Mitigating Adversarial Gray-Box Attacks Against Phishing Detectors

Giovanni Apruzzese*, V. S. Subrahmanian†
*University of Liechtenstein, Hilti Chair of Data and Application Security—Vaduz, Liechtenstein
giovanni.apruzzese@uni.li
†Northwestern University, Dept. of Computer Science & Buffett Institute of Global Affairs—Evanston, IL, USA
vss@northwestern.edu

***Abstract*—Although machine learning based algorithms have been extensively used for detecting phishing websites, there has been relatively little work on how adversaries may attack such "phishing detectors" (PDs for short). In this paper, we propose a set of Gray-Box attacks on PDs that an adversary may use which vary depending on the knowledge that he has about the PD. We show that these attacks severely degrade the effectiveness of several existing PDs. We then propose the concept of *operation chains* that iteratively map an original set of features to a new set of features and develop the "Protective Operation Chain" (POC for short) algorithm. POC leverages the combination of random feature selection and feature mappings in order to increase the attacker's uncertainty about the target PD. Using 3 existing publicly available datasets plus a fourth that we have created and will release upon the publication of this paper[1], we show that POC is more robust to these attacks than past competing work, while preserving predictive performance when no adversarial attacks are present. Moreover, POC is robust to attacks on 13 different classifiers, not just one. These results are shown to be statistically significant at the $p < 0.001$ level.**

***Index Terms*—phishing detection, cybersecurity, adversarial attacks**

I. Introduction

MACHINE learning algorithms are increasingly used in a wide array of cybersecurity applications including malware detection [1], intrusion detection [2], insider threat detection [3], spam detection [4], and the detection of phishing websites [5].

Phishing attacks are one of the most common types of attacks. ProofPoint's 2020 "State of the Phish" report[2] states that over 1.5 million phishing websites are created *every month* and that 90% of businesses reported being a victim of a phishing attack in 2019. Phishing attacks offer one of the easiest ways for malicious hackers to penetrate an enterprise. Considerable work has gone into addressing this problem [5]–[15]. In addition to blacklists maintained by corporations such as Google, there are also publicly available blacklists from sites such as PhishTank[3]. However, these sources become obsolete frequently as malicious hackers move their phishing URLs from site to site in order to evade detection. Rule-based systems were therefore developed by several researchers. For instance, [16] develops a set of 8 rules to capture phishing webpages while other approaches analyze the content of a website [17]. Another effort [18] has examined the use of a discriminative set of features associated with phishing URLs and then checked to see whether a given URL was similar to a known phishing URL based on their associated feature vectors. [19], [20] pioneered the idea of using visual similarity between a legitimate web page (e.g. a bank website) and another website in order to check if the latter might be a phishing website. During the past decade, using machine learning to detect phishing has become widespread. Early efforts in this direction include [21]–[27]. Despite even more recent research efforts proposing increasingly sophisticated machine learning solutions to counter this threat [5]–[14], phishing websites still represent a dangerous menace [28] as is evident from the aforementioned ProofPoint report.

One reason for this is that machine learning classifiers are trained on a training dataset from which they learn a model that separates benign entries from illegitimate ones. However, adversaries (i.e. malicious hackers) are continuously adapting to Phishing Detectors (PDs for short). Often times, these adaptations are very simple, allowing their phishing webpages to evade existing PDs with relative ease. Most work on adversarial machine learning for cyber-security deals with two extremes: "white box" attacks in which the adversary has full knowledge of the defenses used by the PD (e.g. classifier

[1]We provide a sample of our dataset for the referees. We release our resources at: https://lnu-phish.github.io/

[2]https://proofpoint.com/us/resources/threat-reports/state-of-phish

[3]https://www.phishtank.org/

used, list of features used) or "black box" attacks in which the adversary has no knowledge whatsoever. Real world attackers are not likely to have knowledge that is at either of these extremes — their knowledge is most likely somewhere in the "gray area" between full knowledge (white box) and no knowledge (black box) of the defenses. We use the term "Gray Box" to refer to attacks where the attacker's knowledge can lie between these two extremes.

We propose a more robust phishing website detector that is capable of withstanding a large class of Gray Box attacks.[4] Our notion of Gray Box is powerful enough to capture both White Box and Black Box settings. In particular, we make the following contributions.

- *Gray Box Attack Scenarios.* We consider two types of Gray Box attacks: *simple* attacks, where the adversary only knows and modifies a few features and *complex* attacks, where the adversary knows a percentage Δ of the set of features used by the defender. When $\Delta = 100\%$, we have a White Box attack. When $\Delta = 0\%$, we have a Black Box attack.
- We show that phishing detectors based on **13** recent works [5], [7]–[11], [13], [14], [29]–[33] during the 2014–2019 timeframe are susceptible to these attacks. Specifically, we formally define the *Impact* of an attack on a phishing detector and show that these attacks induce a statistically significant impact (performance reduction) against these 13 well-known classifiers, including 2 deep learning classifiers. Thus, our Gray Box attacks encompass the above 2 attack scenarios and apply to 13 classifiers (as opposed to just one as in most past work).[5] The impact of these Gray Box attacks is shown on 3 well-known datasets (DeltaPhish [34], Mendeley [35] and UCI [36]) as well as a fourth (new) dataset that we have created.
- We define the notion of an *operation chain* (*oc*). Operation chains transform existing samples into a new feature space through the iterative application of some simple operators. Even if the adversary knew the original feature set, he is unlikely to know such new feature space[6]. We propose the *Protective Operation Chain* (POC) algorithm.
- We show that POC is more robust to these attacks than 13 existing PDs on the same 4 datasets mentioned above. Past works use a wide range of classification techniques and features—so POC is robust when used on top of many different classification algorithms and feature sets. To validate the claim that POC outperforms existing baselines, we carry out a very rigorous Wilcoxon Signed Rank test (which imposes tougher metrics than, e.g., simple t-test) along with a Bonferroni correction. Such test demonstrates that POC increases the robustness of past baselines at the $p < 0.001$ level, i.e. the probability that POC really outperforms past work (as opposed to doing so by accident) is over 99.9%.[7]
- We show that POC is practical for real deployments. We show that the using POC causes a statistically negligible performance degradation in the *absence* of attacks (with respect to the baselines). We conduct an in-depth analysis showing the pros and cons of POC when used to harden the best baseline PD for each dataset.
- A final contribution is our new dataset, LNU-Phish (short for Liechtenstein and Northwestern University-Phishing). LNU-Phish overcomes several problems affecting existing datasets which we discuss shortly and can serve as a "future-proof" benchmark for developing novel PDs. We will release both LNU-Phish and the code to compute our POC implementation after paper acceptance.

The remainder of this paper is structured as follows. Section II presents related work. Section III motivates our LNU-Phish dataset and explains how it was built. Section IV outlines the adversarial Gray Box attacks proposed in this paper. The description of operation chains and the POC algorithm is provided in Section V. Section VI shows our main experimental results, which are formally analyzed and discussed in Section VII. Section VIII provides additional experiments on a special application of POC. Section IX concludes the paper and suggests avenues for future work.

II. Related work

We divide related work into 3 parts: (i) detection of phishing websites; (ii) vulnerability of ML to adversarial attacks and existing countermeasures; (iii) adversarial attacks against phishing detectors (PD).

A. Detection of Phishing Websites

Though rule based methods (e.g. [6]) were initial used to detect phishing sites, machine learning (ML)

[4]Please note that we do NOT claim robustness to all types of Gray Box attacks, but only to certain classes defined in the paper.

[5]Like past work in the area, we do not handle attacks on custom classifiers that are often kept secret.

[6]Unless of course he has already compromised the enterprise system, but in that case, he would not need to phish employees of the enterprise!

[7]We note that $p < 0.05$ is the common standard for a one-star claim of statistical significance, $p < 0.01$ is the standard used for a 2-star claim, and $p < 0.001$ is the standard used for a 3 star claim. Our results put POC well within this highest statistical significance category.

based approaches are now common. [7], [10], [37], [38] identify phishing URLs by analysing hundreds of features extracted from the corresponding URLs. Other approaches [8], [10], [11], [29], [33] develop classifiers that use a reduced number of URL-based features while achieving similar or superior accuracy (e.g., over 97% in [33]). [39] suggests using information obtained by external sources (e.g., DNS logs) as features. Some papers combine URL-based features with HTML-based features to improve performance[8] [5], [9], [12], [14], [31]. [32] and [13] also consider information provided by external reputation sources (such as DNS records). More recently, [30], [40] leverage image processing with HTML inspection to detect phishing content in compromised websites. [15] develops methods to predict how Twitter is used to lure victims to phishing sites. Despite all these efforts, phishing websites still represent a widespread menace [28].

B. Adversarial Machine Learning

The recent success of deep learning has led to work showing that small perturbations to the input can lead to huge errors in image processing [41]–[45] as well as in text and speech processing [46], [47]. There has also been important work in general cybersecurity [48]–[52].

However, most existing work on adversarial ML makes very strong assumptions about the adversary's knowledge about the defense. Such knowledge is typically denoted with the notion of 'box' threat models. White box models assume the attacker has complete knowledge of the defense including the algorithm used and all the features used [53]–[56]. Conversely, black box models [57]–[61] assume the adversary knows absolutely nothing about the target's defenses. Both of these are extreme cases—in the real-world, defenders might use 'customized' classifiers (e.g. ensembles) with novel features and feature selection and late fusion [62] or custom combinations of supervised and unsupervised learning [63] which would be almost impossible for an adversary to guess correctly.

Some recent work considers other scenarios [64]–[66], but assume the classifier used by the PD is known which is unlikely [67], [68]. Some papers [56], [69] propose methods to harden detectors based on Neural Networks, while [70] proposes an approach to improve the robustness of tree-based mechanisms; other efforts focus on SVM-based techniques [71]–[73].

Our POC algorithm improves upon past work in the following ways: (i) we are the first to propose mapping feature vectors into a new feature space for purposes of increasing robustness to adversarial attacks against phishing detectors[9], (ii) POC is experimentally shown to be robust against attacks on 13 different classifiers as opposed to just one, (iii) POC is robust to different variants of Gray Box attacks, and (iv) we carry out our evaluation on 4 different datasets—not just 1.

[8]Because the word "accuracy" has a specific technical meaning in machine learning, we will use the term "performance" to refer to the quality of results generated by a classifier.

C. Adversarial Attacks Against Phishing Detectors

Most work on adversarial ML in cybersecurity has focused on malware detection [1], spam detection [4], [72]–[74] and network intrusions [2], [75]. An important recent effort [76] reverse engineers and subverts the phishing detector used by the Google Chrome web browser. Another important paper [30] devises a PD by combining the analysis of the webpage HTML and its image data in a white box setting where the attacker has complete control of the entire webpage domain.

Both of these efforts, though very important, make strong assumptions: [76] attacks just *one* phishing detector (albeit an important one); whereas [30] only assumes white box adversary that attacks only one classifier (linear SVM). In contrast, POC hardens multiple classifiers used by a defender and can protect against multiple attack models. Furthermore, POC is robust to attacks against 13 different classifiers including recent ones (e.g., Google's Deep & Wide), as opposed to just one classifier considered in most previous works. Finally, as mentioned earlier, POC's performance is tested on 4 datasets, not just one, using distinct (but similar) feature sets.

III. The LNU-Phish Dataset

There are a number of well-known existing datasets for phishing website detection. They include Mendeley [35], DeltaPhish [34], UCI [36], PhishStorm [77] and Ebbu[10].

A. Problems with Existing Phishing Datasets.

Most existing phishing datasets have one or more major problems that hamper replication by researchers.

1) *Dead.* Many phishing datasets contain lists of URLs. However many of these URLs are no longer functional which means that it is impossible to derive features and/or analyze the webpages associated with those URLs today. Examples of such datasets are PhishStorm and Ebbu.
2) *Feature Only.* Some datasets have the opposite problem: they include feature vectors associated

[9]Note that mapping feature spaces into new feature spaces is not new in other domains — for example, kernel tricks used in Support Vector Machines leverage a similar strategy.

[10]https://github.com/ebubekirbbr/pdd

with some websites but do not explicitly list those websites themselves. This means that defining new features and extracting them from the original webpage is impossible today. This is the problem with the Mendeley and UCI datasets.

3) *Non-Uniform Feature Sets.* Different datasets offer different sets of features. Having a uniform set of features across multiple datasets is necessary for fair evaluation of different PDs across different datasets—yet this turns out to be very difficult.
4) *Non-Replicability.* Some past efforts (e.g., [12], [13], [32]) use public lists such as PhishTank or Alexa Top1Million[11] to build custom datasets. However, these lists are updated very often, hence it is not possible to replicate the exact same data used in those studies for comparison purposes.

