20% of them. The study [25] also shows that 47% to 83% of phishing URL were blacklisted after 12 hours. This delay is a significant issue as 63% of phishing campaigns end within the first 2 hours.

A. Google Safe Browsing API

Google Safe Browsing API enables client applications to validate whether a given URL exists in blacklists that are constantly updated by Google [26]. Although the protocol is still experimental, it is used by Google Chrome and Mozilla Firefox. The current implementation of the protocol is provided by Google, and only consists of two blacklists named *goog-phish-shavar* and *goog-malware-shavar*, for phishing and malware respectively. However the protocol itself is agnostic to the list type as well as to the provider of the list.

The API requires client applications to communicate with providers through HTTP while adhering to syntax specified in Protocolv2Spec [27], which is the second version of the protocol; the first version faced scalability and efficiency issues that are also outlined in [27]. Notable changes in the second version are:

- Version 2 of the protocol divides the URL data into chunks and allows partial updates. Such partial updates were not available in version 1 of the protocol.
- Version 2 of the protocol does not *always* send full 256-bit hashes of blacklisted URLs to web browsers, instead it initially sends a list of 32-bit truncated forms of the hashes. However, if a visited URL had a hash such that its 32-bit truncated form matched any of those in the truncated URLs hash list, then the browser will download all complete 256-bit hashes that have the same first 32-bits as that of the suspect URL. This allows saving bandwidth when transmitting the URLs hash blacklist, specially when knowing that most visited URLs are legitimate in the common case (i.e. the no-match:match ratio is high in reality), and that the first 32-bits are still sufficient (for most cases) to yield mismatches between different URLs. In other words, the 256-bit URL hashes are only sent to resolve possible hash collisions with

regards to the first 32-bits of the hashes (i.e. to avoid false positives due to hash collisions). Version 1 of the protocol did not support such use of truncated hashes, and always sent the full 256-bits (which resulted in inefficient use of bandwidth). More details are presented in the *background* section in Protocolv2Spec [27].

The client application downloads lists from providers, and maintains its freshness by incremental updates that contain additive and subtractive chunks. The protocol follows a pull model that requires the client to connect to the server at certain integer time intervals measured in minutes (t_m). To reduce the server load as the number of clients increases, the following mechanism is followed by the client applications:

1) The first update occurs after 0 to 5 minutes following client start. $t_m \leftarrow rand(0, 5)$.
2) t_m would then be set explicitly by the server for subsequent updates. Otherwise, the client would assume $t_m \leftarrow rand(15, 45)$.

In case of error or timeout:

1) $t_m \leftarrow 1$.
2) If an error or timeout was returned, then $t_m \leftarrow 30(rand(0, 1) + 1)$.
3) If an error or timeout was also returned, then $t_m \leftarrow 2t_m$.
4) Step 3 is repeated until $t_m = 480$. The client then fixes the interval at 480 minutes until the server responds.

Since Version 2 of the protocol distributes the first 32-bit of every SHA-256 hash, the client has to apply Algorithm 1 in order to handle possible hash collisions with regards to the first 32-bit segments of the hashes.

Algorithm 1 Protocolv2Spec phishing detection in pseudo-code

```
1: H_f ← sha256(URL)
2: H_t ← truncate(H_f, 32)
3: if H_t ∈ H_l then
4:     for H_r ← queryProvider(H_t) do
5:         if H_f = H_r then
6:             warnPhishing(URL)
7:         end if
8:     end for
9: end if
```

where H_f is a full SHA-256 hash of the candidate URL as calculated locally by the browser, H_t is a truncated first 32-bit of H_f, H_l is a truncated local hash list, and H_r is the full SHA-256 hashes set as returned from the remote server.

In other words, the client software has a downloaded list of truncated hashes of phishing URL. The hash of suspect URLs are calculated, truncated and then compared against the locally available blacklist. In a case where a match is found, the browser then sends a query to the blacklist provider (Google in this case) to retrieve full hash entries that match the truncated hash. If the non-truncated hash of the suspect URL is found in the returned list, the URL is classified as phishing, or legitimate if otherwise.

Based on our understanding of the protocol's specifications, the use of truncated hashes is mainly to reduce download size, which eventually leads into faster distribution of blacklists into client software.

B. DNS-Based Blacklist

DNS-based Blacklist (DNSBL) providers use the standard DNS protocol. Due to its use of the standard DNS specification, any standard compliant DNS server could act as a DNSBL. However, since the number of listed entries is large, a server that is not optimized for handling large amounts of DNS A or TXT Resource Records (RRs) may face performance and resource strains. *rbldnsd*[7] is a fast DNS server, designed specifically to handle large RR suited for DNSBL duties.

When a Message Transfer Agent (MTA) establishes an inbound SMTP connection, it can verify whether the connecting source is listed in phishing blacklists[8], by sending a DNS A RR query to a DNSBL server on the connecting IP address. For example, if the source IP address of the SMTP connection is 86.96.226.20, and assuming that the DNSBL server is dnsbl-2.uceprotect.net, then a DNS A RR query should be sent for 20.226.96.86.dnsbl-2.uceprotect.net to a DNS server; which is the reverse form of the suspect IP address appended by the DNSBL domain name. If the DNSBL server found an entry, then it would mean that the IP address is blacklisted.

Listing 1 presents the output of a DNS A RR query to a DNSBL server with the program `host` executed in a Unix machine. The output indicates that there exists a DNS A RR entry, which means that the respective IP address (i.e. 86.96.226.20) is blacklisted.

Listing 1. A RR Response from a DNSBL server using *host*.

```
user@local# host -t A 20.226.96.86.dnsbl-3.uceprotect.net
20.226.96.86.dnsbl-3.uceprotect.net A 127.0.0.2
```

Depending on the DNSBL server, the blacklisting cause could be queried via DNS TXT RR query.

Listing 2 presents the output of a DNS TXT RR query to a DNSBL server with the program `host` executed in a Unix machine. The output indicates that there exists a DNS TXT RR entry, which explains the reasons that caused the respective IP address to be blacklisted.

Listing 2. TXT RR Response from a DNSBL server using *host*.

```
user@local# host -t TXT 20.226.96.86.dnsbl-3.uceprotect.net
20.226.96.86.dnsbl-3.uceprotect.net TXT "Your ISP Emirates-
Internet/AS5384 is Uceprotect-Level3 listed for hosting a
total of 4969 abusers. See:
http://www.uceprotect.net/rblcheck.php?ipr=86.96.226.
```

However, if no DNS A RR is found, it would be considered by the MTA as an indication that the source IP address of the SMTP connection is not blacklisted.

Listing 3 presents the output of a DNS TXT RR query to a DNSBL server with the program `host` executed in a Unix machine. The output indicates that no DNS A RR exists with regards to the IP address 1.1.1.1, which means that the IP address is not blacklisted.

[7] http://www.corpit.ru/mjt/rbldnsd.html

[8] Phishing blacklists could be combined with SPAM blacklists.

Listing 3. DNS A RR query response, regarding an unlisted IP address.

```
user@local# host -t A 1.1.1.1.dnsbl-3.uceprotect.net
1.1.1.1.dnsbl-3.uceprotect.net does not exist
```

C. PhishNet: Predictive Blacklisting

Any changes to a Phishing URL would result in no match. PhishNet [28] addresses the *exact match* limitation found in blacklists.

To solve this, PhishNet processes blacklisted URLs (parents) and produces multiple variations of the same URL (children) via 5 different URL variation heuristics, which are listed below:

- Replace Top Level Domains (TLD): Each URL will fork into 3,210 variations, each with a different TLD.
- Directory structure similarity: If multiple Phishing URLs have similar directory structures with minor variations, multiple children URLs will be created to assemble differences across all the attack URLs that have similar directory structures. For example:
  - http://www.abc.com/online/ebay.html.
  - and http://www.xyz.com/online.paypal.com.

  would result into forking the following children URLs:
  - http://www.xyz.com/online/ebay.html.
  - and http://www.abc.com/online.paypal.com.
- IP address equivalence: URLs with similar directory structure but different domain names are considered as a match if they point to the same IP address.
- Query string substitution: Similar to "directory structure similarity" except that it forks multiple variations of a URL with different query strings. For example:
  - http://www.abc.com/online/ebay.php?ABC.
  - and http://www.abc.com/online.paypal.com?XYZ.

  would result into forking the following children URLs:
  - http://www.abc.com/online/ebay.php?XYZ.
  - and http://www.abc.com/online.paypal.com?ABC.
- Brand name equivalence: Multiple children URLs with same URL, but with different brand names. For example http://www.abc.com/online/paypal.html, would also fork http://www.abc.com/online/ebay.html as a child (since they are often attacked).

The heuristics above are designed such that the URL variations (as made by the attackers) are detected. For example, the *brand name equivalence* heuristic aims at detecting a common URL alteration that was performed by a phishing gang, namely RockPhish, as mentioned in [28].

Many of the forked children URLs may not exist or may be innocent irrelevant pages. So in order to filter/remove non-existent children URLs, the following tests are performed:

- DNS query: does the domain name exist?
- TCP connect: is the resolved name running a HTTP server?
- HTTP header response: does the page exist? (HTTP 200/202 == found).
- Content similarity: is the content similar to parent Phishing attack? If the page is different, it might be an innocent page. An external tool[9] was used to measure the content similarity.

D. Automated Individual White-List

Automated Individual White-List (AIWL) [29] maintains a whitelist of features describing trusted Login User Interfaces (LUIs) where the user submitted his/her credentials. Every LUI will cause a warning except if trusted. Once a LUI is trusted, its features will be stored locally in a whitelist.

There are two primary components of AIWL, namely:

- Whitelist — a list of trusted LUIs. Its objective is suppressing warnings with regards to LUIs that are trusted. In order to detect whether a suspect LUI is trusted, the LUI is represented as a feature vector and then compared against those feature vectors in the whitelist. If the LUI features of a suspect page do not match all of the features in any whitelisted LUI, the suspect page is assumed to be untrusted and warnings are presented to the end-user. Such features that uniquely identify LUIs are:
  - URL address of the trusted LUI.
  - IP addresses that correspond to the trusted LUI. Since a DNS A Resource Record can be mapped to more than one IP address (e.g. for reasons related to load balancing), multiple IP addresses can be mapped to a single LUI.
  - The hash of the X.509 certificate that is associated with the trusted LUI.
  - The Document Object Model (DOM) path of username and password input fields of the trusted LUI. For example, if the input field *username* exists in a form named *loginform*, which is in turn in a frame named *mainframe*, then it will be represented as *mainframe/loginform/username*. In other words, username and password input fields are identified as a directory structure. The objective of this structure is detecting phishing attacks that use *iframe* tags to present login forms of legitimate websites. For example, an attacker could construct a phishing page that presents another page (e.g. PayPal) via the *iframe* HTML tag, and then monitors user's interaction with the page via JavaScript. Identifying username and password fields in a directory structure based on DOM helps in detecting differences between input fields that are presented normally in their original website, and those that are presented in *iframe* HTML tags.
- Automated whitelist maintainer — a classifier that decides whether a suspect LUI to be installed in the whitelist. The merit of the whitelist maintainer is that if an end-user logs in *successfully* for enough amount of times via a given LUI, then that LUI is trusted (and thus whitelisted). This means that the automated whitelist maintainer requires a mechanism from which it is able to decide whether a login was successful. Once the classifier

[9]The content similarity tool is "Similar Page Checker" as made available by *webconfs* via http://www.webconfs.com/similar-page-checker.php.

decides whether a login was successful, it needs to keep track of number of successful logins per suspect LUI, and whitelist it if number of successful login attempts exceed a given threshold (a pre-defined parameter). The input to this classifier is features vector that describes the content of a suspect LUI (what is not in the whitelist), and the output is a label that decides whether to trust the LUI. The features that are used by the classifier to decide whether a login is successful are:

- A binary feature that describes whether the suspect page is in the browser history. Legitimate sites are likely to exist in the history as they have previously been visited.
- A binary feature that describes whether the redirected page, following a login attempt, has a password field. The merit behind this feature is that successful login attempts often result in the redirection to another page that does not have password input field.
- A binary feature that describes whether the redirected page, following a login attempt, has a number of links that exceeds a pre-defined threshold. The merit behind this feature is that when a login fails, the user is often redirected to a page with the login form again, which does not have as many links as the main page after a successful login.
- A binary feature that describes whether the redirected page after the login attempt has a username equal to the same username that was supplied in the login process. If a username is seen again in a form, it would be considered as login failure (since some forms keep the username in the retry form). On the other hand, if the username is not seen in any form following a login attempt, it would be considered as successful login.
- A binary feature that describes whether a user keeps browsing a given web site for a time period that exceeds a pre-defined threshold. The merit of this feature is that phishing web pages are often dealt with very quickly and that the user is likely to close the web page following his login attempt, while legitimate websites are more likely to be browsed by the user following a successful login attempt for an extended time period.