Simply put, the four problems listed above make it impossible to test new approaches across these diverse datasets and show if/when the new approaches outperform the old ones. The major existing datasets either disclose the original URLs which are not active anymore, or they do not disclose the URLs in which case follow on research does not know where to look. In neither case can different features (even from existing papers) be added, let alone new features invented by follow on researchers.

B. Solution: LNU-Phish

We address these issues by creating a new dataset, called LNU-Phish (short for Liechtenstein and Northwestern University). With over 23 000 samples, LNU-Phish is one of the biggest labeled datasets for phishing detection (the only bigger labeled datasets we know of are PhishStorm and Ebbu which suffer from the issues mentioned above). A comparison of LNU-Phish with other datasets is given in Table I.

LNU-Phish is a *large* and *fixed* dataset that can be used for future work on PDs. It contains complete information on each sample, such as the *URL*, the *DNS* records, as well as the underlying *HTML* code and a *screenshot* of the webpage. In addition, the dataset includes the features listed in Table II. Hence, even if the webpage is taken down, the data in LNU-Phish captures all information for reproducible future researchers; such information can also be augmented by, e.g., creating novel feature sets.

C. Creation Workflow of LNU-Phish

We collected benign samples from the Alexa Top-1million list and malicious samples from the well-known PhishTank and OpenPhish[12] repositories. All the entries were retrieved in March 2019.

To create a balanced corpus of benign websites, we divided the Alexa Top-1million list into three parts: the "top" partition includes websites from rank 1 to 10 000; the "middle" partition includes websites ranked from 10 001 to 100 000; the "bottom" partition includes all websites ranked below 100 001. We extract $\sim$ 5 000 websites from each partition. Our scripts visited each URL and saved the corresponding HTML as well as the full image representation of the homepage. We also queried and stored information provided by public DNSs for each URL. To populate the phishing entries, we followed a procedure similar to that in [40]. We monitored the PhishTank and OpenPhish sources for 3 weeks. Whenever a new phishing URL was added to these lists, we visited it and—if available—-saved the HTML, the screenshot of the landing webpage, and the information provided by the DNS query.[13]

The resulting 15 773 benign and 7 861 phishing samples represent the proposed LNU-Phish dataset.

Table I: Comparison of LNU-Phish with existing static datasets.

Name	Date	Phishing samples	Legit samples	URL data	HTML data	Reputation data	Screenshot data	Features
PhishLoad	2012	3 510	8 190	✓	✓	✗	✗	✗
UCI	2015	6 050	3 950	✗	✗	✗	✗	✓
DeltaPhish	2017	1 200	4 800	✓	✓	✗	✓	✗
Ebbu	2017	37 175	36 400	✓	✗	✗	✗	✗
PhishStorm	2014	48 000	48 000	✓	✗	✗	✗	✗
Mendeley	2018	5 000	5 000	✗	✗	✗	✗	✓
LNU-Phish	2020	7 861	15 773	✓	✓	✓	✓	✓

D. LNU-Phish Dataset Features

We compute the features (summarized in Table II) for each sample in our LNU-Phish dataset. The features are computed through the methodology in [7] and [29]. We focus on these features because they share many similarities with existing datasets and because they are used by several related efforts [8], [10]–[13], [29], [32]. Using similar feature sets allows a more fair comparison of PDs devised over different datasets.[14]

Nevertheless, the information provided in our LNU-Phish dataset allows the creation of any feature set for future works. In particular, we observe that ML-based detection systems must be periodically updated with recent data to avoid concept-drift problems [40], [78]. To further facilitate such 'updates', we will release our source code to compute the features of LNU-Phish, so

[11] https://www.alexa.com/topsites

[12] www.openphish.com

[13] The LNU-Phish dataset will be publicly released upon publication of the paper.

[14] We are aware that other studies adopt more features (e.g., [7], [37]), but the non-reproducibility of the datasets used to validate their PDs does not allow us to compare our paper with their work.

that future researchers can 'expand' it with more recent data by maintaining a uniform feature set.

Table II: List of features included the LNU-Phish dataset.

URL-features	*REP-features*	*HTML-features*
IP address	SSL final state	SFH
'@' (at) symbol	URL/DNS mismatch	Anchors
'-' (dash) symbol	DNS Record	Favicon
Dots number	Domain Age	iFrame
Fake HTTPS	PageRank	MailForm
URL Length	PortStatus	Pop-Up
Redirect	Redirections	RightClick
Shortener		Objects
dataURI		StatusBar
		Meta-Scripts
		CSS

IV. Proposed Gray Box Attacks on Phishing Detectors

We now describe the Gray Box attacks on Phishing Detectors considered in this paper, which can be divided into *simple* and *complex* attacks. In simple attacks, the attacker knows and targets only few specific features. However, in the complex attacks, we assume the attacker may know (and modify) a potentially huge number of the features used by the PD. We vary the percentage of features used by the defender that the attacker knows from 0 to 100%. This range captures all adversarial scenarios: black box (no features known), white box (all features known), and gray box (some features known).

A. Simple Attacks

We consider three very simple Gray Box attacks:

1) GBA-1: The attacker assumes that the PD uses information about the length of the URL to predict whether it belongs to a phishing website or not—as is done in several existing PDs [10], [11], [31]. Hence it is reasonable to assume that an attacker may try to circumvent such mechanisms. Often times, phishing URLs are longer than benign URLs in order to confuse and trick users into clicking on the link. For this reason, existing PDs are usually trained on malicious samples characterized by longer URLs. Thus, an attacker may try to evade detection by devising phishing URLs with shorter URLs: a possible way to accomplish this is by using a URL shortening service (e.g., tinyurl.com).
2) GBA-2: Here, the attacker assumes that the PD uses features related to the HTML-code (this is done, for instance, in the PD considered in [5]), but he may not know exactly which feature. Hence, he tries to alter some aspects of his HTML code. For example, he might know that some PDs consider the ratio of internal links (i.e. within the URL's domain) to external links. Hence, in this attack, he inserts a number of "internal" links to his URL domain that might fool classifiers that use this feature. We inserted such internal links to a host of resources such as images, favicons, CSS snippets, videos, audio, as well as the usual "textual" links. Figures 1 show how such an attack might be accomplished. The original webpage is in Figure 1a, whereas the adversarial webpage is in Figure 1b. In particular, the top of Figures 1 show the HTML-code, whereas the bottom show the rendered webpage. We can see from the red-box in Figure 1b that by manipulating the HTML code it is possible to insert 'fake' links to internal resources, which may favor an attacker to camouflage a phishing webpage into a benign webpage.
3) GBA-3: This attack is a combination of the GBA-1 and GBA-2 attacks.

Even though these attacks are relatively simple, there is considerable evidence that defenders use the HTML content in phishing websites and the structure of the URLs of phishing websites to build PDs [30], [32]. It is therefore reasonable to assume that attackers will try to use offensive techniques conforming to GBA-1–GBA-3.

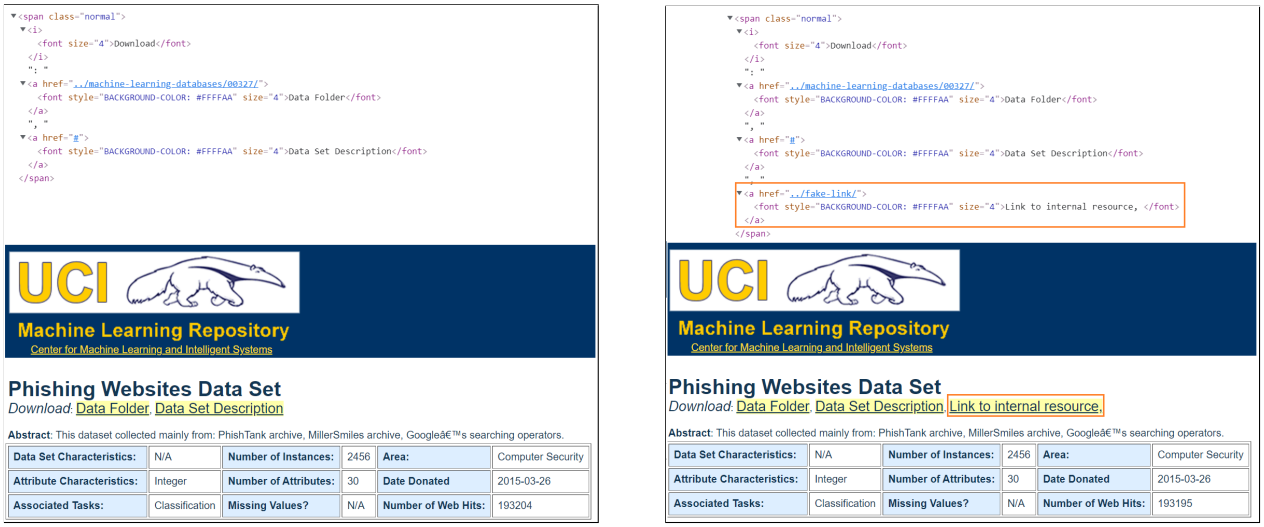
B. Complex attacks

In our more sophisticated attacks GBA-Δ, the adversary knows a variable subset of the features used by the defender. Let $\mathbb{F}_d$ be the set of features used by the defender (the PD) and $\mathbb{F}_a$ be the set of features that the attacker thinks the defender is using. This family of attacks is based on Δ, the percentage of features actually used by the defender that the attacker guessed correctly, i.e. $\Delta = \frac{|\mathbb{F}_d \cap \mathbb{F}_a|}{|\mathbb{F}_d|}$. In GBA-$\Delta$ attacks, we vary Δ by assuming the attacker knows some $\mathbb{F}_a$.

The 27 basic features used (cf. Table II) can be easily manipulated by experts who can easily insert/remove redirections, synthetically modify the URL length, or change any HTML functionality of the webpage to alter the features shown in Table II. This enables launching a huge number of attacks using GBA-Δ.

Note that when $\Delta = 0$, we have a black box attack, and when $\Delta = 1$, we have a white box attack.

In this paper, we evaluate 7 different values of Δ. Hence GBA-Δ represents many attack scenarios (7 in our case) where the adversary is able to modify any selection of $\Delta\%$ of the total set of features considered. This leads to an huge number of possibilities, viz. $2^{\Delta * |\mathbb{F}_d|} - 1$ which is a very large space of possible attacks. Simply put, GBA-Δ captures many possible complex evasion attacks conceivable by well-motivated and expert opponents.



(a) The original webpage.

(b) The 'adversarial' webpage.

Figure 1: An example of the GBA-2 attack. On the top there is the underlying HTML-code of the webpage, whereas at the bottom there is the rendered HTML shown in the webpage. The red box in Figure 1b shows the manipulation to the HTML-code and its result to the real page. The 'fake' link can also be hidden, preventing to be shown in the rendered HTML.

C. Impact of Adversarial Attacks

In this paper, we consider attacks on 13 well known classifiers. 9 are classical classifiers: Random Forest (RF), K-Nearest Neighbor (KNN), Decision Tree (DT), Logistic Regression (LR), Naive Bayes (NB), Support Vector Machines (SVM), Extra Trees (ET), Stochastic Gradient Descent (SGD), Bagging (Bag). We also consider 2 boosting techniques—AdaBoost (AB) and Gradient Boost (GB). Finally, we also evaluate 2 deep learning classifiers: Multi-Layer Perceptrons (MLP) and Google's recent "Deep and Wide" (DnW) method [79]. *Thus, we note that our Gray Box attackers assume that the PD is using any one of these 13 classifiers, but they do not need to know which one.* We report an overview of existing PDs and the datasets used for their evaluation in Table III, from which we observe that many of our classifiers are used by related work.

Table III: Classifiers and Dataset of existing PDs.

Reference	*Classifier*	*Dataset*
[29]	MLP	Custom
[7]	RF, NB, MLP, LR, SVM	Custom
[80]	RF, SVM, NB, MLP, LR	Custom
[10]	RF, NB, LR	Custom
[9]	RF, SVM, AB	UCI
[5]	RF, KNN, SVM, MLP, NB	UCI
[14]	MLP, SVM, KNN, NB, RF	UCI
[30]	SVM	DeltaPhish
[31]	LR, SVM	UCI
[32]	RF	Custom
[13]	RF, KNN, AB	Custom
[33]	RF	Ebbu

We define the *Impact* of attack Att_i^d of type i on dataset d and classifier Clf on a performance metric μ with the following Equation:

$$Impact(Att_i^d, Clf, \mu) = \frac{\mu(Clf|\neg Att_i^d) - \mu(Clf|Att_i^d)}{\mu(Clf|\neg Att_i^d)} \quad (1)$$

where $\mu(Clf|Cond)$ denotes the value of the performance metric μ of classifier Clf when condition $Cond$ is true.[15] In the above formulation, μ can be any measure of classifier performance (e.g. F1-score, Accuracy, etc).