In order for the automated whitelist maintainer to make decisions based on the features described earlier, Ye Cao et. al. propose the use of a Naive Bayes classifier that is given a set of feature vectors that correspond to successful and failure login attempts. Based on such feature vectors, the Naive Bayes classifier constructs a model that is able to generate probabilities associated with regards to future test login attempts. If the probability exceeds a pre-defined threshold, then the login is predicted to be successful.

As stated earlier, once a login attempt in a LUI is predicted to be successful, the automated whitelist maintainer needs also to track the frequency of successful login attempts via the same LUI. If the frequency of such successful attempts exceed a threshold, then the LUI is stored in the local whitelist and the LUI is deemed safe.

VIII. Phishing Detection by Heuristics

Software could be installed on the client or server side to inspect payloads of various protocols via different algorithms. Protocols could be HTTP, SMTP or any arbitrary protocol. Algorithms could be any mechanism to detect or prevent phishing attacks. Phishing heuristics are characteristics that are found to exist in phishing attacks in reality, however the characteristics are not guaranteed to always exist in such attacks. If a set of general heuristic tests are identified, it can be possible to detect zero-hour phishing attacks (i.e. attacks that were not seen previously), which is an advantage against blacklists (since blacklists require exact matches, the exact attacks need to be observed first in order to blacklist them). However, such generalized heuristics also run the risk of misclassifying legitimate content (e.g. legitimate emails or websites).

Currently, major web browsers and mail clients are built with phishing protection mechanisms, such as heuristic tests that aim at detecting phishing attacks. The clients include Mozilla FireFox, Internet Explorer, Mozilla Thunderbird and MS Outlook. Also, phishing detection heuristics could be included in Anti Viruses, similar to ClamAV[10].

A. SpoofGuard

SpoofGuard [30], a web browser plug-in developed by Stanford University, detects HTTP(S)-based phishing attempts as a web browser toolbar, by weighting certain anomalies found in the HTML content against a defined threshold value.

Listing 4 shows samples of HTML heuristically detectable by SpoofGuard.

Listing 4. Sample of heuristic phishing detection based on content.

```
/* Anchor's href attribute is a URI similar to a
   white-listed URL. In this case, www.gmail.com. */
<a href="http://www.gmaii.com">Click Here</a>

/* Anchor's href attribute contains a cloaked URL.
   See RFC2396, Section 3.2.2. */
<a href="http://www.gmail.com@www.eve.com">Click Here</a>

/* Anchor's text attribute is a URL different than the URL
   supplied by href attribute. */
<a href="http://www.eve.com">www.gmail.com</a>

/* Although password fields are harmless themselves,
   some phishing detection heuristics (such as SpoofGuard)
   assign password fields specific values to increase
   level of caution as they might be abused to mimic login
   forms. If the accumulated number for a content exceeded
   a predefined threshold, the content could be reported
   suspected. */
<input type="password" />
```

[10] www.clamav.net

B. Collaborative Intrusion Detection

Many phishing detection and prevention mechanisms are based on finding the source IP address of the attacker.

Fast-flux [31], on the other hand, enables attackers to frequently change their IP addresses. By having a sufficient number of infected hosts (usually home users), hosts can behave as front-end proxies for phishing websites. Multiple front-end proxies relay the traffic back to a main phishing site to fetch the content from (also known as mothership).

Load balancing is achieved by means of low Time to live (TTL) DNS A RR, which enables a quicker change of mapped IP addresses than if higher TTL values were used. Low TTL also helps in reducing the downtime when dead front-end proxies are removed. The DNS A RR are updated by the attacker through a fixed DNS NS RR.

Although fast-flux has a fixed NS record pointing to the attacker's real IP address, double fast-flux complicates the task further by relaying DNS RR update to front-ends as well.

A proposed solution to this is through the use of Collaborative Intrusion Detection System (CIDS) to exchange phishing-related data among a number of Intrusion Detection Systems (IDSs).

The distributed system should be installed globally. Each local CIDS should monitor its local DNS cache, and list DNS zones with a high number of DNS A RR that are combined with low TTL values. The list of IP addresses and domains are sent to global CIDS for further analysis. Each recipient CIDS would in turn have to monitor connections of reported IP addresses (which are infected home users in most cases).

By monitoring the inbound and outbound connections of suspected source IP, it should be possible to find the mothership that is distributing the original phishing content to other front-end proxies.

The proposed mechanism counts the number of connections to the front-end proxies (infected hosts), and distributes such numbers to global CIDS. If the threshold reaches a certain number, it would then be assumed that it is a connection to the mothership (since the mothership has one-to-many connections, it is assumed to have a high number of accumulated connections globally).

This proposed solution has not been implemented yet due to difficulty in studying fast-flux or double fast-flux attacks due to their unrepeated nature so far.

It also faces challenges in determining which connection is toward the mothership. Connection count is not a definitive criteria since many phishing networks could also be connected to other legitimate networks (such as Internet Relay Chat (IRC) or game networks), from which they initially appeared. Thus, doing a simple connection count could lead into false positives.

C. PhishGuard: A Browser Plug-in

The work in [32] bases its protection against phishing on the idea that phishing websites do not often verify user credentials, but merely store them for later use by the phisher.

The authors in [32] acknowledge that, in the future, phishing sites would be more sophisticated and pipe (or tunnel) their output from legitimate sites, acting as a man in the middle attack, which would in turn result in proper success or failure login warnings (as communication is simply piped back and forth). However, the paper states that such use is not common yet, and mainly focuses its detection on non-piped phishing attempts.

PhishGuard's implementation in [32] is a proof of concept that only detects phishing attacks based on testing HTTP Digest authentications. However, integrating other authentication mechanisms, such as through HTML form submission, is possible. PhishGuard follows the following steps to test a suspected page.

1) The user visits a page.
2) If the visited page sends an authentication request, and if the user submitted the authentication form, then PhishGuard starts its testing procedures.
3) PhishGuard would send the same user ID, followed by a random password that does not match the real password, for random n times.
4) If the page responded with *HTTP 200 OK* message, then it would mean the page is a phishing site, and is simply returning fake authentication success messages.
5) If the page responded with *HTTP 401 Unauthorized* message, then it could possibly mean:
   - The site is a phishing site that blindly responds with failure authentication messages.
   - The site is a legitimate site.
6) To distinguish between the two possibilities above, PhishGuard would send real credentials to the website for the $n + 1$ time.
7) If the page responded with a *HTTP 200 OK* message following the login request when the real credentials were used, then the site passes the test and is deemed legitimate.
8) If the page responded with a *HTTP 401 Unauthorized* message, then it could possibly mean:
   - The web site is a phishing site that blindly responds with failure authentication messages. In this case the login credentials of the user were submitted to the phisher. Clearly, the technique is only able to prevent password theft for a subset of phishing websites.
   - The user submitted the wrong password.
9) To ensure that the submitted password is not a wrong password mistyped by the user, PhishGuard stores password hashes in a file and verifies future login requests against it:
   - If the hash of the entered password matches any entry in the file, then PhishGuard concludes that the password was correct, and the site was a phishing website.
   - If no match was found, then PhishGuard would conclude the supplied password is wrong, and the user is then alerted to correct his/her password.

D. Phishwish: A Stateless Phishing Filter Using Minimal Rules

The work in [33] proposes 11 heuristic rules to determine whether an incoming email is a phishing message. The proposed solution aims toward providing:

- Better protection against zero-hour attacks than blacklists.
- A solution that requires relatively minimal resources (11 rules), which is far lower than number of rules in SpamAssassin[11]; at the time of writing the paper [33] SpamAssassin used 795 rules.
- Minimum false positives.

The proposed rules fall under:

- Analysis performed on URL that fall within the email's body.
- Analysis performed on email headers.

The proposed rules are (where *positive* indicates phishiness):

- Rule 1: If a URL is a login page that is not a business's real login page, the result is positive. The paper specifies that this is analyzed based on data returned from search engines.
- Rule 2: If the email is formatted as HTML, and an included URL uses Transport Layer Security (TLS) while the actual Hypertext Reference (HREF) attribute does not use TLS, then the result is positive.
- Rule 3: If the host-name portion of a URL is an IP address, the result is positive.
- Rule 4: If a URL mentions an organization's name (e.g. PayPal) in a URL path but not in the domain name, the result is positive.
- Rule 5: If URL's displayed domain does not match the domain name as specified in HREF attribute, the result is positive.
- Rule 6: If the received SMTP header does not include the organization's domain name, the result is positive.
- Rule 7: If inconsistencies are found in a non-image URL's domain portion, the result is positive.
- Rule 8: If inconsistencies are found in Whois records of non-image URL's domain portion, the result is positive.
- Rule 9: If inconsistencies are found in image URL's domain portion, the result is positive.
- Rule 10: If inconsistencies are found in Whois records of image URL's domain portion, the result is positive.
- Rule 11: If the page is not accessible, the result is positive.

The weighted mean of all of the 11 rules is then considered as the email score, which is then used to predict a class for the email according to a threshold. For example, if the score of a given e-mail is $\geq 50\%$ then it is predicted to be phishing, or legitimate if otherwise.

For example, if a suspect email is being analyzed by Phishwish, the first step is applying the rules from 1 to 11. If a particular rule is not applicable, it is excluded from the calculation of the weighted mean (by setting the multiplier of the rule to zero, which is set to 1 if otherwise). In other words, if an email message does not contain URLs, relevant rules will not be applicable and excluded from the calculation of the weighted mean. For sake of simplifying this example, we will assume that the outcome from the 11 rules above for this example suspect email are: 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, and N/A for the rules 1-11 respectively. In order to calculate the weighted mean, we will assume that all of the rule outcomes are given an equivalent weight of 1 (note that weights other than 1 can be chosen as well). To calculate a score of this suspect email, the outcome of the following equation should be calculated as follows: $\frac{1\cdot1+1\cdot1+1\cdot1+1\cdot1+1\cdot1+1\cdot1+1\cdot0+1\cdot0+1\cdot0+1\cdot0}{1+1+1+1+1+1+1+1+1+1+1} = 0.6$. Since the outcome of the equation is $\geq 50\%$, the suspect email is assumed to be a phishing email. Note how the inapplicable rule (i.e. N/A) was excluded from both the nominator and the denominator as its multiplier was set to 0 (although we did not present the rule multiplier for visual clarity, each rule's weight is also multiplied by either 1 or 0 depending on whether the rule is applicable or not respectively).

E. CANTINA: A Content-Based Approach

CANTINA [34] is an Internet Explorer toolbar that decides whether a visited page is a phishing page by analyzing its content. CANTINA uses Term Frequency-Inverse Document Frequency (TF-IDF), search engines and some heuristics to reduce false positives. The following procedures are performed by CANTINA to detect phishing websites:

1) TF-IDF of each term on a suspected web page is calculated.
2) Top 5 terms with highest TF-IDF values are taken to represent the document (Named lexical signature in [35]).
3) Submit the 5 terms into a search engine (e.g. Google search query: http://www.google.ae/search?q=t_1, t_2, t_3, t_4, t_5,), and store domain names of the first returned n entries.
4) If the suspected domain name is found within the n number of returned results, then the site is legitimate.

TF-IDF of a given term i in document j is the product of TF and IDF values of the same term i. Equations (10) and (11) present how TF and IDF are calculated respectively.

$$TF_{ij} = \frac{n_{i,j}}{\sum_{m=1}^{k} n_{m,j}} \tag{10}$$

where $n_{i,j}$ is the frequency of the term i in document j, $n_{m,j}$ is the frequency of the term m in document j, and k is total number of terms in document j.

$$IDF_i = \log \frac{|D|}{|D_i|} \tag{11}$$

where $|D|$ is the cardinality of all documents available in a given corpus, and $|D_i|$ is the cardinality of documents with the term i in the same corpus. According to the authors in [34], the corpus used to calculate IDF_i in the study is the British National Corpus (BNC)[12].