For instance, suppose the F1-score of a given classifier (say Random Forest) is 80% when no attack is performed, and 60% when an attack is launched on it. In this case, $Impact(Att_i^d, RF, F1) = \frac{0.8-0.6}{0.8} = 0.25$, i.e. there is a 25% reduction in the F1-score. Note that Equation 1 can be used to measure the impact of *any* adversarial attack, not just Gray Box attacks.

We expect that *all* the considered Gray Box attacks (GBA-1–GBA-Δ) have a significant impact on classification algorithms used in the literature [8]–[11], [30]–[33]. But before we evaluate this, we introduce our proposed defensive mechanism.

V. Proposed Countermeasure: The POC Algorithm

In this section, we introduce the concept of *operation chains* or *oc*, and the POC algorithm. The basic idea behind POC is to create a new feature space (Ψ) by randomly mixing some of the features from $\mathbb{F}$ used by

[15] We consider two conditions: if an attack is present or absent.

a given PD. This makes it hard for the attacker to infer *which* features are used by a PD and *how* such features denote a phishing/benign webpage. In particular, even if the attacker knows $\Delta\%$ of the features of the PD, he may not know how to change them in order to make a malicious webpage be classified as benign. To achieve this, we use operation chains which consist of a base set of mapping operators that randomly transform some features into a new feature, which will be included in the actual feature space used by the 'hardened' PD. Simply put, the POC algorithm obfuscates the features used by a PD so that an adversary cannot easily tell what the PD is doing[16] and, hence, offensively react to such PD.

A. Formal description of POC

Without loss of generality, POC assumes that the features are numeric (real valued) and that categorical feature values will be replaced by values from a discrete domain. The obfuscation provided by POC is done via certain kinds of mappings.

A *unary (resp. binary) feature mapping* operator α (resp. β) is a mapping from $\mathbb{R}$ (resp. $\mathbb{R}\times\mathbb{R}$) to $\mathbb{R}$. We assume the existence of a set Fmop of unary and binary feature mapping operators. To obfuscate the underlying mechanism of POC, the Fmop set should consist of mappings that are hard to reverse. There are many such operations of course, and so we choose a suite of well-known, nonlinear mappings. In our implementation, we use $\mathsf{Fmop} = \{\log,\sin,\cos,\tan, exp^i, +, -, *, /\}$ as our feature mapping operators where $\{\log, \sin, \cos, \tan\}$ are unary operators, exp^i is not one but a family of numeric unary operators which take an input value x and return i^x for $i \in \{-3,-2,\ldots,2,3\}$. The arithmetic operators $+,-,*,/$ are binary feature mapping operators[17]. Note that our definitions below apply to virtually any choice of unary and binary operators in Fmop—we are not limited in any way to the specific operators chosen in our implementation—new ones and the definition of operation chains below can be seamlessly incorporated into our framework[18].

The POC algorithm maps a given set of features $\mathbb{F}$ into a new feature space Ψ, composed of operation chains, oc. Each oc is based on a subset $\overline{\mathbb{F}} \subseteq \mathbb{F}$ of features, which are combined via the unary or binary operators in Fmop. The final Ψ is then created by randomly selecting ψ (representing the dimensionality of Ψ) oc. Specifically, given a set $\mathbb{F}$ of (original) features, we can recursively define *operation chains*, each based on $\overline{\mathbb{F}}\subseteq\mathbb{F}$, as follows:

1) Each feature $f\in\overline{\mathbb{F}}$ is an oc of size 0.
2) For each unary operator $\alpha\in\mathsf{Fmop}$ and for each oc of size s, $\alpha(oc)$ is an oc of size $s+1$.
3) For each binary operator $\beta\in\mathsf{Fmop}$ and for each pair of operator chains oc_1, oc_2 of sizes s_1, s_2 respectively, $\beta(oc_1, oc_2)$ is an oc of size s_1+s_2+1.

Suppose $\mathbb{F}$ is the list of features shown in Table II and suppose $\overline{\mathbb{F}}$ consists of any two of these features, e.g. $\overline{\mathbb{F}} = \{f_1, f_2\}$. Then examples of operation chains based on $\overline{\mathbb{F}}$ and on the proposed set of Fmop include:

1) f_1 and f_2 are both oc of size 0.
2) $\sin(f_1), \cos(f_2)$ are oc that create new features by taking the sine and cosine, respectively of values of features f_1, f_2 respectively. They have size 1.
3) $\sin(f_1)+\cos(f_2)$ is an oc that generates a new feature that creates feature values by summing up the sine of the value of feature f_1 and the cosine of the value of feature f_2. This oc has size 3.
4) $exp(\sin(f_1)+\cos(f_2))$ creates a new feature whose value is $e^{\sin(f_1)+\cos(f_2)}$. The size of this oc is 4.

Thus our POC framework creates Ψ as follows.

1) (Initialization) First, we select $\overline{\mathbb{F}}$ from $\mathbb{F}$, representing the features used to create each oc. We then define the set of mapping operators, Fmop, and choose $MaxSize$ which is an integer greater than 0; finally, we set ψ, representing the cardinality of the new feature space Ψ.
2) (Operation Chain Transformations) We create oc of size $MaxSize$ or less by combining the features in $\overline{\mathbb{F}}$ via the operators in Fmop.
3) (Random Selection) We randomly select ψ operation chains, which will represent Ψ.

The POC Algorithm that formally captures the informal process described above is shown in Algorithm 1.

B. Analysis of POC

We now analyze our POC algorithm.

Relationship between Ψ and $\mathbb{F}$. We note that each feature in Ψ is represented by an oc that uses a subset of the features in $\mathbb{F}$. Hence, the new Ψ and the original $\mathbb{F}$ can be linked by how many features of $\mathbb{F}$ are included in Ψ. Let us define the *prevalence* of $\mathbb{F}$ w.r.t. a given Ψ as $\mathcal{P}(\mathbb{F},\Psi)$, which denotes the percentage of features in $\mathbb{F}$ that are included among all oc composing Ψ. As an example, if $\mathbb{F}=(f_1,f_2,f_3,f_4)$ and $\Psi=(oc_1,oc_2)$ with $oc_1=(f_1+f_2)$ and $oc_2=(sin(f_3)+f_1)$ then Ψ contains three out of four features of $\mathbb{F}$ (specifically, f_1,f_2,f_3), meaning that $\mathcal{P}(\mathbb{F},\Psi)$=75%. Two cases are possible:

[16]Our work is different from the type of obfuscation that a might perform in order to stop his/her code from being reverse engineered. But the idea is similar in both cases.

[17]Again, please note that these are the i's used in our implementation. The theory allows i to range over any set of integers.

[18]We do not claim that our selected mappings are the best. There is an infinite space of such mappings which can be used in POC.

Algorithm 1: Proposed POC algorithm.

Input: List of features $\bar{\mathbb{F}}$ included in a given dataset d; ϕ, number of new features that will compose the new set of $mapped\_features$; $MaxSize$ maximum size of an operation chain; Fmop_α set of unary feature mapping operators; Fmop_β set of binary feature mapping operators

Output: The $mapped\_features$ representing the new space Φ

$mapped\_features \leftarrow$ emptyList();
for $h \leftarrow 0$ **to** ϕ **by** 1 **do**
  $new\_feature \leftarrow$ computeNewFeature($\bar{\mathbb{F}}$, $MaxSize$);
  Insert $new\_feature$ in $mapped\_features$;
return $mapped\_features$
// Procedure that generates a $mapped\_feature$
Function *computeNewFeature($\bar{\mathbb{F}}$, $MaxSize$)*
  $feature\_block \leftarrow$ chooseFeatures($\bar{\mathbb{F}}$, $MaxSize$);
  $L \leftarrow$ len($feature\_block$);
  for $h \leftarrow 0$ **to** L **by** 1 **do**
    $\bar{f} \leftarrow$ unaryOperation($feature\_block[h]$);
    Replace $feature\_block[h]$ with $\bar{f}$;
  for $h \leftarrow 1$ **to** L **by** 1 **do**
    $\bar{f} \leftarrow$ binaryOperation($feature\_block$[0],$feature\_block[h]$);
    Replace $feature\_block$[0] with $\bar{f}$;
  $new\_feature \leftarrow feature\_block$[0];
  return $new\_feature$
// Procedure that determines the features from $\bar{\mathbb{F}}$ considered to generate a $mapped\_feature$
Function *chooseFeatures($\bar{\mathbb{F}}$, $MaxSize$)*
  $feature_block \leftarrow$emptyList();
  for $h \leftarrow 0$ **to** *randomChoice($MaxSize$)* **by** 1 **do**
    Insert randomChoice($\bar{\mathbb{F}}$) in $feature\_block$;
  return $feature\_block$
// Procedure that chooses and applies an unary operation among $Fmop_\alpha$ on a given feature f that composes a given $new\_feature$.
Function *unaryOperation(f)*
  $\bar{f} \leftarrow$ Apply $randomChoice(\mathsf{Fmop}_\alpha)$ to f;
  return $\bar{f}$
// Procedure that chooses and applies a binary operation among $Fmop_\beta$ on a given pair of features (f_1, f_2) that composes a given $new\_feature$.
Function *binaryOperation(f_1, f_2)*
  $\bar{f} \leftarrow$ Apply $randomChoice(\mathsf{Fmop}_\beta)$ to f_1 and f_2;
  return $\bar{f}$

- (complete *prevalence)* $\mathcal{P}(\mathbb{F}, \Psi) = 100\%$, i.e., Ψ uses *all* the features in $\mathbb{F}$. In this case[19], we can expect that using POC results in a PD with similar performance *in the absence of adversarial attacks* as a PD that does not use POC, because they will both use the same amount of information available to analyze each sample.
- (incomplete *prevalence*) $\mathcal{P}(\mathbb{F}, \Psi) < 100\%$, i.e., Ψ does not contain some features of $\mathbb{F}$. In this case, using POC will result in PDs that are trained on less information (due to the 'excluded' features), but with the capability of completely nullifying those adversarial attacks that target features of $\mathbb{F}$ not included in the *oc* of Ψ (by leveraging the well-known *feature removal* strategy [81], [82]).

We will investigate both of these circumstances.

Goal of POC. POC assumes that attackers can only change some features.[20] Hence, POC seeks to prevent hackers from: (i) reverse-engineering the PD by identifying its complete feature set; and (ii) changing one or two small things to evade a PD.

The first goal is achieved by randomly using the old features ($\mathbb{F}$) to create a new feature space (Ψ) which makes it harder to reverse-engineer (or 'steal' [83]) the classifier used by the PD (e.g. the attack against the Google Chrome filter [76]). This is because the attacker's manipulation will affect multiple features simultaneously and differently, making it hard for him to infer the features used by the PD. Moreover, as described in Section III-D, ML detectors must be updated with new data to prevent concept drift [40], [78]. Therefore, each new application of POC will result in a new feature space, ensuring that attackers that 'cracked' the old PDs have to repeat the process again.

The second goal is achieved as a direct consequence of the above. The new feature space Ψ induces confusion, e.g. a feature $f \in \mathbb{F}$ may be mapped to $sin(f)$ and then further combined into an *oc* such as $2^{sin(f)+log(f)}$. By using irreversible feature mappings, it is unlikely that the attacker can recover the original features—even if the attacker were to get hold of the POC-feature vectors describing each sample. Hence, an attacker may be successful in evading a PD by "making a URL shorter" (as in GBA-1), thus manipulating the corresponding feature. However, against a POC-hardened PD, the manipulated feature will affect many *oc*s in unpredictable ways. Increased protection is also provided by feature-removal (cf. Section V-B)—e.g., if the features manipulated in GBA-1 are not used (or nullified[21]) by a POC-hardened PD. Nevertheless, as operation chains get longer, recovering the original features from the new feature values is very challenging. Therefore, the selected mapping operators (e.g. *log*, *sin*, *cos* used in this paper) should include many irreversible functions that make the attacker's job even more difficult.

[19]Of course, this can only be true if $\bar{\mathbb{F}} = \mathbb{F}$.

[20]The assumption is realistic, because some features cannot be changed without altering the malicious nature of a webpage, or require a huge resource investment (e.g., modifying reputation features based on DNS records requires compromising the respective DNS servers).

[21]Feature removal can occur 'indirectly' even if $\mathcal{P}(\mathbb{F}, \Psi)=100\%$, e.g., if Ψ contains an *oc* where a feature is multiplied by another feature whose value is 0 for most samples.

Summary. As long as Ψ contains enough of the original features (e.g., $\mathcal{P}(\mathbb{F}, \Psi) \geq 70\%$), and as long as such Ψ has high dimensionality (e.g., $\psi \geq 15oc$), a POC-hardened PD has the potential to: capture enough of the distributional properties of the original data (providing good classification power); obfuscating the features of the PD (confusing the attacker); and mitigating the attacks. Finally, we stress that the mappings we use in this paper can be replaced (or expanded) by other mappings (e.g. hyperbolic tangent) that are irreversible.

VI. Experiments

We use Python3 code (and leverage Scikit-Learn) to extract all the features in Table II, as well as all the Gray Box attacks and the POC algorithm.

Our experiments focus on validating POC w.r.t. (i) mitigating Gray Box attacks and (ii) maintaining high performance when there are no attacks. For this, we first assess the performance of existing PDs and their POC versions in the absence of attacks. This shows how POC performs when no attacks occur, and establishes the performance of the selected PDs. We then evaluate the *Impact* of the attacks on existing PDs and their POC versions. We first show performance w.r.t. *simple* attacks (GBA-1 to GBA-3), and then show the results against *complex* attacks (the 7 variants of GBA-Δ). All these evaluations involve the 13 classifiers trained on each of the 4 datasets considered in the paper. We discuss all these results in Section VII.

The rest of this section assesses POC with complete *prevalence*, i.e., when $\mathcal{P}(\mathbb{F},\Psi) = 100\%$. The motivation is twofold: ensure a fair comparison of the performance between the baseline and POC-hardened classifiers; and assess the hardening of POC provided by its (random) feature mapping, and not due to the exclusion of the 'attacked' features (cf. Section V-B). We will also evaluate POC when $\mathcal{P}(\mathbb{F}, \Psi) < 100\%$ in Section VIII.

A. Testbed

Baselines. Our experimental testbed contains 13 'baseline' classifiers (used in existing phishing detectors) and the 4 different datasets considered in this paper (UCI, Mendeley, DeltaPhish, LNU-Phish—see Section III). For each dataset, all such baseline classifiers adopt the same set of features. For the LNU-Phish and DeltaPhish datasets, we use the features in Table II (we manually compute the *REP*-features in the DeltaPhish dataset using the same methodology adopted to create our LNU-Phish dataset—see Section III). For the other datasets (Mendeley, UCI), we use all the features provided by their creators. We note that all these datasets share similar *URL*- and *HTML*-features, which are all affected by the attacks considered in this paper. We apply an 80:20 split for the training and test partitions, where the proportion of benign-to-malicious samples is as reported in Table I. We did extensive hyper-parameter optimization using grid search across the parameter space of each classifier in order to tune that classifier to achieve best performance. The results show the performance after this hyper-parameter optimization (the most influential parameters for our algorithms are provided in the supplementary material).

POC-hardened classifiers. We consider the (fine-tuned) 'baseline' variant of each classifier as basis. We use the Fmop in Section V-A, and specify $\bar{\mathbb{F}} = \mathbb{F}$, $MaxSize$=3, and ψ=20, meaning that Ψ is composed of 20 oc. Because we are considering $\mathcal{P}(\mathbb{F}, \Psi)$=100%, the POC classifiers include—across their 20 oc—all features of the baseline classifiers (yielding a 'pseudo-random' POC). We train (and test) each POC classifier on all the datasets considered in the paper by using the same splits. Each classifier (on each dataset) adopts the mapping produced by POC that achieves the best performance during development (i.e., in the no-attack case): in reality only a single PD (i.e., the one with best performance) is deployed, and the adversarial attacks cannot be anticipated. The Gray Box attacks are generated by using the malicious samples in the test partition as base. When we compute the *Impact* of an attack, we rely on the Recall as performance metric (see Equation 1), because we consider *evasion* attacks which involve only malicious samples. The 7 attacks of GBA-Δ family are repeated 10 times, each time by modifying different features (but always corresponding to the same Δ), and the reported values correspond to the average of these 10 trials.

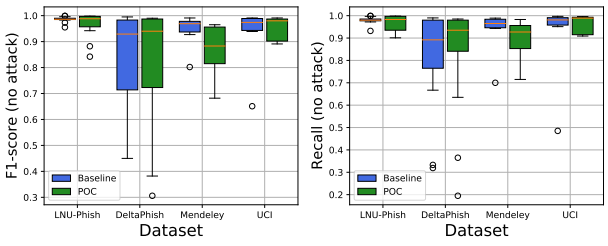
POC is not likely to yield good results when used with randomly chosen parameters. During training time, we identify the best parameter settings for each classifier using standard grid-search based hyper-parameter optimization. [22] Once the classifier is trained, the resulting POC 'hardened' classifier obtains a performance comparable to the 'baseline' (in the absence of adversarial attacks) on test data not seen during training. Though costly, training is a one time operation. We will discuss the run-time for training in Section VI-E.

B. No Attack Case: Performance of Baselines vs. POC

We first assess all PDs (the 'baselines' and their POC-hardened variants) when no adversarial attacks occur.

[22] Grid search looks at all the possible combinations of $\mathbb{F}$ that result in Ψ. To make this humanly feasible, we considered 100 combinations and choose the best one for each classifier.

Table IV shows the result of using the 13 baseline classifiers on the four datasets as done by past work, while Table V shows the result of using the POC approach using all the features in the dataset. We tested the efficacy of POC in the no attack case because of concerns that the random manipulation of features could lead to a drop in performance. By comparing Table IV with Table V, we observe that the notion of operation chains used by POC does lead to a small reduction in performance of the best classifier in each case (see the last row of Tables IV and V), but we note that this drop is very small (we will discuss such results in Section VII-A). To better visualize the magnitude of the drop, Figures 2 shows the distribution of the F1-score (Figure 2a) and Recall (Figure 2b) achieved by the baseline classifiers (blue boxplots) and their POC variant (green boxplots) on each dataset. We see that some nontrivial degradation only occurs with the Mendeley dataset, but a close look at Table V reveals that the best classifiers still yield high performance in the absence of attacks: for instance, the baseline ET classifier on Mendeley has 0.99 F1-score and Recall, and its hardened POC variant has 0.96 F1-score and 0.95 Recall.



(a) Distribution of the F1-score. (b) Distribution of the Recall.

Figure 2: Comparison (baseline vs POC) in the no-attack case.

However, the next few experiments show that POC outperforms past work when the adversary carries out any of the considered attacks (both simple and complex), and hence this negligible reduction in performance is amply compensated by POC's increased robustness.

C. Attacks against existing PDs

We assess the *Impact* of the considered attacks on the baseline classifiers. We begin with the simple attacks (GBA-1, GBA-2 and GBA-3) and then proceed with the complex attacks (the 7 variants of GBA-Δ).

Impact of simple attacks: Table VI shows the *Impact* of the GBA-1–GBA-3 attack on each of the 4 datasets and 13 classifiers (used in existing phishing detectors). We see, for instance, that GBA-1 against the RF of the LNU-Phish dataset leads to a 12.4% drop, but the drop is 96.6% on the DeltaPhish dataset. On average (last rows of subtables in Table VI), the GBA-1–GBA-3 attacks lead to significant drops on all datasets (varying from 11.7% to 90.2%), irrespective of the classifier used.

Impact of complex attacks: We now consider the complex attacks represented by GBA-Δ, which assume that the attacker knows $\Delta\%$ of the features used by the targeted PD. We vary Δ from $10-70\%$ in steps of 10%. Table VII shows the *Impact* of these attacks on existing PDs on all 4 datasets. Unsurprisingly, as Δ increases, the attacks have a greater *Impact* on average (the last line showing "averages" in the 4 subtables of Table VII showing steady increases from left to right). Moreover, some of the attacks are very effective—for instance, if the attacker knows 30% of the features used by the defender, the *Impact* ranges from 15.7% to 43.1% which is very substantial. A formal statistical analysis of such results in presented in Section VII-A.

We observe that some attacks caused a negative *Impact* (e.g., the NB in Table VIIc), implying that the PD was able to correctly recognize *more* phishing samples than in the absence of attacks. Such occurrence is a byproduct of a less than optimal training phase, because the adversarial manipulation resulted in a sample that the classifier considers to be "more malicious" than its non-modified variant (as shown in Table IV, the NB classifier on the Mendeley dataset achieves the lowest performance).

D. Attacks against POC

We now assess the effectiveness of POC in protecting against the considered Gray Box attacks. We do so by measuring the *Impact* of every attack against the POC version of each baseline classifier, and computing the *Impact difference* between the baseline and its POC variant. This allows an immediate understanding of the results: if the number is greater than 0, then POC mitigated the attack; otherwise, it was more affected.

We begin by evaluating the simple attacks, and then conclude with the complex attacks.

Simple attacks: We assess POC against the simple GBA-1–GBA-3 attacks. These results are reported in Table VIII. A positive difference (shown in bold) means that POC was more resilient to the attack than the baselines, while a negative number means POC was less resilient; higher values are highlighted with a darker background. Table VIII consists mostly of bold entries, showing that POC is more resilient than past work for almost all combinations of dataset and classifier used. Additionally, we see from the last rows ("average") that on average POC exhibited superior performance for each of the 4 datasets considered: the *Impact* of the GBA-1–

Table IV: No Attack Case: Results of the 'baseline' PDs for each dataset and classifier (using all features in the dataset).

	LNU-Phish				DeltaPhish				Mendeley Phishing				UCI Phishing			
Classifier	*F1-score*	*Acc*	*FPR*	*TPR*	*F1-score*	*Acc*	*FPR*	*TPR*	*F1-score*	*Acc*	*FPR*	*TPR*	*F1-score*	*Acc*	*FPR*	*TPR*
RF	0.973	0.982	0.013	0.972	0.959	0.989	0.001	0.926	0.978	0.975	0.033	0.989	0.973	0.973	0.045	0.975
SVM	0.983	0.989	0.004	0.974	0.450	0.876	0.025	0.319	0.927	0.928	0.092	0.943	0.943	0.936	0.113	0.958
KNN	0.996	0.998	0.002	0.997	0.971	0.991	0.007	0.971	0.974	0.975	0.037	0.984	0.984	0.983	0.045	0.997
SGD	0.985	0.990	0.003	0.977	0.458	0.874	0.030	0.333	0.931	0.932	0.083	0.943	0.939	0.932	0.114	0.951
DT	0.986	0.991	0.006	0.985	0.988	0.996	0.002	0.985	0.946	0.948	0.056	0.948	0.990	0.989	0.017	0.991
LR	0.985	0.990	0.003	0.977	0.521	0.776	0.293	0.765	0.937	0.938	0.073	0.946	0.942	0.935	0.115	0.957
NB	0.955	0.971	0.010	0.932	0.714	0.915	0.050	0.667	0.802	0.832	0.046	0.700	0.651	0.715	0.008	0.485
MLP	0.990	0.994	0.001	0.983	0.929	0.978	0.007	0.892	0.976	0.976	0.023	0.975	0.984	0.983	0.026	0.985
AB	0.986	0.991	0.002	0.977	0.872	0.961	0.019	0.833	0.949	0.951	0.054	0.952	0.947	0.941	0.107	0.962
ET	0.999	0.999	0.001	0.999	0.988	0.996	0.001	0.980	0.991	0.991	0.006	0.988	0.991	0.990	0.015	0.992
GB	0.999	0.999	0.001	0.999	0.995	0.998	0.001	0.990	0.990	0.991	0.009	0.989	0.990	0.988	0.020	0.993
DnW	0.988	0.992	0.002	0.980	0.929	0.980	0.005	0.886	0.969	0.970	0.027	0.965	0.974	0.970	0.051	0.981
Bag	0.987	0.992	0.003	0.980	0.983	0.995	0.003	0.980	0.986	0.987	0.012	0.984	0.989	0.988	0.020	0.991
best	0.999	0.999	0.001	0.999	0.995	0.998	0.001	0.990	0.991	0.991	0.007	0.989	0.991	0.990	0.023	0.997

Table V: No Attack Case: Results of the POC-hardened PDs for each dataset (using all features in the dataset).