[11] A free open source software project that aims at detecting spamming email messages, which also incorporates the detection of phishing emails. The project's website can be accessed via http://spamassassin.apache.org/

[12] http://www.natcorp.ox.ac.uk/.

Thus, TF-IDF values for each term i is calculated by Equation (12).

$$TFIDF_{i,j} = TF_{i,j} \cdot IDF_i \tag{12}$$

For example, in order to calculate the TF-IDF of the term "bank", the following steps are performed:

1) Assuming that the term "bank" appeared 5 times in a suspect document, and that total number of words in the same document is 500, then $TF = 5/500 = 0.01$.
2) Assuming that the term "bank" appeared in 10,000 documents as counted against a corpus of 1,000,000 documents, then $IDF = log(1,000,000/10,000) = 2$.
3) Finally, $TFIDF = TF \cdot IDF = 0.01 \cdot 2 = 0.02$.

To reduce false positives, the following heuristics are used:

- Domain age — if older than 12 months then it is likely to be legitimate.
- Suspicious page URI — if contains - or @ then it is a phishing website.
- Suspicious link in content — if contains - or @ then it is a phishing website.
- Dots in URI — if contains more than 5 dots then it is a phishing website.
- Forms — if HTML forms exist, then it's likely to be a phishing website.
- TF-IDF — whether a page is suspected phishing based on CANTINA's document classification, which is based on TF-IDF.

Each of the heuristics are given values based on experiments conducted by the authors, and the state of each page is calculated by (13):

$$S = f(\sum w_i \cdot h_i) \tag{13}$$

Where h_i is each heuristic's result, w_i is the weight of each heuristic accordingly and f is a threshold function that returns 1 and -1 to represent legitimate and phishing sites respectively.

F. A Phishing Sites Blacklist Generator

The authors in [36] proposes a mechanism to build blacklists dynamically through the use of search engines, such as Google. The proposed work detects phishing sites, and then stores them in a database.

The evaluation data set is composed of 500 legitimate websites as randomly chosen from Google search results when given random search keywords[13], and 30 phishing websites as collected from PIRT[14].

The proposed heuristics in [36] are:

1) Extract company name from the suspected URL.
2) Search for the extracted company name in Google, and return the first 10 results.
3) If the suspected URL belongs to the first 10 returned Google results, then the page is legitimate.
4) If the suspected URL does not belong to the first 10 returned Google results, then the suspected URL is classified as phishing.
5) If the suspected URL is classified as phishing, it will be saved in a database.

When the above heuristic test was applied against the legitimate websites, only 45 out of the 500 websites were misclassified as phishing websites (i.e. 9% FP rate), and when the test was applied against the phishing websites, all of them were correctly classified as phishing (i.e. 100% TP rate).

The authors acknowledge that this process introduces additional delay in the Internet browsing experience, thus they suggest placing the blacklist generator in a mail server, where all URLs in all messages are analyzed by the blacklist generator on the mail server. This enables the mail server to process the URLs before the users request them and if the outcome of the blacklist generator is cached, the delay on the user side can be reduced noticeably.

As claimed by the authors, the choice of email server is desirable since email messages are the most popular mechanism for spreading links to phishing sites.

IX. Phishing Detection by Visual Similarity

This section outlines a number of proposed solutions that attempt to detect phishing attacks based on their visual appearance, as opposed to analyzing the underlying source code or network-level information.

A. Classification with Discriminative Keypoint Features

Unlike other anti-phishing mechanisms, the proposed solution in [37] approaches phishing detection based on the content presentation, instead of content code. In other words, this phishing detection mechanism is agnostic to the underlying code or technology that renders the final visual output to the users eyes. For example, a phishing site that mimics a legitimate site by displaying similar content using *img* HTML tags (instead of other Hypertext Markup Language (HTML) tags that can be used to render similar visual output) might bypass anti-phishing mechanisms that base their detection on the HTML content, as image contents are invisible to them. Such phishing attempts are expected, by this proposal, to be detected.

This proposed solution requires the web browser to take a snapshot of every suspected site. The snapshot is then matched against a whitelist of protected websites that are likely to be targeted by phishers (e.g. PayPal, Amazon, eBay, banks websites, etc...).

Once the snapshot of the suspected site is taken, RGB channels are converted into a Grayscale channel by averaging Red, Green and Blue values.

The resultant Grayscale image is analyzed to find key features or salient points. The mechanism used to detect salient points is through detection of corners. This proposal uses the Harris-Laplace algorithm to detect corners in an image. The advantage of Harris-Laplace algorithm is its accuracy in conditions of different resolutions and rotations.

[13]Ten keywords were chosen randomly from a pool of 150,000 dictionary words, and the top 50 best ranked websites were taken as legitimate website samples.

[14]Phishing Incident Reporting and Termination.

Each Harris-Laplacian corner (or salient point) found in the snapshot is then described by a 48-dimensional vector, as specified by Contrast Context Histograms (CCH) [38], which is simply based on the relative contrast values of pixels surrounding the Harris-Laplacian corner. Pixels surrounding corners are ones that fall under the coverage of a defined log-polar grid, centered by the detected corner. 24 sub-regions exist in the polar grid. Each sub-region includes the number of pixels. The difference between the contrast values of each of these pixels against the Harris-Laplacian corner's contrast value is calculated. As a result, each sub-region would have a number of positive and negative pixels, which represent their difference against the corner. Positive and negative numbers would be separately accumulated on per-sub-region basis. For example, a sub-region that has *+1, +5, -10, -7, +32, -2* pixels, would be accumulated into *+38, -19*.

CCH descriptors ($CCH(p_c)$) of corners or salient points (p_c) are described by their surrounding relative positive and negative contrast values within each sub-region in the polar grid, which results in a 48-dimensional vector ($24 \times 2 = 48$). See Equation (14).

$$CCH(p_c) = \{p_1+, p_1-, p_2+, p_2-, \ldots, p_{24}+, p_{24}-\} \quad (14)$$

where p_1 to p_{24} represent sub-regions 1 to 24 respectively, and p_n+, p_n- represent positive and negative relative contrast values against p_c in sub-region n.

Matching salient points is based on calculating the Euclidean distance between salient points found on a suspected site, against salient points found on protected whitelisted sites. The distance ratio would determine whether a match for a salient point exists. Ratio equal to 1 would mean perfect match for a particular salient point, while 0 would mean a perfect mismatch.

Practically, achieving 1 or 0 ratios is not realistic. Thus, a threshold of 0.6 was used in experiments of this proposal to identify matches.

Salient points matching, alone, might trigger high false positive rates. For example, websites that publish the logos of VeriSign or Anti-Hacker might falsely be detected as phishing attacks.

To solve the above problem, locations of salient points is taken into account, through the use of k-means clustering algorithm.

Once k-means clusters converge, salient points would be grouped into 4 clusters[15], which are in turn used as a mechanism to indicate the relative location of salient points. For example, a salient point that's expected to be in the header, would be matched against a similar location on the opposite grayscale snapshot.

A suspect site is considered visually matched against a whitelisted CCH if the number of salient point matches within a k-means cluster is more than 50% of the total number of unmatched salient points in the same cluster. The suspect site is therefore considered legitimate if its URL matches that of the whitelisted CCH entry, or phishing if otherwise.

[15]Assuming $k = 4$.

In other words, the visual appearance of a particular page is allowed to be accessed by trusted URLs only, and if another suspect URL contains content that is visually identical to that of another website while having a different URL then the suspect URL will be considered a phishing website.

B. Visual Similarity-based Detection without Victim Site Information

The goal of the work in [39] is detecting phishing sites based on visual similarity without the need of whitelisting pictures of all legitimate websites, and is based on the fact that most phishing websites aim at being visually similar to their target websites (e.g. PayPal phishing websites aim to look visually similar to the legitimate PayPal website in order to maximize their chances of persuading more victims).

To achieve the above objectives, the following steps are performed:

1) A list of previously known legitimate W_L and phishing W_P websites are processed to function as baseline for the classifier. Each of the sites are processed and stored in a database that presents each website by the following elements:
   - A label (legitimate/phishing) that determines the class of the website.
   - A screen-shot of the website as rendered by a web browser.
   - The domain name of the website.
2) Each suspect website's (W_s) screen-shot (as rendered by a web browser) is taken and then used to find other websites in the database that have similar visual appearance. The visual similarity search is performed by imgSeek[16], which returns a number that reflects how similar an input image is compared to other images in the database. Using a threshold function, a crisp classification decision can be made.
3) Based on the outcomes of imgSeek, Algorithm (2) is executed.
4) To reduce false positives, a whitelist of domain names is used.

Algorithm 2 Phising website detection based on visual similarity

if $imgSeek\Big(img(W_s), img(\{W_L, W_P, W_U\})\Big)$ **then**
  $W_b \leftarrow bestMatch(W_s)$
  if $domain(W_s) = domain(W_b)$ **then**
    $return(C_l)$
  else
    $return(C_p)$
  end if
else
  $store(\{img(W_s), domain(W_s), C_u\})$
end if

[16]A collection of opensource image similarity tools. More details can be found in http://www.imgseek.net/

In Algorithm (2), img is a function that returns the image of its input website, $imgSeek$ is a function that accepts two arguments and returns true when a match is found for the image passed as its first argument against a list of images passed as its second argument, W_U is a set of websites that are classified as "unknown", $bestMatch$ is a function that returns the best matching website according to imgSeek's visual similarity, C_l is the classification label for legitimate websites, C_p is the classification label for phishing websites, C_u is the classification label for unknown websites, and $store$ is a function that stores its input vector in the websites database.

The motive behind storing previously-unseen websites with a classification label of "unknown" is to enable browsers to expand their database of trusted websites as the end-user keeps browsing more web pages. If a user browses a website with a unique visual appearance, it will be assumed to be a website that was never previously browsed. Storing the features vector of this website (i.e. its domain name, screen-shot and its classification label) will insure that subsequent visual matches against this website is considered suspicious.

For example, if an end-user browses PayPal for the 1^{st} time, and if PayPal's features vector is not stored in the local database, then it will be stored as an unknown website. In a later time, other websites (i.e. a websites with different domain names than PayPal's) are considered phishing websites if their screen-shot is visually similar to PayPal's recently stored features vector.

X. Phishing Detection by Data Mining

Techniques that are described in this section consider the detection of phishing attacks as a document classification or clustering problem, where models are constructed by taking advantage of Machine Learning and clustering algorithms, such as k-Nearest Neighbors (k-NN), C4.5, Support Vector Machines (SVM), k-means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

For example, k-NN stores training instances in memory which are represented as multi-dimensional vectors, where each vector component represents the extracted value from a particular feature (e.g. number of URLs in an email message). The classification task is then performed by similarly processing testing instances, and calculating the distance (e.g. euclidean distance) between the testing instance and the other training instances. When $K = 3$, the classes of the 3 nearest neighbors (as obtained during the training phase) are considered. When the task is classification, majority voting can be used to determine the class of the testing instance.

Algorithms such as C4.5 and SVM follow a different approach where they generalize a classification model (as opposed to k-NN, which does not generalize a model). For example, C4.5 constructs a decision tree that should be generic enough to correctly classify unseen instances. The decision tree is composed of nodes with splitting branches. The splitting is generally performed to maximize the conditional Information Gain after the split. On the other hand, SVM aims at finding an effective separation plane in a vector space by analyzing the training instances. The separation plane should be generic enough so that it should still be able to separate unseen instances.

However, clustering algorithms such as k-means and DBSCAN partition instances in an unsupervised manner (i.e. knowing the class label is not required to construct the clusters). k-means algorithm aims at constructing k partitions by randomly setting k initial partition centers, followed by iteratively assigning instances to a partition with the smallest distance (e.g. euclidean distance) towards its center, and then updating the partition center to be the mean of the instances in the same partition. This iterative process is repeated until the clusters converge. On the other hand, DBSCAN is able to partition the data based on the density (i.e. using a distance function measure, such as euclidean distance) of the instances. Contrary to k-means, DBSCAN does not need to know beforehand the number of partitions that should be found, which is achieved by the concept of *density reachability*.