	LNU-Phish				DeltaPhish				Mendeley Phishing				UCI Phishing			
Classifier	*F1-score*	*Acc*	*FPR*	*TPR*	*F1-score*	*Acc*	*FPR*	*TPR*	*F1-score*	*Acc*	*FPR*	*TPR*	*F1-score*	*Acc*	*FPR*	*TPR*
RF	0.997	0.998	0.001	0.997	0.990	0.997	0.001	0.980	0.965	0.963	0.033	0.963	0.988	0.987	0.028	0.994
SVM	0.957	0.972	0.009	0.935	0.306	0.866	0.020	0.195	0.682	0.646	0.382	0.715	0.898	0.888	0.184	0.913
KNN	0.997	0.998	0.001	0.997	0.985	0.995	0.004	0.985	0.933	0.925	0.124	0.983	0.986	0.985	0.032	0.993
SGD	0.842	0.888	0.121	0.904	0.409	0.841	0.105	0.365	0.815	0.800	0.208	0.831	0.902	0.892	0.174	0.915
DT	0.989	0.993	0.004	0.986	0.990	0.997	0.001	0.980	0.881	0.875	0.104	0.869	0.987	0.986	0.026	0.991
LR	0.942	0.963	0.008	0.904	0.382	0.689	0.422	0.635	0.781	0.785	0.128	0.723	0.895	0.884	0.187	0.909
NB	0.882	0.921	0.071	0.901	0.723	0.902	0.125	0.850	0.734	0.642	0.599	0.927	0.891	0.879	0.209	0.913
MLP	0.990	0.993	0.001	0.982	0.940	0.982	0.014	0.935	0.883	0.868	0.180	0.933	0.980	0.979	0.045	0.989
AB	0.987	0.991	0.005	0.984	0.921	0.977	0.014	0.900	0.870	0.864	0.108	0.853	0.923	0.915	0.148	0.940
ET	0.998	0.998	0.001	0.997	0.987	0.996	0.001	0.975	0.962	0.960	0.029	0.954	0.988	0.987	0.028	0.994
GB	0.998	0.998	0.001	0.998	0.985	0.995	0.001	0.975	0.961	0.958	0.040	0.962	0.991	0.990	0.022	0.996
DnW	0.987	0.992	0.004	0.983	0.901	0.970	0.006	0.841	0.901	0.912	0.114	0.913	0.968	0.964	0.077	0.985
Bag	0.988	0.992	0.003	0.983	0.990	0.997	0.001	0.980	0.956	0.954	0.044	0.956	0.987	0.986	0.029	0.993
best	0.998	0.998	0.001	0.998	0.990	0.997	0.001	0.980	0.962	0.965	0.030	0.963	0.991	0.990	0.022	0.996

Table VI: Simple attack case: *Impact* of GBA-1 to GBA-3 on every baseline classifier for each dataset (lower is better).

Clf	LNU-Phish	DeltaPhish	Mendeley	UCI
RF	0.124	0.966	0.305	0.750
SVM	0.107	0.784	0.397	0.730
KNN	0.066	0.436	0.017	0.189
SGD	0.121	0.422	0.341	0.730
DT	0.126	0.942	0.813	0.755
LR	0.003	0.080	0.484	0.746
NB	0.267	0.911	0.096	0.506
MLP	0.137	0.696	0.293	0.712
AB	0.114	0.994	0.252	0.748
ET	0.190	0.965	0.619	0.696
GB	0.132	0.984	0.373	0.616
DnW	0.007	0.610	0.249	0.269
Bag	0.124	0.980	0.675	0.723
average	0.117	0.751	0.378	0.628

(a) *Impact* of GBA-1.

Clf	LNU-Phish	DeltaPhish	Mendeley	UCI
RF	0.290	0.099	0.161	0.112
SVM	0.437	0.103	0.070	0.200
KNN	0.567	0.081	0.970	0.361
SGD	0.672	0.573	0.954	0.157
DT	0.097	0.078	0.121	0.159
LR	0.022	0.270	0.280	0.200
NB	0.364	1.000	−0.056	0.502
MLP	0.048	0.134	0.123	0.272
AB	0.075	0.126	0.128	0.104
ET	0.189	0.139	0.081	0.090
GB	0.172	0.062	0.080	0.455
DnW	0.102	0.004	0.301	0.739
Bag	0.145	0.119	0.732	0.162
average	0.244	0.214	0.303	0.270

(b) *Impact* of GBA-2.

Clf	LNU-Phish	DeltaPhish	Mendeley	UCI
RF	0.311	1.000	0.669	1.000
SVM	0.540	0.972	0.611	1.000
KNN	0.654	0.709	0.972	0.673
SGD	0.697	0.926	0.998	1.000
DT	0.287	0.942	0.823	1.000
LR	0.015	0.603	0.831	1.000
NB	0.460	1.000	0.062	1.000
MLP	0.171	0.930	0.452	0.998
AB	0.132	1.000	0.481	1.000
ET	0.198	1.000	0.628	1.000
GB	0.185	1.000	0.510	1.000
DnW	0.106	0.652	0.469	0.999
Bag	0.285	1.000	0.951	0.940
average	0.310	0.902	0.650	0.97

(c) *Impact* of GBA-3.

GBA-3 attacks on the baselines are 1% to 53.5% higher than for POC (last row of the subtables in Table VIII).

Complex attacks: We now turn to the value of POC in protecting against the 7 variants of the GBA-Δ attacks. The results are reported in Table IX. A positive difference (denoted in bold) means that POC was more resilient to the attack than the baselines, while a negative number means POC was less resilient; higher values are highlighted with a darker background. We see that most entries in the table are in boldface, suggesting that POC is more resilient to the GBA-Δ attack irrespective of the dataset and classifier used. As can be seen from the last rows ("average"), POC exhibited superior performance for 27 of 28 combinations of dataset and classifier; the one exception is the UCI dataset with $\Delta = 60\%$ where the performance of the baseline is very slightly better than that of POC.

Finally, Figure 3 shows the aggregated results of all our attacks on all datasets and classifiers. Specifically, Figure 3 shows 10 pairs of boxplots: each pair represents one of our considered attacks (the 3 simple, and the 7 complex attacks). The blue (resp. green) boxplot of each pair represents the distribution of the *Impact* of the corresponding attack against the baseline (resp. POC) classifiers (we exclude the few outliers). The figure shows that, in general, the POC classifiers are less affected by the attacks.

All tables showing the *Impact* of the attacks (i.e., Tables VI to IX) are provided with the same format as those of the *no attack* case (Table IV and Table V) in the supplementary material.

Table VII: Complex attack case: *Impact* of the GBA-Δ attacks on the baseline PDs for every dataset (lower is better).

LNU-Phish	**Features Modified (Δ)**						
Classifier	*10%*	*20%*	*30%*	*40%*	*50%*	*60%*	*70%*
RF	0.001	0.044	0.084	0.145	0.173	0.178	0.137
SVM	0.072	0.243	0.481	0.496	0.577	0.681	0.686
KNN	0.001	0.013	0.022	0.026	0.043	0.043	0.034
SGD	0.258	0.278	0.307	0.360	0.425	0.496	0.612
DT	−0.002	−0.001	0.099	0.099	0.205	0.204	0.204
LR	0.082	0.089	0.239	0.325	0.368	0.387	0.441
NB	0.068	0.084	0.125	0.199	0.340	0.519	0.639
MLP	0.102	0.113	0.193	0.375	0.450	0.742	0.650
AB	0.005	0.005	0.101	0.000	0.000	−0.003	−0.002
ET	−0.002	0.005	0.011	0.047	0.090	0.087	0.087
GB	0.045	0.049	0.153	0.344	0.369	0.478	0.530
DnW	0.012	0.010	0.009	0.056	0.009	0.089	0.264
Bag	0.003	0.008	0.219	0.271	0.277	0.277	0.337
average	0.049	0.072	0.157	0.211	0.255	0.321	0.356

(a) GBA-Δ: *Impact* for the LNU-Phish dataset.

DeltaPhish	**Features Modified (Δ)**						
Classifier	*10%*	*20%*	*30%*	*40%*	*50%*	*60%*	*70%*
RF	0.038	0.198	0.356	0.389	0.678	0.599	0.567
SVM	0.049	0.221	0.037	0.046	−0.123	−0.082	−0.405
KNN	0.002	0.157	0.327	0.491	0.568	0.699	0.557
SGD	−0.064	0.106	0.267	0.111	0.302	0.484	0.112
DT	−0.001	0.080	0.110	0.191	0.189	0.270	0.269
LR	0.135	0.327	0.522	0.543	0.625	0.659	0.523
NB	0.037	0.105	0.254	0.474	0.631	0.640	0.693
MLP	0.081	0.211	0.317	0.389	0.528	0.626	0.763
AB	−0.023	0.167	0.224	0.223	0.176	0.599	0.637
ET	0.010	0.096	0.235	0.254	0.503	0.583	0.687
GB	0.018	0.080	0.138	0.220	0.292	0.376	0.415
DnW	0.012	0.010	0.009	0.011	0.009	0.098	0.265
Bag	0.087	0.150	0.298	0.319	0.351	0.393	0.400
average	0.030	0.147	0.238	0.282	0.363	0.458	0.421

(b) GBA-Δ: *Impact* for the DeltaPhish dataset.

Mendeley	**Features Modified (Δ)**						
Classifier	*10%*	*20%*	*30%*	*40%*	*50%*	*60%*	*70%*
RF	0.033	0.235	0.293	0.474	0.618	0.593	0.596
SVM	0.189	0.317	0.345	0.299	0.467	0.596	0.643
KNN	0.132	0.240	0.408	0.631	0.653	0.706	0.700
SGD	0.041	0.212	0.218	0.184	0.245	0.320	0.336
DT	0.095	0.247	0.371	0.461	0.517	0.585	0.522
LR	0.082	0.117	0.213	0.397	0.433	0.393	0.461
NB	−0.072	−0.148	−0.233	−0.227	−0.202	−0.169	−0.105
MLP	0.118	0.141	0.148	0.221	0.239	0.300	0.374
AB	0.049	0.156	0.226	0.375	0.450	0.548	0.453
ET	0.196	0.345	0.564	0.659	0.758	0.672	0.783
GB	0.034	0.116	0.261	0.323	0.406	0.431	0.614
DnW	0.037	0.081	0.185	0.271	0.456	0.568	0.631
Bag	0.129	0.444	0.570	0.580	0.746	0.805	0.886
average	0.083	0.193	0.275	0.358	0.445	0.488	0.531

(c) GBA-Δ: *Impact* for the Mendeley dataset.

UCI	**Features Modified (Δ)**						
Classifier	*10%*	*20%*	*30%*	*40%*	*50%*	*60%*	*70%*
RF	0.022	0.304	0.554	0.481	0.553	0.329	0.038
SVM	0.135	0.337	0.501	0.577	0.537	0.352	0.338
KNN	0.163	0.374	0.481	0.557	0.800	0.864	0.814
SGD	0.201	0.298	0.371	0.532	0.657	0.601	0.400
DT	0.300	0.390	0.403	0.389	0.435	0.480	1.000
LR	0.199	0.429	0.435	0.604	0.445	0.316	0.220
NB	0.400	0.350	0.237	0.085	−0.221	−0.422	−0.735
MLP	0.254	0.467	0.715	0.639	0.695	0.331	0.261
AB	0.200	0.310	0.401	0.389	0.454	0.312	0.334
ET	0.110	0.212	0.348	0.607	0.110	0.085	0.086
GB	0.123	0.270	0.364	0.612	0.730	0.565	0.464
DnW	0.205	0.250	0.384	0.491	0.472	0.586	0.627
Bag	0.056	0.192	0.409	0.517	0.616	0.666	0.695
average	0.182	0.322	0.431	0.498	0.483	0.390	0.350

(d) GBA-Δ: *Impact* for the UCI dataset.

Table VIII: Simple attack case. Differences between the *Impact* of GBA-1 to GBA-3 on the baselines and on POC (higher is better).

Clf	**LNU-Phish**	**DeltaPhish**	**Mendeley**	**UCI**
RF	**0.102**	**0.061**	**0.284**	**0.075**
SVM	**0.115**	**0.791**	**0.372**	**0.080**
KNN	**0.044**	**0.157**	−0.030	**0.013**
SGD	**0.179**	**0.001**	**0.349**	**0.027**
DT	**0.130**	**0.110**	**0.807**	**0.030**
LR	**0.028**	**0.076**	**0.477**	**0.025**
NB	**0.133**	**0.348**	**0.096**	−0.152
MLP	**0.124**	**0.273**	**0.284**	−0.073
AB	**0.089**	**0.068**	**0.252**	**0.235**
ET	**0.171**	**0.115**	**0.544**	**0.086**
GB	**0.115**	**0.084**	**0.342**	**0.106**
DnW	−0.001	−0.015	**0.050**	**0.030**
Bag	**0.120**	**0.083**	**0.633**	**0.225**
average	**0.103**	**0.166**	**0.344**	**0.055**

(a) GBA-1: *Impact* difference.

Clf	**LNU-Phish**	**DeltaPhish**	**Mendeley**	**UCI**
RF	**0.082**	−0.365	**0.058**	**0.057**
SVM	**0.426**	**0.026**	**0.377**	**0.191**
KNN	**0.048**	−0.876	**0.903**	**0.256**
SGD	**0.763**	−0.223	**0.682**	**0.137**
DT	−0.001	−0.210	**0.032**	**0.009**
LR	**0.101**	**0.274**	**0.290**	**0.197**
NB	**0.270**	**0.077**	−0.125	**0.540**
MLP	−0.106	**0.054**	**0.058**	**0.079**
AB	−0.019	−0.408	**0.067**	**0.116**
ET	−0.011	−0.215	−0.061	**0.034**
GB	−0.002	−0.398	−0.023	**0.223**
DnW	−0.03	−0.219	**0.288**	**0.130**
Bag	−0.059	−0.386	**0.591**	**0.058**
average	**0.112**	−0.220	**0.241**	**0.156**

(b) GBA-2: *Impact* difference.

Clf	**LNU-Phish**	**DeltaPhish**	**Mendeley**	**UCI**
RF	−0.005	**0.108**	**0.348**	**0.242**
SVM	**0.590**	**0.080**	**0.415**	**0.845**
KNN	**0.040**	**0.538**	**0.905**	**0.420**
SGD	**0.357**	**0.511**	**0.038**	**0.806**
DT	**0.152**	**0.223**	**0.569**	**0.484**
LR	−0.43	**0.859**	**0.958**	**0.831**
NB	**0.038**	**0.067**	**0.127**	**0.987**
MLP	−0.059	**0.129**	**0.092**	**0.508**
AB	**0.025**	**0.046**	**0.248**	**0.550**
ET	−0.081	**0.185**	**0.284**	**0.450**
GB	**0.021**	**0.075**	**0.161**	**0.237**
DnW	−0.011	−0.019	**0.155**	**0.356**
Bag	−0.036	**0.116**	**0.625**	**0.233**
average	**0.046**	**0.224**	**0.378**	**0.535**

(c) GBA-3: *Impact* difference.

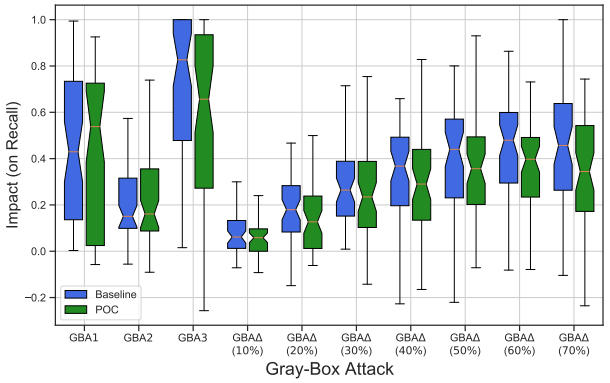


Figure 3: Aggregated *Impact* on all the baseline and POC classifiers.

E. Run Time of Training Phase

Table X shows the time (in seconds) required to train the baseline version and the POC-hardened variant of each classifier.[23]

Overview. We see that neural net classifiers (MLP, DnW) require the most time to train, regardless of whether POC is applied or not. As these classifiers do not provide great detection performance (as shown in previous sections), we do not recommend them as phishing detectors.

Baseline vs POC. Surprisingly, the training time of POC is comparable to that of the corresponding baseline.

[23]Experiments were performed on an Intel 7700HQ CPU (4 cores, 8 threads, 2.8 GHz) with 32GB RAM. We parallelized computations of those classifiers that support multiprocessing (according to scikit-learn).

Table IX: Complex attack case. Differences between the *Impact* of the GBA-Δ attack on the baselines and on POC (higher is better).

LNU-Phish	Features Modified (Δ)						
Classifier	*10%*	*20%*	*30%*	*40%*	*50%*	*60%*	*70%*
RF	**0.003**	**0.004**	−0.003	−0.111	−0.197	−0.315	−0.408
SVM	**0.048**	**0.107**	**0.196**	**0.210**	**0.150**	**0.146**	**0.084**
KNN	**0.004**	**0.004**	**0.005**	**0.003**	**0.005**	**0.008**	**0.002**
SGD	**0.183**	**0.179**	**0.209**	**0.161**	**0.144**	**0.104**	**0.070**
DT	**0.000**	**0.001**	**0.006**	−0.029	−0.034	−0.237	−0.133
LR	**0.054**	**0.017**	**0.091**	**0.118**	**0.096**	**0.062**	**0.069**
NB	**0.067**	**0.076**	**0.098**	**0.116**	**0.115**	**0.056**	**0.063**
MLP	**0.001**	**0.105**	**0.093**	**0.090**	**0.247**	**0.499**	**0.559**
AB	**0.008**	**0.008**	−0.027	−0.136	−0.166	−0.370	−0.368
ET	**0.002**	−0.001	0.000	**0.005**	**0.006**	−0.075	−0.188
GB	**0.013**	**0.013**	**0.011**	**0.107**	**0.053**	**0.171**	**0.189**
DnW	−0.015	−0.004	−0.012	**0.03**	−0.004	**0.005**	**0.025**
Bag	**0.003**	**0.003**	**0.014**	**0.104**	**0.209**	**0.209**	**0.305**
average	**0.029**	**0.039**	**0.053**	**0.051**	**0.048**	**0.021**	**0.021**

(a) GBA-Δ: *Impact* differences for the LNU-Phish dataset.

DeltaPhish	Features Modified (Δ)						
Classifier	*10%*	*20%*	*30%*	*40%*	*50%*	*60%*	*70%*
RF	−0.032	**0.072**	**0.161**	**0.127**	**0.260**	**0.157**	**0.054**
SVM	**0.311**	**0.283**	**0.425**	**0.532**	**0.753**	**0.764**	**0.251**
KNN	**0.024**	**0.169**	**0.197**	**0.369**	**0.391**	**0.296**	**0.314**
SGD	**0.188**	**0.164**	**0.409**	**0.027**	**0.374**	**0.291**	**0.347**
DT	−0.084	−0.141	−0.238	−0.211	−0.220	−0.231	−0.219
LR	**0.041**	**0.070**	**0.191**	**0.086**	**0.173**	**0.127**	**0.243**
NB	−0.058	−0.067	−0.027	**0.114**	**0.131**	**0.154**	−0.051
MLP	−0.130	−0.174	−0.188	−0.140	**0.022**	**0.219**	**0.602**
AB	−0.023	−0.022	−0.080	−0.216	−0.405	−0.093	−0.027
ET	−0.051	−0.045	−0.044	−0.094	**0.011**	−0.037	**0.033**
GB	−0.014	−0.019	−0.077	−0.074	−0.039	−0.042	−0.026
DnW	−0.058	−0.003	**0.0310**	**0.037**	−0.010	**0.032**	**0.037**
Bag	**0.050**	**0.050**	**0.104**	**0.096**	**0.136**	**0.157**	**0.090**
average	**0.012**	**0.0259**	**0.066**	**0.051**	**0.121**	**0.138**	**0.126**

(b) GBA-Δ: *Impact* differences for the DeltaPhish dataset.

Mendeley	Features Modified (Δ)						
Classifier	*10%*	*20%*	*30%*	*40%*	*50%*	*60%*	*70%*
RF	−0.034	−0.003	**0.011**	**0.099**	**0.113**	−0.002	−0.052
SVM	**0.282**	**0.258**	**0.142**	**0.464**	−0.158	**0.040**	**0.200**
KNN	**0.078**	**0.112**	**0.150**	**0.292**	**0.264**	**0.288**	−0.037
SGD	**0.052**	**0.175**	**0.114**	**0.089**	**0.075**	**0.128**	**0.265**
DT	−0.063	**0.027**	−0.086	−0.016	**0.084**	**0.129**	**0.174**
LR	**0.097**	**0.126**	**0.127**	**0.234**	**0.232**	**0.125**	−0.106
NB	−0.073	−0.135	−0.196	−0.183	−0.143	−0.106	−0.037
MLP	−0.008	−0.028	−0.114	−0.182	−0.003	**0.116**	−0.056
AB	**0.023**	**0.049**	**0.114**	**0.067**	**0.066**	**0.319**	**0.162**
ET	**0.159**	**0.073**	**0.158**	**0.245**	**0.242**	**0.194**	**0.138**
GB	−0.044	**0.059**	**0.007**	**0.037**	**0.058**	−0.060	**0.076**
DnW	**0.068**	**0.073**	**0.055**	**0.001**	**0.124**	**0.111**	**0.049**
Bag	−0.034	**0.173**	**0.155**	**0.062**	**0.115**	**0.096**	**0.242**
average	**0.039**	**0.074**	**0.049**	**0.093**	**0.083**	**0.106**	**0.079**

(c) GBA-Δ: *Impact* differences for the Mendeley dataset.

UCI	Features Modified (Δ)						
Classifier	*10%*	*20%*	*30%*	*40%*	*50%*	*60%*	*70%*
RF	−0.034	**0.002**	**0.140**	−0.199	**0.102**	−0.019	−0.067
SVM	−0.070	**0.040**	**0.068**	−0.016	**0.113**	**0.009**	**0.105**
KNN	**0.072**	**0.134**	**0.096**	**0.018**	**0.062**	**0.133**	**0.308**
SGD	−0.161	−0.176	−0.027	**0.221**	**0.430**	**0.327**	**0.305**
DT	**0.060**	−0.007	−0.078	−0.172	−0.063	**0.100**	**0.795**
LR	**0.116**	**0.093**	−0.021	**0.015**	−0.115	**0.001**	**0.174**
NB	**0.122**	−0.141	−0.220	−0.264	−0.586	−0.343	−0.652
MLP	**0.010**	−0.032	−0.039	−0.189	−0.235	−0.568	−0.097
AB	−0.007	**0.113**	**0.028**	**0.004**	**0.137**	−0.035	**0.158**
ET	**0.026**	**0.008**	**0.173**	**0.553**	**0.045**	**0.048**	−0.136
GB	−0.017	**0.016**	**0.006**	**0.010**	**0.174**	**0.115**	**0.401**
DnW	**0.015**	**0.052**	−0.014	**0.048**	**0.016**	**0.043**	**0.020**
Bag	−0.017	−0.015	−0.035	**0.004**	−0.009	**0.101**	**0.294**
average	**0.010**	**0.007**	**0.006**	**0.003**	**0.006**	−0.005	**0.124**

(d) GBA-Δ: *Impact* differences for the UCI dataset.

Table X: Training Times. For each dataset and PD, we report the time (in seconds) required to train its baseline and POC variants.

Dataset	LNU-Phish		DeltaPhish		Mendeley		UCI	
Classifier	Base	POC	Base	POC	Base	POC	Base	POC
RF	0.46	1.03	0.50	0.25	0.59	0.24	0.41	0.36
SVM	1.15	2.66	0.10	0.11	0.75	0.64	0.52	0.77
KNN	0.06	0.03	0.02	0.01	0.01	0.01	0.11	0.06
SGD	0.22	0.19	0.01	0.02	0.15	0.06	0.14	0.13
DT	0.07	0.24	0.02	0.04	0.03	0.01	0.02	0.04
LR	0.11	0.30	0.07	0.03	0.89	0.64	0.12	0.07
NB	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01
MLP	18.7	22.7	8.03	4.71	15.7	13.6	14.2	28.9
AB	0.56	0.94	1.24	1.61	2.57	1.44	0.25	0.25
ET	1.32	1.63	0.38	0.38	0.11	0.11	1.13	1.22
GB	2.14	29.7	2.16	3.69	3.33	4.19	5.59	6.79
DnW	53.4	62.4	21.9	12.7	22.7	18.2	25.0	35.4
Bag	0.94	1.73	1.24	0.4	1.38	1.31	1.45	1.28

POC requires slightly more time on LNU-Phish and UCI, but slightly less time on Mendeley and DeltaPhish. We reiterate that POC must be trained. If random hyper-parameters are used, it is unlikely to yield great performance (i.e., low false positives and high true positives). The results in Table X denote the time required to train the 'best' configuration of POC after our extensive grid-search optimization. Real-world deployments must train. Fortunately, training only needs to be done infrequently (e.g. when hackers change their phishing methods and retraining is needed to prevent concept-drift [78]).