A. Automatic Detection of Phishing Target from Phishing Webpage

Gang Liu et al. [40] suggest that it is possible to detect phishing websites and their targets by finding similar websites to suspect pages. If a suspect website has similar websites to it with a different domain name, then the suspect website is assumed to be a phishing website, and the most similar website to the suspect website is assumed to be the targeted website. For example, if Paypal is found to be the most similar website to a suspect website, then the suspect website is assumed to be a phishing website that targets PayPal. This is based on the assumption that phishing websites attempt to be similar to their target websites in order to maximize the deception of their victims.

The evaluation data set is composed of 8,745 phishing websites collected from PhishTank, and 1,000 legitimate websites collected from Random Yahoo! Link, a website that returns random URLs of other websites.

In order to detect phishing websites and their targets, the proposed technique processes a suspect website W_s as follows:

1) W_s is expanded into multiple other similar websites by:
   - Extracting URLs of other websites $W_D = \{d_1, \ldots, d_{|W_D|}\}$ that are *directly* linked in the HTML body of W_s.
   - Extracting keywords $K_S = \{k_1, \ldots, k_{|K_S|}\}$ (e.g. keywords found in meta-tag, title, organization name, body) from W_s, and then submitting keywords in the K_S set to a search engine to find a set of other websites $W_I = \{i_1, \ldots, i_{|W_I|}\}$ which are considered to be *indirectly* associated with W_s.
2) The following features are extracted from each website in the sets W_D and W_I:
   - Link relationship LR — counts the total number of forward links $NL_{W_s \to Page_j}$ from W_s to $Page_j$, where $Page_j$ is the j^{th} page in $W_D \cup W_I$, and then normalizes it by the total number of links NL in W_s as in Equation (15). This feature returns a normalized number from 0 (when there is no linking

relationship) to 1 (when there is a perfect linking relationship).

- Ranking relationship RR — keywords found in the set K_S are submitted to a search engine to return a ranked list of websites. This list is used to assign ranks for domain names of websites in the sets W_D and W_I. The ranks are then normalized as in Equation (16). This feature returns a normalized number from 0 (when the domain of $Page_j$ is not returned in the ranked list) to 1 (when the domain of $Page_j$ is returned as the 1^{st} in the ranked list).
- Text similarity relationship TS — the cosine similarity between term vectors x and y which are extracted from the websites W_s and $Page_j$ respectively, as presented in Equation (17). Since the term vectors were constructed by TF-IDF, and since TF-IDF can not be negative, then the returned value of this feature ranges from 0 (no similarity) to 1 (exact similarity).
- Layout similarity relationship LS — the number of visually similar blocks between W_s and $Page_j$, normalized by the total number of blocks in W_s as presented in Equation (18). This feature returns a normalized number from 0 (for no layout similarity between the pages) to 1 (for exact layout match).

3) Following the feature extraction phase, each website $Page_j$ is represented as four dimensional vector $V_j = \{LR, RR, TS, LS\}$. The suspect website W_s is presented as $V_s = \{1, 1, 1, 1\}$ as it is assumed to make an identical match to itself. The closer the vector V_j is to V_s, the more likely that W_s is a phishing website and that $Page_j$ is a targeted website.
4) In order to detect the similarity between W_s and $Page_j$, the authors used DBSCAN to test whether a density-based cluster can be formed around the vector V_s. The best DBSCAN parameters were found to be Eps[17] = 0.11 and MinPts[18] = 4.

$$LR = \frac{NL_{W_s \to Page_j}}{NL} \tag{15}$$

$$RR = \frac{NR - (NR_{K_S \to Page_j} - 1)}{NR} \tag{16}$$

where NR is total number of ranked websites, and $NR_{K_S \to Page_j}$ is the rank of the domain name of $Page_j$ when keywords K_S are submitted as a search engine query.

$$TS = \frac{x \cdot y}{||x||||y||} \tag{17}$$

$$LS = \frac{NB_{W_s} \cap NB_{Page_j}}{NB_{W_s}} \tag{18}$$

where NB_{W_s} is total number of blocks in W_s, and NB_{Page_j} is total number of blocks in $Page_j$.

[17]The length of the vector diameter. This is used to search for number of other vectors that may fall within this diameter.

[18]The minimum number of other vectors that should exist within the Eps region in order to consider the vector as density reachable.

B. Detecting DNS-poisoning-based phishing attacks from their network performance characteristics

Kim, H. et al. [41] propose a mechanism to detect DNS-poisoning attacks (which could potentially be used to direct users to phishing websites) by analyzing network-level characteristics of end-users communications. The collected data was composed of 10,000 routing information items, 50% of which were towards phishing websites (websites URLs obtained from PhishTank), and the remaining 50% were destined towards legitimate servers.

The routing information of the network datagrams were extracted as follows:

- The existence of a firewall before the accessed service.
- The mean Round-Trip Time (RTT) towards the accessed service.
- The standard deviation of RTT.
- The hop count between the end-user network and the accessed service.

The evaluation data set was composed of routing information that represent 10,000 websites, with 1:1 phishing to legitimate website ratio. The data set was constructed by collecting routing information for 5,000 instances for 50 highly targeted websites (10 instance per website), and 5,000 phishing instances (100 instance per website).

As preliminary analysis, it was found that only 19% of the phishing websites were behind firewalls, while 79% of the legitimate websites were behind firewalls. The authors justified this by the fact that phishing websites are hosted in less secure shared web hosts than their legitimate counterparts, which enabled attackers to illegally host their phishing websites.

The authors then processed the extracted routing information by a number of learning algorithms, such as SVM and k-NN. Their findings showed that the best performing classifier was k-NN with $k = 1$.

As a future work, the authors suggest the addition of more features, such as the RTT between the end-user's network and every layer 3 hop in the path towards the target service.

C. Textual and Visual Content-Based Anti-Phishing: A Bayesian Approach

Haijun Zhang et al. [42] propose a technique that guesses a class label for suspect websites by analyzing their textual and visual contents. Similar to [40], the basic idea is detecting websites that are similar to known legitimate websites. If such condition is met (i.e. a website is detected to be similar to a legitimate website, such as PayPal), then the suspect website is assumed to be a phishing website.

The evaluation data set is composed of 10,272 legitimate URLs as collected from Google, a set[19] of phishing URLs as collected from PhishTank, and 8 protected[20] websites as follows:

[19]The paper [42] does not state how many were the phishing URLs. However, based on our analysis of PhishTank's URL's blacklist between Jun 15 2010 and March 18 2012, the number of phishing URLs in PhishTank ranged from 2,306 to 9,826.

[20]A protected website is a website that no other website should mimic its identity.

- https://signin.ebay.com
- https://www.paypal.com/c2
- https://ssl.rapidshare.com/premiumzone.html
- http://www.hsbc.co.uk/1/2/HSBCINTEGRATION/
- https://login.yahoo.com
- https://www.mybank.alliance-leicester.co.uk/index.asp
- https://www.optuszoo.com.au/login
- https://steamcommunity.com

The authors proposed a number of techniques, and the most effective technique was composed of the following:

- A naive Bayesian model that returns a normalized number reflecting how likely the textual content of a suspect website is phishing (or legitimate).
- An image processing technique that compares the visual characteristics of a suspect website against that of legitimate websites. The intuition is similar to the naive Bayesian model with the exception of processing visual appearance of the website as opposed to its textual content.
- A method to combine the above techniques to return a final classification label.

The naive Bayesian model returns a normalized number that reflects how likely a suspect website is to be a phishing website. In order to achieve this, the following computations are made:

- The human readable text is extracted from the websites (for both training and testing phases), words are stemmed[21], and stop words are removed. This results in a set of keywords $K_j = \{k_1, \ldots, k_{|K_j|}\}$ for each website j.
- The posterior probability of website j being a phishing website is calculated by Equation (19). $P(Phishing)$ is total number of phishing instances in the training set, divided by the total number of instances in the same training set. However, since calculating $P(K_j|Phishing)$ is difficult[22], $P(K_j|Phishing)$ is measured as in Equation (20) which follows the naive assumption that words in natural languages occur independently from each other. Similarly, $P(K_j)$ is calculated by following the same naive assumption of words independence as presented in Equation (21).
- In order to guess a crisp class label for the suspect website, a threshold function is applied on the returned output from $P(Phishing|K_j)$.

$$P(Phishing|K_j) = \frac{P(K_j|Phishing)P(Phishing)}{P(K_j)} \quad (19)$$

$$P(K_j|Phishing) = \prod_{i=1}^{|K_j|} P(k_i|Phishing) \quad (20)$$

$$P(K_j) = P(K_j|Phishing) + P(K_j|Legitimate) \quad (21)$$

[21]In order to reduce number of unique words in the training database.

[22]Due to the extremely complex semantic relationship between words in natural languages (i.e. words in K_j).

The visual similarity technique functions as proposed previously in [43], which takes snapshots of websites as rendered by a web browser (Internet Explorer in their case), transforms them to 100 x 100 pixels images, generates a features vector, and then returns a normalized number that reflects the similarity between the images. The returned normalized number is 1 when the images are identical, or 0 when they are not. In order to guess a crisp class label for the suspect website, a threshold function is applied on the returned number.

In order to combine the returned normalized numbers from both naive Bayesian and the visual similarity techniques, the authors divided the returned interval [0, 1] into bins (or partitions of sub-intervals) $[B_0, B_1], \ldots, [B_{i-1}, B_i]$ and then measured the probability of each technique being correct given that the returned normalized number falls within a particular bin. For example, $P(Text_C|B_i)$ is the probability that the output from the textual analysis by the naive Bayesian technique $Text_C$ is correct, given that the returned value falls in the bin B_i. Similarly, $P(Visual_C|B_i)$ is the probability that the output from the visual analysis technique $Visual_C$ is correct, given that the returned value falls in the bin B_i.

If $P(Text_C|B_i) \geq P(Visual_C|B_i)$ then the output from the naive Bayesian technique is used for guessing a class label for the suspect website j (ignoring the returned value from the visual analysis technique). However, if $P(Text_C|B_i) < P(Visual_C|B_i)$ is the case, then the output from the visual analysis technique is used to guess a class label for the suspect website j.

D. Detecting Phishing Emails Using Hybrid Features

The work in [44] proposes the evaluations of multiple features (using different methods) in a document (hence the name hybrid) to decide whether a document contains phishing attempts.

Once each document j is evaluated against multiple features $F = \{f_1, \ldots, f_n\}$ where n is the total number of features, each document is represented as a multi-dimensional vector $D_j = (f_{1,j}, f_{2,j}, f_{3,j}, \ldots, f_{n,j})$.

Each element in the document's vector D_j is a value that represents the outcome of a feature evaluation against document j normalized to the range [0,1]. Some features return binary values, such as whether a form exists in a document, while other features may return numbers of blacklisted word matches in a document which could be numbered in a hundreds. The features evaluation output is normalized since different features result numbers in different ranges.

Information Gain (IG) is then used to select a features subset that is, hopefully, more effective in enabling Machine Learning (ML) algorithms to construct accurate classification models while reducing the quantity of the features.

Once a feature set is chosen via IG, multiple learning algorithms are used (e.g. C4.5, Random Forests (RF), SVM) to construct classification models. Each classification model is evaluated and the most accurate model is then used to classify email messages.

E. Large-Scale Automatic Classification of Pages

The authors in [4] describe an anti-phishing solution implemented by Google to rapidly and promptly classify pages while maintaining low false positives. Once the classification is done, the result is aggregated into a blacklist that is published through Google Safe Browsing API (which is used by Firefox, Google Chrome, and Apple Safari).

It addresses a delay problem in human-driven phishing classification such as what PhishTank does, which (as claimed and referenced in the paper) took PhishTank 50 hours (as the median) in order to verify a phishing URL in June 2009. Google's approach significantly reduces human involvement in order to provide blacklists more promptly.