VII. Discussion

We highlight the key findings from our huge experimental analysis by providing a formal statistical analysis of our results, as well as an in-depth assessment of a pragmatic application of POC, showing its pros and cons.

A. Statistical Analysis

We conduct a statistical analysis of our results with the goal of answering three questions:

1) is the slight performance drop of POC in the no-attack case significant?
2) is the *Impact* of the considered attacks on the baseline classifiers significant?
3) does POC provide better protection against such attacks than the baseline classifiers?

To answer all these questions, we rely on the Wilcoxon Signed Rank test which performs a pairwise comparison of the samples of two populations. The output of the test is a p-value that provides the probability that the two populations were generated by the same underlying process: if the resulting p-value is *higher* (resp. lower) than a given target threshold α, then the two populations can be considered to be statistically equivalent (resp. different). Typically, α is chosen to be 0.05, meaning the chance of a correct claim is 95%.

1) No-attack case performance: We statistically compare the populations of the Recall (i.e., detection rate) and F1-score achieved by the baseline and POC classifiers in the no attack case; all populations consist of 52 elements (given by 13 classifiers and 4 datasets). The resulting p-value of these comparisons is 0.37 for the Recall, and 0.23 for the F1-score. Both p-values are much higher than α, meaning that the populations in both tests can be considered to be statistically equivalent. Therefore, the performance drop of POC is negligible. The reason for this is that our POC implementation in Section VI assumes complete *prevalence*—meaning that the baseline and POC classifiers use the same amount of information to perform their inference (but the POC variants maps such information in a different space).

2) Impact Assessment: we compare the populations containing the Recall *before* and *after* the execution of each Gray Box attack (hence, the populations have 52 elements for each comparison) on the 'baseline' PD. The resulting p-value are not only *always* lower than α, but are also almost always equal to 0—the only exception is for GBA-Δ with Δ=10%, which has a p-value of 0.00003. Therefore, all attacks induce a statistically significant drop in the baseline detection rate. This motivates the search for a solution that mitigates such *Impact*.

3) Protection of POC: our experiments suggest that hardening classifiers with POC yields results that are superior to those of their baseline variants, but in some cases, the difference is small (see Figure 3) and in other cases the baselines are better (i.e., the negative values in Tables VIII and IX). Hence, answering the third question requires a more fine-grained investigation. For each dataset, we compare two populations containing the *Impact* of the considered attacks (GBA-1-GBA-3, and GBA-Δ in its 7 variants—10 attacks in all). The first population represents the *Impact* against the baseline classifiers, and the second population represents the *Impact* against the POC versions of the classifiers. Hence, each population has 130 samples (13 Clf * 10 Att). Since we distinguish the populations on a per-dataset basis, we apply the Bonferroni Correction, thus resulting in a target α=0.0125 (because we are considering 4 different scenarios, one per dataset). Table XI shows the resulting p-values and Effect Sizes of the test. We see that all p-values are below our target α=0.01. Furthermore, the different Effect Sizes also confirm the low chance that the two populations were generated by the same underlying stochastic process. The results confirm that using POC yields more resilient classifiers against the Gray Box attacks considered in this paper.

Table XI: Statistical comparison of the *Impact* against the baselines and POC on each dataset (via a Wilcoxon Signed-Rank test).

Metric	LNU-Phish	DeltaPhish	Mendeley	UCI
p-value	< 0.0001	0.0005	< 0.0001	< 0.0001
Effect Size	0.2470	0.3256	0.1345	0.2696

Intuitively, POC is effective[24] against our Gray Box attacks because the baseline PDs use 'fixed' features whose modifications result in highly distinct samples. In contrast, when using POC, the combination of feature mapping and mixing leads to 'smoother' feature modifications that do not result in samples deviating greatly from their unmodified variants. Let us explain this with an example. Suppose a sample x is described by (among others) two binary features f_1 and f_2, so that $f_1(x)$=1 and $f_2(x)$=1. Assume an attack that modifies the value of f_1 from 1 to 0. This translates to an (adversarial) sample $\bar{x}$ whose value of f_1 is 'the opposite' of its original variant x. Now consider an implementation of POC that, in its Ψ, has an oc that sums the two 'original features' f_1 and f_2, implying that its application to x results in $oc(x)$=2. If the attacker modifies f_1 of sample x from 1 to 0, then *after applying* POC, this modification would result in $oc(\bar{x})$=1 which is 'less' different from its original variant. To achieve the same effects, the attacker must modify both f_1 and f_2. This may therefore lead to a more accurate classification.

We note that the 'favorable' results in Table XI are mostly due to the good hardening performance of POC against the simple Gray Box Attacks, i.e., GBA-1–GBA-3 (shown in Table VIII). The hardening provided by POC against the complex attacks of GBA-Δ (shown in Table IX) is smaller. Despite this, the next section showcases a pragmatic use-case which shows the low-level benefits of POC.

B. Pragmatic Use Case

We evaluated a huge number of classifiers in different conditions. However, in reality only a single classifier is used as PD—and the choice is made depending on its performance at training-time (i.e., in the no-attack scenario). Hence, we now investigate a pragmatic use case of POC, where we analyze its benefits and tradeoffs when applied to 'harden' the best classifier for each dataset (according to Table IV). Figure 5 shows the Recall of the best baseline classifier alongside its POC variant when they are subject to the Gray Box attacks considered in our paper. Each subfigure focuses on a

[24]The classifiers are more resilient, but we do not claim that POC yields PDs that are immune to such attacks!

specific dataset, and the red line in each subfigure reports the detection rate in the no-attack case. We analyze these subfigures, and then make some final recommendations.

1) LNU-Phish analysis: Figure 5a focuses on the LNU-Phish dataset, where the best classifier is GB which obtains near perfect performance—a result shared by its POC variant. However, we can see that the latter is significantly more robust against GBA-1 as well as against two GBA-Δ (where $\Delta = 60\%$ or 70%), as the Recall is above 10% superior. Against all other attacks, the detection rate is either equivalent, or marginally superior than the baseline (up to 5% increased Recall).

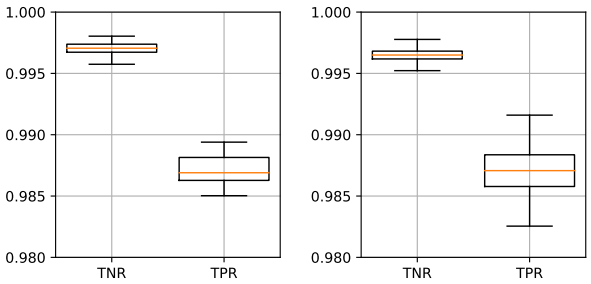
2) DeltaPhish analysis: Figure 5b focuses on the DeltaPhish dataset, where the best baseline classifier is also GB, whose POC variant has only a 0.01 less F1-score in the no-attack case. On this dataset, the performance of POC is slightly inferior to the baseline against all the GBA-Δ attacks, but significant differences arise in the simple attacks: for GBA-1 and GBA-3, the baseline does not detect any attack whereas POC can detect a small amount (8%). In contrast, GBA-2 barely affects the baseline GB but half of its samples can evade POC. This is the most significant 'defeat' of POC—although its application on other strong baselines can be beneficial (e.g., for the deep learning MLP classifier, the Recall against GBA-2 of POC is 0.86 vs 0.77).

3) Mendeley analysis: Figure 5c focuses on the Mendeley dataset, where the best baseline classifier is ET. POC provides a significant mitigation (above 10% better Recall) against 8 out of 10 attacks: the only exceptions are GBA-Δ with $\Delta = 20\%$, where the improvement is of smaller entity (4%), and GBA-2 where it is slightly worse than the baseline (by about 5%). Specifically, POC is barely affected by GBA-1, as it can successfully detect this attack with 0.85 Recall against the 0.37 of the baseline. All these benefits come at the 'cost' of a 0.03 reduction in F1-score when no adversarial attack occurs.

4) UCI analysis: Figure 5d focuses on the UCI dataset, where the best baseline classifier is ET, whose hardened POC variant achieves the same performance in the absence of attacks. We note that POC exhibits weaker Recall (around 8%) than the baseline only against GBA-Δ with $\Delta = 70\%$. *In all other cases, POC is superior.* Noteworthy are the successes against GBA-Δ with $\Delta = 40\%$, where POC has a Recall of over 90% against the 40% of the baseline; and also against GBA-3, as the baseline cannot detect *any* attack, whereas POC can detect above 40%.

C. POC Without Training

We assess the performance of POC when it is applied without training (i.e. without using an optimal choice of Ψ). To make the analysis humanly feasible, we focus on our proposed LNU-Phish dataset, for which we consider the 'best' baseline classifier: GB. We then "blindly" apply POC 100 times to this baseline and then we re-do the experiment by applying POC 1000 times. We perform this experiments with no training, and by using the same configuration parameters described in Section VI (i.e., having $\mathcal{P} = 100\%$). The performance is measured by computing the TPR and TNR (i.e., 1-FPR) on the test-set (i.e., 20% of LNU-Phish). The results are shown in Figures 4.



(a) Results of 100 'blind' Ψ. (b) Results of 1000 'blind' Ψ.

Figure 4: Distribution of TNR and TPR achieved by 'blindly' applying POC to GB on LNU-Phish (outliers are not shown).

The boxplots show that POC yields practical performance even without training: the high TNR (always above 0.99) denotes low rates of false alarms, whereas the high TPR (always above 0.98) shows that phishing webpages are ably detected. It is encouraging that the distribution barely changes despite going from 100 to 1000 trials. [25].

D. Takeaway Message

By taking into account all above observations, we can draw the following conclusions. When used to harden the best classifier, POC is a pragmatic solution against our Gray Box Attacks in 3 of 4 datasets. This is because POC provides better (or same) adversarial robustness, but does not induce a significant performance drop in the absence of adversarial attacks. In contrast, on the DeltaPhish dataset, using POC to harden the best baseline PD is not recommended: although it can detect some instances of GBA-1 and GBA-3 (against none of the

[25]Of course, we do not claim this to be valid 'anywhere-anytime', as such experiments focus on just a single configuration of a classifier (GB) on a single dataset (LNU-Phish)

baseline), the other results cannot justify its application to harden the ET classifier on this dataset. This is due to the specialized nature of the DeltaPhish dataset: while the other three datasets have malicious samples corresponding to 'general' phishing webpages, the DeltaPhish dataset captures a *specific* type of phishing attack. The entries in DeltaPhish are 'legitimate' webpages that have fallen under the control of an attacker, which are different from phishing pages that are specifically created by an attacker. Hence, the high specificity of this dataset may yield a suboptimal hardening by POC (on the very best baseline PD) against the proposed Gray Box Attacks.

Finally, we remark that POC has—by definition—the additional benefit of yielding PDs that are hard to reverse engineer. This increases the difficulty of launching model stealing attacks, such as those conducted against the Google's Chrome phishing filter in [76]. Furthermore, even if an attacker were able to fully 'crack' a POC-hardened PD and infer its feature set, the attacker must repeat the process when the PD is periodically updated (to mitigate concept-drift [40]) with more recent data, as the feature mapping is likely to change.

VIII. Experiments: POC without complete prevalence

As a final contribution of this paper, we evaluate the effectiveness of POC when $\mathcal{P}(\mathbb{F},\Psi) < 100\%$, meaning that some features of $\mathbb{F}$ are not included in any oc composing Ψ. The expectation is that the robustness against attacks will increase (due to the 'explicit' feature removal), but the performance in the absence of adversarial attacks will decrease because some information is lost (cf. Section V-B).

A. Experimental Settings

For simplicity, we perform experiments only on the LNU-Phish dataset, where we consider the classifier yielding the best 'baseline' PD—specifically, the GB classifier (cf. Section VII-B1). The experimental settings are exactly the same as those described in Section VI-A, but we do not require that $\mathcal{P}(\mathbb{F},\Psi)=100\%$. In particular, we assess POC for different $\mathcal{P}$. Hence, we apply POC so that the resulting $\mathcal{P}(\mathbb{F},\Psi)$ falls within 6 values ranging from 65% to 90% (at 5% increments). As an example, since the LNU-Phish dataset contains 27 features (cf. Table II), when $\mathcal{P}(\mathbb{F},\Psi)=70\%$ it means that the POC-hardened PD uses 19 features (across all its oc).