The proposed classifier operates as follows:

- GMail users manually classify unwanted emails as *Junk*.
- Google considers URLs that fall within Junk emails as candidates only if the same was submitted by a number of users (to reduce abuse).
- Candidate URLs are processed for feature extraction. Examples of URL features are:
  - Whether hostname is an IP address.
  - Number of subdomains (e.g. 5 subdomains are more common in Phishing URLs than legitimate).
  - Whether a path contains certain tokens. Some tokens are more common in Phishing URLs than legitimate (e.g. trademarks of victim sites).
- Candidate URLs are sent to processes that extract features from page content. The content is obtained from Google's cache, which is built by their web crawler. Examples of features are:
  - Whether password fields are included.
  - Highest TF-IDF terms.
  - The extent of linking objects to other domains (i.e. some Phishing sites use images from victim sites directly instead of copying them to their temporarily hosted Phishing campaign site).
- A classifier (which is an administered machine learner classifier) then uses the resultant output of extracted features. The outcome is scaled from 1.0 to 0.0 (0.0 being highly phishing, while 1.0 being highly legitimate). Practically, as tested by Google, the results were mostly at the two extreme ends, which makes classification clearer. The same classifier is also run against pages visited by Google crawlers to detect more Phishing attacks.
- The above classifier learns as it classifies more documents.

F. Bayesian Anti-Phishing Toolbar (B-APT)

The work in [45] proposed a FireFox toolbar that uses:

- Bayesian classification to decide whether a web page is a phishing page. This is achieved by statistically analyzing keywords (or tokens[23]) in a web page. For example, if a token appeared 10 times in phishing websites, and 90 times in legitimate websites (according to the training set), then the token is considered to be 90% legitimate or 10% phishing. The probability that each website is phishing or legitimate is dependent on the probability of the token components, which need to be combined in order to represent the overall probability of the page.
- A whitelist to reduce false positives.
- A recommendation system.
- A new user interface that increases the chance of proper user reaction.

B-APT uses Bogofilter[24] for its Bayesian classifier implementation, which follows a naive assumption that tokens are independent of each other (i.e. probability that token 1 exists is not influenced by existence of token 2), which is inaccurate as keywords in natural languages are dependent on each other. However, this naive assumption simplifies the implementation and reduces computational cost.

Bayesian filters are initially trained by sample documents. Document samples are manually grouped as phishing or legitimate in order to allow the learner to weight tokens more accurately. For example, if word W_1 appeared 20 times in sampled phishing documents, while 2 times in sampled legitimate documents, it would then mean that a site that has W_1 word is likely to be a phishing site.

A whitelist is used by B-APT to reduce false positives. This is due to the fact that phishing sites are very close to authentic ones, and comparing site tokens against Bayesian classifiers may lead to high false positives. If a requested URL is found in the whitelist, it would be allowed to be viewed without warnings.

Whitelists are stored in *(Company name, URL)* tuples. If the requested URL exists in the whitelist, it would be viewed without warnings. However, if a URL does not exist in a whitelist, but matched some tokens in an existing tuple in the whitelist, the whitelist would return a list of matched URLs. The recommendation system would instead hint to the user whether he wanted to visit the returned URL list (which is legitimate as it is already whitelisted).

If a phishing site is detected, the toolbar places a gray overlay over the page, in addition to a red warning message on top of the overlay and a pop-up window.

G. Natural-Language Processing for Intrusion Detection

Allen Stone in [15] proposes the use of Natural Language Processing (NLP) for intrusion detection systems, which also detects Phishing attacks. The IDS is capable of detecting pattern matches, however its uniqueness is when it detects messages based on their semantics. The system is named EBIDS, and its operation overview is as follows:

1) Message bodies are extracted and sent to a detection mechanism.
2) The detection mechanism relies on OntoSem to categorize the message — the component that deploys NLP techniques.
3) The IDS signatures are based on literal string matches against OntoSem's output.

[23] A token could be a word, hostname, IP address, etc...

[24] An open source Bayesian spam classifier.

4) Based on the matches, an email is classified as either phishing or legitimate.

EBIDS uses four sets of signature rules, namely:

- Account compromise — a rule that is expected to detect phishing attacks formed as email messages that semantically claim the victim's account is compromised.
- Account change — a rule that is expected to detect phishing attacks formed as email messages that semantically claim the victim's account is modified.
- Financial opportunity — a rule that is expected to detect Phishing attacks formed as email messages that semantically claim financial opportunities, such as monetary gains.
- Opportunity — a rule that is expected to detect Phishing attacks formed as email messages that semantically claim opportunities that do not strictly refer to monetary gains, such as free vacations.

H. Model-Based Features

Andre Bergholz et. al. in [46] proposed a novel ML phishing email classifier based on basic features, external features, model-based features and image processing. The features are described below:

- Basic features — essentially heuristic features, such as whether an email's content is HTML, or whether it contains JavaScript.
- External features — used to receive email classification output from another classifier. The paper [46] used SpamAssassin class predictions output as input to the SVM classifier. In other words, this classifier makes decisions based on its own heuristics as well as output from other classifiers to further enhance its prediction accuracy.
- Model-based features — namely Dynamic Markov Chain (DMC) feature and Latent Class-Topic Model (CLTOM) feature (more details later).
- Image processing — used to detect hidden salting attempts. For example, phishers might insert human-unreadable text into a message in order to confuse classifiers. Image processing techniques are used here to detect how the text appears to the human eye, rather than how it appears to the machine.

The DMC features basically use Markov Chains to estimate the probability of a message originating from a class by analyzing email's text sequences on a bit by bit basis. In order for the DMC model to be constructed, a training phase would be required.

The CLTOM features attempt to find words that when they co-occur, they indicate the class of an email. For example, if words *click* and *account* co-existed in a single message, then the message is likely to be phishing. Similar to DMC, a CLTOM model requires a training phase.

Using the features above, each input email is converted into a multi dimensional vector, where the vector elements are values returned by the features described above. The vector is then processed via SVM to construct a classification model. The constructed SVM classification model is then used to predict classes of unclassified input emails.

In other words, there are two training phases:

- a training phase to construct DMC and CLTOM feature models (and thus the name model-based features),
- and once the DMC and CLTOM models are constructed, another training phase is required to construct the parent SVM classification model.

I. R-Boost: A Classifier Ensemble

Authors in [47] were able to raise the recall rate of phishing email classification by proposing a novel boosting technique.

The authors used combinations of C5.0, k-NN with $k = 3$ and 4, and SVM. The used ensembling technique is majority voting. For example, all individual learners return their guessed class for an input test instance, and the majority class would be assumed to be the final class returned by the ensemble. Since majority voting is used, the authors selected an odd number of learners in order to avoid ties during the voting process (3 learners in this case).

To represent the input email instances as feature vectors, the authors used only five simple numerical heuristic features to speed up the classification task. The features are:

- IP address: a binary feature that is 1 if an IP address is found in a suspect email.
- HTML: a binary feature that is set to 1 if the email is written in HTML.
- Script: a binary feature that is set to 1 if the email contains JavaScript code.
- Number of URLs: A numeric feature that represents the total number of links found in a suspect email.
- Maximum number of periods in a URL: the highest number of periods/dots found in a URL in a suspect email.

Following the training phase, a number of classification models are created:

- C5.0.
- k-NN where $k = 3$.
- k-NN where $k = 4$.
- SVM.

The classifiers, k-NN where $k = 3$ and 4, and SVM are then combined by majority voting to form an ensemble.

In order to classify email messages, the following steps are performed:

1) Input email messages are processed in order to be represented as five dimensional feature vectors.
2) Each features vector is passed as input to the C5.0 classification model, where the model outputs a label (e.g. either phishing or legitimate).
3) If the C5.0 model outputs "phishing" as the class label, then the email is considered to be phishing.
4) However, if the C5.0 model returned "legitimate" as the class label, then the corresponding input features vector is passed to the ensemble to re-evaluate its class. The ensemble returns either "phishing" or "legitimate" for the instances that were classified as "legitimate" by the parent C5.0 model.

The motive behind this technique is enhancing the recall rate of the classifier (i.e. the rate of correctly classified phishing instances out of all existing phishing instances).

XI. Evaluations of Human Training Approaches

Not all user training materials have, necessarily, the same effect. This section presents an evaluation of different educational materials as conducted by Steve Sheng et al. [1]:

- Popular training materials — three materials, namely: Microsoft Online safety [48], OnGuardOnline phishing tips [49], and National Consumer League Fraud tips [50].
- Anti-Phishing Phil — an educational interactive game created by Carnegie Mellon University (CMU) and commercialized by Wombat[25] that is playable via web browsers [51].
- PhishGuru Cartoon — PhishGuru is a training system embedded into mail systems, which attempts to teach users at the "most teachable moment", which is once the user falls victim of a phishing attack by (for example) clicking on a link in a test phishing email. The phishing emails are simulated and expected to be generated by the system administrators [11]. However, this study only uses the educational interventions, which are cartoon images to train the users; that is, PhishGuru Cartoon in this study is essentially PhishGuru without the embedded training part [1].
- Anti-Phish Phil with PhishGuru Cartoon — simply both of the user training materials combined.

The evaluation of the above user training materials was made online, with a total number of 1001 participants. The evaluation was formatted as a role play where participants were given scenarios of email messages and questions to answer.

There were the following groups of participants:

- Control — participants in this group are not given any educational materials.
- Popular training — participants in this group are only given popular training materials described above.
- Anti-Phish Phill — participants in this group are only given Anti-Phish Phill as their training material.
- PhishGuru — participants in this group are only given PhishGuru as their training material.
- Combined — participants in this group are given two educational materials, namely: Anti-Phish Phill and PhishGuru.

Each of the participating groups had to complete two role plays:

- 1^{st} role play — none of the participating groups above are given any training materials.
- 2^{nd} role play — all of the participating groups are given their respective training materials, with the exception of the *control* members which are not trained.

The participants were randomly assigned to either the Control (no training) group, or the training groups (one of the listed groups above). The reactions of the participants were then recorded to assess the effectiveness of the user training materials.

The study concluded that user training approaches are promising. Virtually all user training approaches increased TP rate. However, the study paid attention to an important point which is that most of the user training materials have also decreased the TN rate; in other words, some users did not react to some legitimate emails as their training materials scared them off. See Table II for details.

Training approach	FN		TN	
	1^{st} role play	2^{nd} role play	1^{st} role play	2^{nd} role play
Control	50%	47%	70%	74%
Popular training	46%	26%	67%	61%
Anti-Phish Phil	46%	29%	73%	73%
PhishGuru	47%	31%	70%	64%
PhishGuru + Anti-Phish Phil	47%	26%	68%	59%

TABLE II
Evaluation results of user training approaches.

Table II shows that all training solutions decrease FN by 39.79% in average (good), and that all of them decrease TN by 7.69% in average (bad) except for Anti-Phish Phill which did not reduce the TN rate.

Interestingly, Table II also shows that the participants in the untrained group (i.e. *control*) were able to achieve a lower FN rate in the 2^{nd} role play despite the fact that they were not given any training. The untrained participants were also the only group that increased the TN rate in the 2^{nd} role play. However, these differences were considered statistically insignificant according to the evaluation's authors [1].

Generally speaking, with user awareness approaches, the aim is to educate the user to be better aware of the technology which he or she is dealing with. Another way of looking at user training is that it follows a more "machine-centric" approach where we expect users to adapt to how the technology is functioning. On the other hand, enhancing software classifiers follows an opposite (human-centric) approach where the software is modified to filter unwanted phishing emails on behalf of the user.

Stefan Gorling [13] argues that security is (practically) a secondary goal, while productivity is the primary, and since users are already busy doing their own work (which could be a non-technical job) expecting them to get educated about technology (or phishing attacks) will not work as it will hit the maximum cognition limits of the users who are already busy with other tasks.

From the system usability perspective, in the article [14], Gerry Gaffney argues that there is no such thing as a "stupid user" but a bad system usability design instead, and the latter being the fault of the designer. The article says that if system users are stupid, then we should design systems accordingly for stupid users.

[25] http://www.wombatsecurity.com/

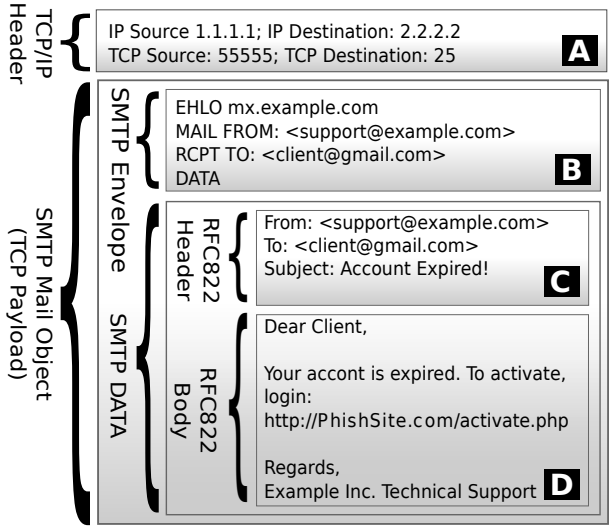

Fig. 9. Data diagram identifying data parts analyzed by email phishing detection techniques.

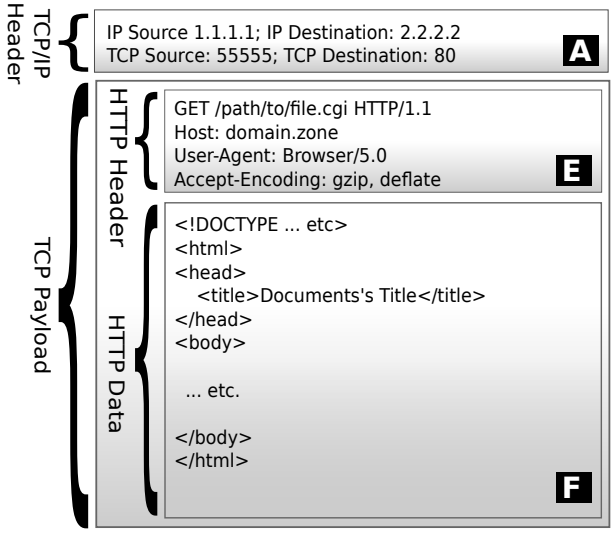

Fig. 10. Data diagram identifying data parts analyzed by HTTP website phishing detection techniques.

XII. Evaluations of Software Phishing Detection Techniques

Analyzed data parts by phishing detection techniques are depicted in Figures 9 and 10, which are used in Table III to compare the detection techniques as they are evaluated in the literature. This implies the use of different data sets, and that the results are not directly comparable. However, since the evaluation samples are taken from the same population (i.e. phishing and legitimate websites and emails in the Internet), the differences should not be significant. The rates FP and FN are measured as presented in Section V.

Table III presents generic approaches (e.g. blacklists, whitelists, heuristics, Machine Learning) used by the phishing detection, and their detection rates in terms of FP and FN. The table also indicates whether the presented techniques require access to resources over the Internet in order to function. It can be observed that the majority of the techniques do not rely on the resources over the Internet in order to perform classification decisions (six out of the twenty surveyed techniques require access to the Internet for performing such classification decisions). We believe that this is reflected due to the fact that accessing resources over the Internet can be expensive, and could form a potential bottleneck. It should be also noted that among the techniques that require Internet access include CIDS (which was not implemented [52]; also reflects the associated difficulty with such distributed system over the Internet), and two techniques that are related to Google [4], [34] (which is a scenario where accessing search engine rankings is not necessarily over the Internet since the results are cached and analyzed as part of Google's internal infrastructure).

Blacklists are able to achieve low FP rates. However, blacklists have been evaluated to be ineffective against zero-hour attacks, as they only detect 20% of phishing attacks at hour zero [25].

On the other hand, heuristics are effective against zero-hour attacks, however they need to be manually modified to adapt to future phishing trends and they tend to cause high FP rates. As presented in Table III, all techniques that used blacklists combined with heuristics, such as PhishNet and PhishWish have generally high FP rates. For example, PhishNet and PhishWish have 5% and 8.3% FP rates respectively, which is caused by the heuristic tests of the detection techniques respectively.

Similar to rule-based heuristics, Machine Learning-based classifiers are able to detect zero-hour phishing attacks in addition to superior adaption to future phishing characteristics. The adaption to changes in phishing trends can be achieved through reinforcement learning or simply repeating the learning phase periodically to construct a newer model with better adaption to current phishing attack characteristics.

The best performing anti-phishing email classifiers used Machine Learning techniques, namely those by Andre Bergholz et. al. [46] and Fergus Toolan et. al. [47]. Contrary to heuristics, Machine Learning techniques were also able to achieve low FP rates. For example, Andre Berghol'z model-based email classifier and Google's large scale website classifiers achieved 0% FP rates. Fergus Toolan's R-Boost achieved 1.3% FP rate, however it should be noted that the aim of the R-Boost technique is to minimize the FN rate only, which the technique reached by achieving an FN rate of 0%.

Natural Language Processing (NLP) techniques are rarely found in the phishing mitigation literature, which — to the best our knowledge — to be due to lack of sufficient maturity in NLP techniques with regards to correctly understanding the semantics of messages written in natural languages that also may contain typos. Email and web browsing are critical tasks and software techniques still finds it extremely challenging to correctly understand the semantics of natural languages. Moreover, many email messages can have typos, which further complicates the job of NLP techniques. An example of an NLP-based anti-phishing technique is EBDIS [15] with an FP rate of 1.9%, and an FN rate of 25% which is not as accurate as the competition. However, although NLP techniques —

alone — do not achieve competing FP and FN rates, their addition to other techniques (e.g. ML) can be promising.

Existing visual similarity-based phishing detection techniques heavily rely on use of blacklists or whitelists of website snapshot salient point descriptors. Conceptually, they are still blacklists or whitelists and require frequent updates. However, blacklists of website snapshot salient points are able to address zero-hour attacks, while URL blacklists can not. On the other hand, visual similarity-based detection techniques assume that phishing websites are always similar to the websites of targeted brands (e.g. PayPal phishing websites look similar to PayPal's legitimate website), which might not be always true. To assess their true effectiveness, it would be necessary to empirically measure the susceptibility of end-users to fall victims for phishing websites that are not very similar to their targeted brands. Until such demographic study is made, the effectiveness of visual similarity techniques might not be accurately known, but rather assumed.

TABLE III: Classification performance of phishing detection techniques as evaluated in the literature.

Detection Technique	Blacklists	Whitelists	Heuristics	Visual Similarity	Machine Learning	Network Communication	False Positives	False Negatives
PhishNet [28]	✓		✓				5%	3%
Comments: Expands blacklisted URLs (parents) and produces multiple variations of the same URL (children) via heuristics, thus it analyzes **E** data parts only. However, this can significantly increase the size of the expanded blacklist, which increase bandwidth demands. Reducing bandwidth demand also reduces the time needed to exchange the blacklists, which is one of the key objectives of Google Safe Browsing API [26]. Moreover, a FP rate of 5% is considered too high for practical daily use.								
AIWL [29]		✓	✓	✓			0%	0%
Comments: Classification accuracy is highly dependent on the individual user. If the user trains his/her own browser incorrectly, the whitelist will be constructed incorrectly too. For this reason, a number of other solutions (such as [4]) use a collaborative approach where a centralized model is constructed instead. Analyzes data parts **E** and **F**.								
SpoofGuard [30]			✓				38% [53]	9% [53]
Comments: Too high FP rate. Its advantage is being able to detect zero-hour phishing attacks. Analyzes data parts **E** and **F**								
CIDS [52]			✓			✓	–	–
Comments: CIDS was not evaluated nor implemented due to difficulties associated with reproducing double fast-flux bot networks. Analyzes data parts **A**, **B**, **C**, **D**, **E** and **F**. The technique requires data communication over the network (i.e. in order for the various IDS devices to collaborate), which can be a limiting performance factor of the system under certain network conditions.								
PhishGuard [32]			✓				–	–
Comments: Uses heuristic tests and can be effective against zero-hour attacks. However, no performance evaluation was presented in [32]. Analyzes data parts **E** and **F**.								
PhishCatch [54]			✓				1%	20%
Comments: The solution lacks easy adaption to phishing pattern changes as it is dependent on hard-coded rules that are the result of manual analysis of phishing emails by humans. Moreover, relative to other proposed works in the literature, a 20% FN rate is too high for the 1% FP. Analyzes data parts **E** and **F**.								
PhishWish [33]	✓		✓				8.3%	2.5%
Comments: High FP rate. Lacks adaptability as phishing patterns change. This is due to the fact that the proposed solution is hard-coded with the 11 rules. A solution to this issue might be periodical update of the rules, or use of ML technique to periodically construct classification models. Analyzes data parts **E** and **F**								
CANTINA [34]			✓			✓	3%	11%
Comments: The solution results in sending additional Hypertext Transfer Protocol (HTTP) GET requests to query search engines over the network, which adds delay in the browsing experience from the end-user perspective. An FP rate of 3% is generally considered too high by researchers, such as Colin Whittaker et. al. [4]. Analyzes data parts **E** and **F**.								
A phishing sites blacklist generator [36]			✓			✓	9%	0%

TABLE III: Classification performance of phishing detection techniques as evaluated in the literature.

Detection Technique	Blacklists	Whitelists	Heuristics	Visual Similarity	Machine Learning	Network Communication	False Positives	False Negatives
Comments: Although the FN rate is low, the FP rate is too high. However, the approach can be promising as an individual heuristic feature in a more complete classifier to reduce the FP rate. Analyzes data parts **E** and **F**. The technique requires data communication over the network (for the querying search engines), which can be a limiting performance factor of the system under certain network conditions.								
Fighting phishing with discriminative keypoint features of webpages [37]	✓			✓			$< 0.1\%$	$< 0.1\%$
Comments: The evaluation shows accurate detection rates. However, since whitelists are used, the FN rate could increase as new websites are visited. Another drawback is that the whitelists size can be large due to storing 48-dimensional CCH descriptors for each salient point (Harris corner) found in a websites snapshot, which may impose additional network load. Moreover, use of image processing on end-user web client software might introduce noticeable delay. Analyzes data parts **E** and **F**.								
Visual similarity-based phishing detection without victim site information [39]	✓	✓		✓			17.4%	8.3%
Comments: Too high FP rate. Use of image processing on end-user web client can impose a noticeable delay in the browsing experience. Analyzes data parts **E** and **F**.								
Automatic Detection of Phishing Target from Phishing Webpage [40]				✓	✓	✓	3.4%	8.56%
Comments: Uses a clustering technique (i.e. DBSCAN) to detect whether a page is similar to a protected page (e.g. login pages of highly targeted websites, such as PayPal's), based on the assumption that phishing pages mimic their targets. Analyzes data parts **E** and **F**. The technique requires data communication over the network (e.g. for querying search engines to find indirectly similar pages and their rank), which can be a limiting performance factor of the system under specific network conditions.								
Detecting DNS-poisoning-based phishing attacks from their network performance characteristics [41]					✓	✓	0.7%	0.6%
Comments: Detects DNS spoofing attacks based on the network performance characteristics of legitimate and phishing traffic. Analyzes the data part **A**. The advantages are that FP and FN rates are low, however it should be noted that the aim is to only detect DNS spoofing attacks (i.e. not all phishing attacks in general), which is implies that it should be used with other phishing detection techniques in order to enhance the overall detection. The technique requires access to the network (e.g. to measure round trip times), which can limit the performance of the system under busy network conditions.								
Textual and Visual Content-Based Anti-Phishing: A Bayesian Approach [42]	✓	✓	✓		✓		0-0.02%	0-1.95%

TABLE III: Classification performance of phishing detection techniques as evaluated in the literature.

Detection Technique	Blacklists	Whitelists	Heuristics	Visual Similarity	Machine Learning	Network Communication	False Positives	False Negatives
Comments: Uses an ensembling technique to combine visual and textual classification models by using a Bayesian model to detect whether a page is visually and textually similar to a protected page (e.g. login pages of highly targeted websites, such as PayPal's), based on the assumption that phishing pages mimic their targets. Analyzes data parts **E** and **F**..								
Large-scale automatic classification of pages [4]	✓	✓	✓		✓	✓	0% [53]	16-30% [53]
Comments: Google Safe Browsing API [26] is used to publish the results as a blacklist. However, since blacklists were used, the performance against zero-hour attacks is poor (i.e. it had the worst FN rate). Analyzes data parts **A**, **E** and **F**. The technique requires data communication over the network (e.g. for querying website ranks), this can affect the performance of the system under certain network conditions.								
B-APT [45]		✓			✓		3%	0%
Comments: Use of naive Bayesian filters is successful for spam classification, its use for phishing website classification can result in high level of classification errors. This is due to the fact that many phishing websites use the same words as their legitimate counterparts. For this reason, B-APT used an extensive whitelist to further minimize the FP rate. Although an FP rate of 3% is considered high for practical use, the actual B-APT's FP rate (without the extensive whitelist) is even higher. Analyzes the data part **F**.								
EBDIS [15]					✓		1.9%	25%
Comments: Too high FN rate. Use of OntoSem increased the run time execution. Analyzes the data part **F**.								
Detecting phishing emails using hybrid features [44]			✓		✓		–	–
Comments: The evaluation presented in [44] used 10-fold cross-validation and measured the classification model's performance using ACC only, which resulted in 99.27% (the average accuracy of all of the 10 folds). Although this number might seem good, it lacks enough details to know the TN and FN rates. This is specifically important to know as the evaluation data set was composed of 613,048 phishing and 46,525 legitimate emails. In other words, if ZeroR[26] is used as a classifier the prediction accuracy will be $613,048 \div (613,048 + 46,525) \times 100 = 92.95\%$. This raises the question how well does the proposed model perform given that ZeroR performs similarly on the same dataset, and to answer it we may need more performance evaluation metrics such as TN and FN rates.								
Model-based features [46]			✓		✓		0%	1%
Comments: According to the surveyed literature, this is the most accurate publicly known email phishing detection technique. Low FP and FN rates with the ability to detect zero-hour attacks (since no blacklists are used). However, use of image processing and model-based features may increase the run time and space complexity of the classifier. Analyzes data parts **C** and **D**.								
R-Boost [47]			✓		✓		1.4%	0%

[26]ZeroR is a simple classifier that ignores all features and predicts that every tested instance to belong to the class of the majority instances. For example, if the majority are legitimate emails, it will predict that all tested instances are legitimate. ZeroR is often used as the baseline to measure how well a classifier is performing.

TABLE III: Classification performance of phishing detection techniques as evaluated in the literature.

Detection Technique	Blacklists	Whitelists	Heuristics	Visual Similarity	Machine Learning	Network Communication	False Positives	False Negatives
Comments: According to the surveyed literature, this is the 2^{nd} most accurate publicly known email phishing detection technique by using only 5 simple heuristic features. However, its FP rate is somewhat high and its use of multiple classification models in its ensemble may also increase the run time and space complexity of the proposed solution (especially as one of the ensemble classifiers is a lazy learner; k-NN). Analyzes data parts **C** and **D**.								

XIII. Learned lessons

This section presents our view on the various phishing detection approaches that are presented in the survey.

A. User education and awareness

Since phishing is a social engineering attack, an obvious solution can be educating the end-user. However, as discussed in [13] education and user awareness — alone — are not enough, and that what is needed is regulating the behavior of end-users instead.

Various incidents, such as the one described in [2], have shown that even security providers themselves have fallen victims for phishing attacks. Although phishing seems to be a simple attack that exploits the naivety of end-users, it is able to persuade security-aware engineers as well.

This indicates the possibility that systems' complexities are raising beyond the cognition limits of many humans, and that simply educating them is not enough. A more promising solution could be enhancing the system usability via:

- Better user interfaces. It is visible to us that the software industry is moving towards this direction. For example, older versions of web browsers used passive warnings, while recent ones moved to active warnings. A security warning should be active (i.e. blocks the content) and should visually hint the user of risks even without reading its content (since most end-users do not read warning messages [23]).
- Enhancing the behavior of the systems, so that the harmful messages are automatically detected and quarantined on behalf of the end-user. For example, blacklists, heuristic rules and ML techniques can be used to automatically filter harmful content from end-users. Such features are currently implemented in web browsers, email clients and server-side filters.

The ideal phishing mitigation direction seems to us to be debatable so far, which could be due to the short history depth of information technology in general. However, in our opinion, systems are in reality moving towards adapting to their end-users (as opposed to having end-users adapting to their systems).

B. Blacklists

Blacklists are effective when minimal FP rates are required, which is achieved due to the way blacklists are constructed (e.g. many of them involve human administration, such as PhishTank).

Blacklists also have the advantage of requiring low resources on the host machine. This elevates the need of extensively analyzing the content of websites and emails.

However, blacklists are known to be behind the line when the objective is mitigation of zero-hour phishing attacks. It is critical to mitigate against zero-hour phishing attacks since most of the phishing campaigns are short-lived [25].

The primary reason behind blacklists inability to mitigate zero-hour phishing campaigns is due to the following factors:

- The time spent to blacklist a resource (e.g. phishing URLs) — a phishing campaign's URL or IP address should be detected first prior to its addition to the blacklist. Blacklist providers, such as PhishTank, rely on humans to vote on phishing URLs. According PhishTank's statistics[27], the median detection time is 2 hours. This is a significant delay as 63% of phishing campaigns end within the first 2 hours as well [25].
- The time spent to propagate an updated blacklist to end-user clients — For example, according to Google Safe Browsing API v2, the browser should synchronize its blacklist after 0–5 minutes (chosen randomly) after its startup, and then the browser should keep updating it periodically as specified by the blacklist server. This can leave a 5 minutes gap following the browser startup, and an arbitrary amount of time as specified by the blacklist server. This also includes delay caused by network and application service providers, specially with DNS-Based Blacklists (DNSBLs) as they send DNS queries for each suspect resource (e.g. IP address).

It should also be noted that DNSBLs can result in excessive queries with heavily loaded servers (e.g. email servers). The excessive DNS queries can in turn trigger flood mitigation techniques in intermediate DNS servers, which in turn can result in further delay.

Due to the limitations above, we suggest that blacklists — alone — should not be used whenever possible, and be combined with other techniques that can possibly mitigate zero-hour attacks. However, due to the low FP rate of blacklists, they can be combined with other techniques without causing significant raise in FP rate.

C. Heuristic tests and visual similarity

Unlike blacklists, heuristic tests are able to constantly detect phishing campaigns including zero-hour attacks [25]. However, they tend to have higher FP rates [25] than blacklists, which — in our opinion — is caused due to the following:

- The complicated nature of adversarial attacks makes it hard for humans to manually construct heuristic tests that mitigate the attacks without causing FP.
- Techniques used by phishing campaigns evolve with time, and thus heuristic tests require to be continuously updated. Although this periodic update is less frequent than blacklists', it is more expensive as it requires classifier designers to analyze the data and generalize tests that mitigate the phishing attacks.

Heuristic tests address the zero-hour gap that is left by blacklists, however they often increase FP rates. This raise in FP is undesired as it can hinder end-user productivity.

Techniques that rely on visual similarity in the literature achieved similar characteristics (i.e. higher FP rates than blacklists). However, they also require taking snapshots of how websites visually appear in web browsers, and storing the snapshots or the descriptors of extracted salient points in the

[27] https://www.phishtank.com/stats

snapshots. This can raise the computational cost of techniques that rely on visual similarities for a number of reasons:

- In order to take a snapshot of an analyzed website, its content should be rendered first (i.e. similar to Internet Explorer and FireFox). This requires parsing the HTML, Cascading Style Sheets (CSS), and JavaScript codes, as well as Flash and Java objects, which will add an arbitrary amount of processing time depending on the website's content.
- Storing information that describes images can be expensive. For example, storing snapshots (or descriptors of snapshots) of websites may require more space than simply storing their URL, which is the case with techniques that are presented in this survey.

According to the surveyed literature, heuristics and visual similarity tests tend to have higher computational cost, associated with higher FP rates than blacklists. However, they could prove to be effective additions in larger classifiers. For example, a classifier that takes advantage of various techniques as its input to output a more reliable classification label.

D. Machine Learning-based classifiers

Similar to heuristic tests, ML-based techniques can mitigate zero-hour phishing attacks, which makes them advantageous when compared with blacklists. Interestingly, ML techniques are also capable of constructing their own classification models by analyzing large sets of data. This elevates the need of manually creating heuristic tests as ML algorithms are able to find their own models. In other words, ML techniques have the following advantages over heuristic tests:

- Despite the complicated nature of adversarial attacks, it is possible to construct effective classification models when large data set samples are available, without the need of manually analyzing data to discover complex relationships.
- As phishing campaigns evolve, ML classifiers can automatically evolve via *reinforcement learning*. Alternatively, it is also possible to periodically construct newer classification models by simply retraining the learner with updated sample data sets.

ML-based anti-phishing classifiers in the literature, such as those presented in [4], [46], have shown that it is possible to achieve less than 1% FP, and more than 99% TP rates. According to the surveyed literature, ML-based classifiers are the only classifiers that achieved such high classification accuracy while maintaining their ability to detect zero-hour phishing attacks.

XIV. Conclusion

User education or training is an attempt to increase the technical awareness level of users to reduce their susceptibility to phishing attacks.

It is generally assumed that the addition of user education materials compliments technical solutions (e.g. classifiers). However, the human factor is broad and education alone may not guarantee a positive behavioral response.

As shown in the previous sections, most of the educational materials were also associated with a decrease in the TN rate, with an exception of only one educational material, namely: *Anti-Phish Phil*. This shows that the addition of user training approaches is not *always* the right answer.

Essentially, this is an area of debate, and experts argue that user education might not be the answer [13], [14].

User education materials can complement software solutions. However it should also be noted that none of the existing studies empirically show enough evidence that user education can practically complement software solutions. This is due to the fact that all of the publicly available user education studies have evaluated educational materials independently from software solutions.

The study in [1] concludes that *Anti-Phish Phil* training material reduced FN rate from 46% to 29%, which is not enough evidence to assume that it would also complement software solutions that, for example, achieve a FN rate of less than 1%. The un-answered question is: what is the percentage of overlap between the classification performed by end-users following a user training phase, and the classification performed by a software classifier? If the overlap is 100%, then the addition of user training can be redundant and will not be worth the added cost and complexity. However, if the overlap is less than 100%, then they can be complementary to each other — however, such a study is not available in the public literature.

This survey reviewed a number of anti-phishing software techniques. Some of the important aspects in measuring phishing solutions are:

- Detection accuracy with regards to zero-hour phishing attacks. This is due to the fact that phishing websites are mostly short-lived and detection at hour zero is critical.
- Low false positives. A system with high false positives might cause more harm than good. Moreover, end-users will get into the habit of ignoring security warnings if the classifier is often mistaken.

Generally, software detection solutions are:

- Blacklists.
- Rule-based heuristics.
- Visual similarity.
- Machine Learning-based classifiers.

The findings in Section XII show that the use of Machine Learning techniques is promising as they have led to the most effective phishing classifiers in the publicly known literature. The Machine Learning-based detection techniques achieved high classification accuracy for analyzing similar data parts to those of rule-based heuristic techniques.

As a future work in this field, it would be beneficial to conduct a study that:

- measures the effect of the addition of user training from the perspective of correcting software classification mistakes.
- analyzes the phishing detection techniques from the perspective of their computational cost and energy consumption.

ACKNOWLEDGMENT

This research is partially supported by Buhooth[28]. The authors would also like to thank the anonymous reviewers for their valuable comments.

Mahmoud Khonji Held network and system engineering positions in the industry prior to joining Khalifa University where he received MSc by Research in mitigation of phishing attacks, in 2012. He received Bachelors in Computer Networking with distinction with highest honor from Higher Colleges of Technology, UAE, and an appreciation certificate for academic excellence. His research interests include: security usability, spam classification, authorship detection, machine learning and computer networking.

Dr. Youssef Iraqi received his M.Sc. and Ph.D. degrees in Computer Science from the University of Montreal, Canada, in 2000 and 2003 respectively. He is currently an Associate Professor in the Department of Computer Engineering at Khalifa University, UAE. Before joining Khalifa University, Dr. Iraqi was the chair of the Computer Science Department at Dhofar University, Oman for four years. From 2004 to 2005 he was a Research Assistant Professor in the School of Computer Science at the University of Waterloo, Canada. Before that he was a Research Assistant at the Computer Science Research Institute of Montreal, Canada. Dr. Iraqi has published more than 70 research papers in international journals and refereed conference proceedings. In 2008, he received the IEEE Communications Society Fred W. Ellersick prize. Dr. Iraqi is a senior member of the IEEE.

Dr. Andrew Jones After 25 years service with the British Army's Intelligence Corps during which he was awarded an MBE, he became a manager and a researcher and analyst in the area of Information Warfare and computer crime at a defence research establishment. In 2002, he left the defense environment to take up a post as a principal lecturer at the University of Glamorgan in the subjects of Network Security and Computer Crime and as a researcher on the Threats to Information Systems and Computer Forensics. He developed and managed a well equipped Computer Forensics Laboratory and took the lead on a large number of computer investigations and data recovery tasks.

In January 2005 he joined the Security Research Center at BT where he became a Chief Researcher and the head of information security research. During his time at BT he managed a number of research projects and led a series of projects into residual data on second hand media. He is currently the Program Chair for the MSc. in Information Security at Khalifa University of Science, Technology and Research in Sharjah, UAE, and holds posts as an adjunct professor at Edith Cowan University in Perth and the University of South Australia in Adelaide, Australia. He holds a Ph.D. in the area of threats to information systems. He has written five books on topics including Information Warfare, Risk management and Digital forensics and Cyber Crime and is currently writing a book on digital forensic procedures and practices.

[28]http://www.buhooth.ae/

APPENDIX

Table IV lists symbols and their meanings as used in this survey.

TABLE IV
SYMBOLS LIST

Symbol	Description
$N_{P\to P}$	Number of phishing instances that are classified as phishing
$N_{P\to L}$	Number of phishing instances that are classified as legitimate
$N_{L\to L}$	Number of legitimate instances that are classified as legitimate
$N_{L\to P}$	Number of legitimate instances that are classified as phishing
TP	True Positive rate
FP	False Positive rate
TN	True Negative rate
FN	False Negative rate
R	Recall rate
P	Precision rate
f_1	f_1 score, the harmonic mean of P and R
ACC	Classification accuracy
W_{Err}	Weight error rate according to weight λ
TF_{ij}	Term Frequency for for term i in document j
$IDFi$	Inverse Document Frequency for term i
$TFIDF_{ij}$	TF-IDF for term i in document j
$CCH(p_c)$	CCH descriptor for salient point p_c in an image
W_s	Suspect website
W_L	A set of legitimate websites
W_P	A set of phishing websites
W_U	A set of websites with unknown classes
W_D	A set of websites that are directly linked to W_s
W_I	A set of websites that are indirectly linked to W_s
K_S	A set of keywords that are extracted from W_s
K_j	A set of keywords that are extracted from website j
V_s	A features vector that represents W_s

REFERENCES

[1] S. Sheng, M. Holbrook, P. Kumaraguru, L. F. Cranor, and J. Downs, "Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions," in *Proceedings of the 28th international conference on Human factors in computing systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 373–382.

[2] B. Krebs, "HBGary Federal hacked by Anonymous," http://krebsonsecurity.com/2011/02/hbgary-federal-hacked-by-anonymous/, 2011, accessed December 2011.

[3] B. Schneier, "Lockheed Martin hack linked to RSA's SecurID breach," http://www.schneier.com/blog/archives/2011/05/lockheed_martin.html, 2011, accessed December 2011.

[4] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in *NDSS '10*, 2010.

[5] X. Dong, J. Clark, and J. Jacob, "Modelling user-phishing interaction," in *Human System Interactions, 2008 Conference on*, may 2008, pp. 627 –632.

[6] W. D. Yu, S. Nargundkar, and N. Tiruthani, "A phishing vulnerability analysis of web based systems," in *Proceedings of the 13th IEEE Symposium on Computers and Communications (ISCC 2008)*. Marrakech, Morocco: IEEE, July 2008, pp. 326–331.

[7] Anti-Phishing Working Group (APWG), "Phishing activity trends report — second half 2010," http://apwg.org/reports/apwg_report_h2_2010.pdf, 2010, accessed December 2011.

[8] Anti-Phishing Working Group (APWG), "Phishing activity trends report — first half 2011," http://apwg.org/reports/apwg_trends_report_h1_2011.pdf, 2011, accessed December 2011.

[9] Anti-Phishing Working Group (APWG), "Phishing activity trends report — second half 2011," http://apwg.org/reports/apwg_trends_report_h2_2011.pdf, 2011, accessed July 2012.

[10] B. Schneier, "Details of the RSA hack," http://www.schneier.com/blog/archives/2011/08/details_of_the.html, 2011, accessed December 2011.

[11] P. Kumaraguru, Y. Rhee, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge, "Protecting people from phishing: the design and evaluation of an embedded training email system," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ser. CHI '07. New York, NY, USA: ACM, 2007, pp. 905–914.

[12] A. Alnajim and M. Munro, "An anti-phishing approach that uses training intervention for phishing websites detection," in *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 405–410.

[13] S. Gorling, "The Myth of User Education," *Proceedings of the 16th Virus Bulletin International Conference*, 2006.

[14] G. Gaffney, "The myth of the stupid user," http://www.infodesign.com.au/articles/themythofthestupiduser, accessed March 2011.

[15] A. Stone, "Natural-language processing for intrusion detection," *Computer*, vol. 40, no. 12, pp. 103 –105, dec. 2007.

[16] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," in *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, ser. eCrime '07. New York, NY, USA: ACM, 2007, pp. 60–69.

[17] C. Yue and H. Wang, "Anti-phishing in offense and defense," in *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, 8-12 2008, pp. 345 –354.

[18] P. Knickerbocker, D. Yu, and J. Li, "Humboldt: A distributed phishing disruption system," in *eCrime Researchers Summit*, 2009, pp. 1–12.

[19] L. James, *Phishing Exposed*. Syngress Publishing, 2005.

[20] J. S. Downs, M. Holbrook, and L. F. Cranor, "Behavioral response to phishing risk," in *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, ser. eCrime '07. New York, NY, USA: ACM, 2007, pp. 37–44.

[21] H. Huang, J. Tan, and L. Liu, "Countermeasure techniques for deceptive phishing attack," in *International Conference on New Trends in Information and Service Science, 2009. NISS '09*, 2009, pp. 636 – 641.

[22] T. Moore and R. Clayton, "Examining the impact of website take-down on phishing," in *eCrime '07: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*. New York, NY, USA: ACM, 2007, pp. 1–13.

[23] S. Egelman, L. F. Cranor, and J. Hong, "You've been warned: an empirical study of the effectiveness of web browser phishing warnings," in *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 1065–1074.

[24] M. Wu, R. C. Miller, and S. L. Garfinkel, "Do security toolbars actually prevent phishing attacks?" in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, ser. CHI '06, New York, NY, USA, 2006, pp. 601–610.

[25] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in *Proceedings of the 6th Conference in Email and Anti-Spam*, ser. CEAS'09, Mountain view, CA, July 2009.

[26] Google, "Google safe browsing API," http://code.google.com/apis/safebrowsing/, accessed Oct 2011.

[27] Google, "Protocolv2Spec," http://code.google.com/p/google-safe-browsing/wiki/Protocolv2Spec, accessed Oct 2011.

[28] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks," in *INFOCOM'10: Proceedings of the 29th conference on Information communications*. Piscataway, NJ, USA: IEEE Press, 2010, pp. 346–350.

[29] Y. Cao, W. Han, and Y. Le, "Anti-phishing based on automated individual white-list," in *DIM '08: Proceedings of the 4th ACM workshop on Digital identity management*. New York, NY, USA: ACM, 2008, pp. 51–60.

[30] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell, "Client-side defense against web-based identity theft," in *NDSS*. The Internet Society, 2004.

[31] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and Detecting Fast-Flux Service Networks," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2008.

[32] P. Likarish, D. Dunbar, and T. E. Hansen, "Phishguard: A browser plug-in for protection from phishing," in 2^{nd} *International Conference on Internet Multimedia Services Architecture and Applications, 2008. IMSAA 2008*, 2008, pp. 1 – 6.

[33] D. L. Cook, V. K. Gurbani, and M. Daniluk, "Phishwish: A stateless phishing filter using minimal rules," in *Financial Cryptography and Data Security*, G. Tsudik, Ed. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 182–186.

[34] 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*, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 639–648.

[35] T. A. Phelps and R. Wilensky, "Robust Hyperlinks and Locations," *D-Lib Magazine*, vol. 6, no. 7/8, Jul. 2000.

[36] M. Sharifi and S. H. Siadati, "A phishing sites blacklist generator," in *Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications*, ser. AICCSA '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 840–843.

[37] K.-T. Chen, J.-Y. Chen, C.-R. Huang, and C.-S. Chen, "Fighting phishing with discriminative keypoint features," *Internet Computing, IEEE*, vol. 13, no. 3, pp. 56 –63, may-june 2009.

[38] C.-R. Huang, C.-S. Chen, and P.-C. Chung, "Contrast context histogram-an efficient discriminating local descriptor for object recognition and image matching," *Pattern Recogn.*, vol. 41, pp. 3071–3077, October 2008.

[39] M. Hara, A. Yamada, and Y. Miyake, "Visual similarity-based phishing detection without victim site information," in *IEEE Symposium on Computational Intelligence in Cyber Security, 2009. CICS '09*, 2009, pp. 30 – 36.

[40] G. Liu, B. Qiu, and L. Wenyin, "Automatic detection of phishing target from phishing webpage," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, aug. 2010, pp. 4153 –4156.

[41] H. Kim and J. Huh, "Detecting dns-poisoning-based phishing attacks from their network performance characteristics," *Electronics Letters*, vol. 47, no. 11, pp. 656 –658, 26 2011.

[42] H. Zhang, G. Liu, T. Chow, and W. Liu, "Textual and visual content-based anti-phishing: A bayesian approach," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1532 –1546, oct. 2011.

[43] A. Y. Fu, L. Wenyin, and X. Deng, "Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd)," *IEEE Trans. Dependable Secur. Comput.*, vol. 3, no. 4, pp. 301–311, Oct. 2006.

[44] L. Ma, B. Ofoghi, P. Watters, and S. Brown, "Detecting phishing emails using hybrid features," in *Proceedings of the 2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, ser. UIC-ATC '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 493–497.

[45] P. Likarish, D. Dunbar, and T. E. Hansen, "B-apt: Bayesian anti-phishing toolbar," in *IEEE International Conference on Communications, 2008. ICC '08*, 2008, pp. 1745 – 1749.

[46] A. Bergholz, J. De Beer, S. Glahn, M.-F. Moens, G. Paaß, and S. Strobel, "New filtering approaches for phishing email," *J. Comput. Secur.*, vol. 18, pp. 7–35, January 2010.

[47] F. Toolan and J. Carthy, "Phishing detection using classifier ensembles," in *eCrime Researchers Summit, 2009. eCRIME '09.*, 20 2009-oct. 21 2009, pp. 1 –9.

[48] "How to recognize phishing email messages or links," http://www.microsoft.com/security/online-privacy/phishing-symptoms.aspx, accessed March 2011.

[49] "How not to get hooked by a phishing scam," http://www.onguardonline.gov/topics/phishing.aspx, accessed March 2011.

[50] "Avoiding getting hooked by phishers," http://www.fraud.org/tips/internet/phishing.htm, accessed March 2011.

[51] S. Sheng, B. Magnien, P. Kumaraguru, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge, "Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish," in *Proceedings of the 3rd symposium on Usable privacy and security*, ser. SOUPS '07. New York, NY, USA: ACM, 2007, pp. 88–99.

[52] C. V. Zhou, C. Leckie, S. Karunasekera, and T. Peng, "A self-healing, self-protecting collaborative intrusion detection architecture to trace-back fast-flux phishing domains," in *NOMS Workshops 2008. IEEE Network Operations and Management Symposium Workshops, 2008*, 2008, pp. 321 – 327.

[53] L. Cranor, S. Egelman, J. Hong, and Y. Zhang, "Phinding phish: An evaluation of anti-phishing toolbars," www.cylab.cmu.edu/files/cmucylab06018.pdf, 2006, accessed Oct 2011.

[54] W. D. Yu, S. Nargundkar, and N. Tiruthani, "Phishcatch – a phishing detection tool," in 33^{rd} *Annual IEEE International on Computer Software and Applications Conference, 2009. COMPSAC '09*, 2009, pp. 451 – 456.