B. Results

We evaluate all such POC-hardened variants of the GB classifier both in the absence of attacks and against all the 10 adversarial attacks considered in our paper, and report the results in Figures 6.

Figure 6a shows the *false positive rate* as a function of $\mathcal{P}$; where the two leftmost bars report the FPR of the 'baseline' GB and the POC-hardened GB with complete *prevalence* (from Section VI). In contrast, Figure 6b shows the *detection rate* against all the considered adversarial attacks (on the horizontal axis), as well as in the no-attack case (the leftmost value); the dotted lines represent the results reported in Section VI (included for comparison), whereas full lines represent the POC-hardened PDs with varying *prevalence*.

From Figure 6 we can see that—in the absence of attacks—the performance of POC when $\mathcal{P} \leq 90\%$ is worse with respect to the results shown in Section VI-B. Indeed, when no attacks occur, the FPR (Figure 6a) is higher and the detection rate (leftmost value in Figure 6b) is also inferior. This is due to the loss of information induced when $\mathcal{P}<100\%$. However, such increased FPR is ably compensated by the greater detection rate in the presence of adversarial attacks. As shown in Figure 6b. with the sole exception of GBA-Δ where Δ=10 or 20%, the full lines denote better results than the dotted lines. As an example, when $\mathcal{P}=70\%$, the corresponding PD is not affected at all by the simple attacks, and its detection rate never goes below 83%—but, it also achieves the greatest FPR (0.073 according to Figure 6a).

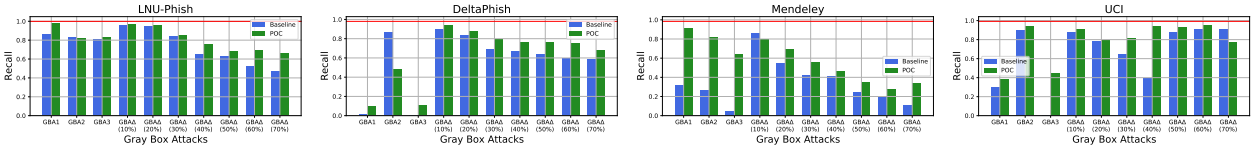
In summary, these results match our expectations. From a practical perspective, using POC without complete *prevalence* is beneficial if a PD is likely to be targeted by the proposed Gray Box attacks, and if the deployment setting can accept a slightly worse performance in the absence of such attacks.

IX. Conclusions

It is clear from ProofPoint's 2020 "State of the Phish" report that despite decades of work to counter phishing attacks, phishing represents a major attack vector for malicious hackers.

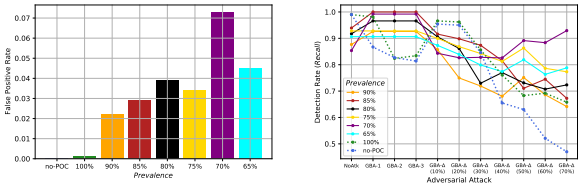
In this paper, we propose a series of complex and simple Gray Box attacks on existing machine learning based classifiers for phishing website detection. We formally define the Impact of an attack on a dataset and classifier in terms of the percentage drop in predictive performance and show that these attacks cause a significant drop in performance of past work using ML classifiers.

We develop the POC algorithm that uses a mix of randomization (to reduce the probability that the adversary can guess the features used) and feature transformation (to further reduce this probability). We show that POC—despite not representing a universal panacea



(a) LNU-Phish. Best baseline: GB (F1-score: 0.99, and 0.99 for POC). (b) DeltaPhish. Best baseline: GB (F1-score: 0.99, and 0.98 for POC). (c) Mendeley. Best baseline: ET (F1-score: 0.99, and 0.96 for POC). (d) UCI. Best baseline: ET (F1-score: 0.99, and 0.99 for POC)

Figure 5: Effectiveness of attacks (as measured via the Recall) against the best baseline PD and its POC-hardened version on each dataset. The red line on each subfigure is the Recall in the absence of attacks. The caption of each subfigure reports the F1-score in the no-attack case.



(a) False Positive Rate. (b) Detection Rate.

Figure 6: Performance of POC with varying *prevalence*.

against adversarial attacks—is more robust against all the considered Gray Box attacks than past classifiers, and does not degrade their performance in the absence of such attacks.

Our paper considers 13 classifiers (including new classifiers such as Google's Deep & Wide that have not been used previously for phishing detectors to the best of our knowledge) using 4 datasets (including the new LNU-Phish dataset that we release as an additional contribution of this paper). In contrast, most past work on adversarial phishing detectors consider only one dataset and one classifier.

Acknowledgements. We thank the anonymous referees for their excellent comments. We are also grateful to ONR grant N00014-20-1-2407.

REFERENCES

[1] D. Maiorca, B. Biggio, and G. Giacinto, "Towards adversarial malware detection: lessons learned from PDF-based attacks," *ACM Comp. Surv.*, vol. 52, no. 4, pp. 1–36, 2019.

[2] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tut.*, vol. 18, no. 2, pp. 1153–1176, 2015.

[3] M. Mayhew, M. Atighetchi, A. Adler, and R. Greenstadt, "Use of machine learning in big data analytics for insider threat detection," in *IEEE Conf. Milit. Comm.*, 2015, pp. 915–922.

[4] M. Zhao, B. An, W. Gao, and T. Zhang, "Efficient label contamination attacks against black-box learning models," in *Int. Joint Conf. Artif. Intell.*, 2017, pp. 3945–3951.

[5] A. Subasi, E. Molah, F. Almkallawi, and T. J. Chaudhery, "Intelligent phishing website detection using random forest classifier," in *IEEE Int. Conf. Elec. Comput. Tech. Appl.*, 2017, pp. 1–5.

[6] T. W. Moore and R. Clayton, "The impact of public information on phishing attack and defense," *Communications and Strategies*, no. 81, pp. 45–68, 2011.

[7] R. Basnet, "Learning to Detect Phishing URLs," *Int. J. Res. Eng. Tech.*, vol. 03, pp. 11–24, 2014.

[8] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection based associative classification data mining," *Elsevier Expert Syst. Appl.*, vol. 41, no. 13, pp. 5948–5959, 2014.

[9] N. Abdelhamid, F. Thabtah, and H. Abdel-jaber, "Phishing detection: A recent intelligent machine learning comparison based on models content and features," in *Proc. IEEE Int. Conf. Intel. Secur. Inform.*, 2017, pp. 72–77.

[10] R. Verma and K. Dyer, "On the character of phishing urls: Accurate and robust statistical learning classifiers," in *Proc. ACM Conf. Data Appl. Secur. Privacy*, 2015, pp. 111–122.

[11] S. C. Jeeva and E. B. Rajsingh, "Intelligent phishing url detection using association rule mining," *Springer Hum-Cent. Comput. Info.*, vol. 6, no. 1, p. 10, 2016.

[12] C. L. Tan, K. L. Chiew, K. Wong *et al.*, "Phishwho: Phishing webpage detection via identity keywords extraction and target domain name finder," *Elsevier Decis. Support Syst.*, vol. 88, pp. 18–27, 2016.

[13] A. Niakanlahiji, B.-T. Chu, and E. Al-Shaer, "Phishmon: A machine learning framework for detecting phishing webpages," in *Proc. IEEE Int. Conf. Intel. Secur. Inf.*, 2018, pp. 220–225.

[14] W. Ali, "Phishing website detection based on supervised machine learning with wrapper features selection," *Int. J. Adv. Comp. Sci. Appl.*, vol. 8, no. 9, pp. 72–78, 2017.

[15] E. Lancaster, T. Chakraborty, and V. Subrahmanian, "Malt^p: Parallel prediction of malicious tweets," *IEEE T. Computational Social Systems*, vol. 5, no. 4, pp. 1096–1108, 2018.

[16] D. L. Cook, V. K. Gurbani, and M. Daniluk, "Phishwish: a stateless phishing filter using minimal rules," in *Proc. Springer Int. Conf. Financ. Crypt. Data Secur.*, 2008, pp. 182–186.

[17] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *Proc. ACM Int. Conf. World Wide Web*, 2007, pp. 639–648.

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

[19] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proc. ACM Int. Conf. Secur. Privacy Commun. Netw.*, 2008, p. 22.

[20] M. Hara, A. Yamada, and Y. Miyake, "Visual similarity-based phishing detection without victim site information," in *Proc. IEEE Symp. Comput. Intel. Cyber Secur.*, 2009, pp. 30–36.

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

[22] G. Liu, B. Qiu, and L. Wenyin, "Automatic detection of phishing target from phishing webpage," in *Proc. IEEE Int. Conf. Pattern Recogn.*, 2010, pp. 4153–4156.

[23] H. Zhang, G. Liu, T. W. Chow, and W. Liu, "Textual and visual content-based anti-phishing: a bayesian approach," *IEEE T. Neural Netw.*, vol. 22, no. 10, pp. 1532–1546, 2011.

[24] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," Google AI, Tech. Rep., 2010.

[25] L. F. Cranor, S. Egelman, J. I. Hong, and Y. Zhang, "Phinding phish: An evaluation of anti-phishing toolbars." in *Netw. Distrib. Syst. Symp.*, 2007, pp. 1–19.

[26] A. Bergholz, J. De Beer, S. Glahn, M.-F. Moens, G. Paaß, and S. Strobel, "New filtering approaches for phishing email," *J. Comp. Secur.*, vol. 18, no. 1, pp. 7–35, 2010.

[27] F. Toolan and J. Carthy, "Phishing detection using classifier ensembles," in *IEEE eCrime Researchers Summit*, 2009, pp. 1–9.

[28] H. Kettani and P. Wainwright, "On the Top Threats to CyberSystems," in *IEEE Int. Conf. Inf. Comp. Tech.*, 2019, pp. 175–179.

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

[30] I. Corona, B. Biggio, M. Contini, L. Piras, R. Corda, M. Mereu, G. Mureddu, D. Ariu, and F. Roli, "Deltaphish: Detecting phishing webpages in compromised websites," in *Proc. Springer Europ. Symp. Res. Comput. Secur.*, 2017, pp. 370–388.

[31] M. Babagoli, M. P. Aghababa, and V. Solouk, "Heuristic nonlinear regression strategy for detecting phishing websites," *Springer Soft Comput.*, vol. 23, no. 12, pp. 4315–4327, 2019.

[32] A. K. Jain and B. B. Gupta, "Towards detection of phishing websites on client-side using machine learning based approach," *Springer Telecom. Syst.*, vol. 68, no. 4, pp. 687–700, 2018.

[33] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from urls," *Elsevier Expert Syst. Appl.*, vol. 117, pp. 345–357, 2019.

[34] "Deltaphish dataset," https://www.pluribus-one.it/it/chi-siamo/blog/84-cybersecurity/77-deltaphish, accessed: Sept. 2020.

[35] "Mendeley phishing dataset," https://data.mendeley.com/datasets/h3cgnj8hft/1, accessed: Sept. 2020.

[36] "Uci phishing websites dataset," https://archive.ics.uci.edu/ml/datasets/phishing+websites, accessed: Sept. 2020.

[37] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2009, pp. 1245–1254.

[38] W. Wang and K. Shirley, "Breaking bad: Detecting malicious domains using word segmentation," *arXiv: 1506.04111*, 2015.

[39] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *Proc. ACM Workshop Recurring Malcode*, 2007, pp. 1–8.

[40] K. Tian, S. T. Jan, H. Hu, D. Yao, and G. Wang, "Needle in a haystack: Tracking down elite phishing domains in the wild," in *Proc. Internet Measurement Conf.*, 2018, pp. 429–442.

[41] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.

[42] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE Euro Symp. Secur. Privacy*, 2016, pp. 372–387.

[43] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv 1412.6572*, 2014.

[44] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. ACM Workshop Artif. Intel. Secur.*, 2017, pp. 15–26.

[45] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evolut. Comput.*, 2019.

[46] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *USENIX Secur. Symp.*, 2016, pp. 513–530.

[47] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," in *Proc. Conf. Empiric. Methods Natur. Lang. Process.*, 2017, pp. 2021–2031.

[48] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Elsevier Pattern Recogn.*, vol. 84, pp. 317–331, 2018.

[49] Z. Katzir and Y. Elovici, "Quantifying the resilience of machine learning classifiers used for cyber security," *Elsevier Expert Syst. Appl.*, vol. 92, pp. 419–429, 2018.

[50] M. Kantarcioglu and B. Xi, "Adversarial data mining: Big data meets cyber security," in *Proc. ACM Conf. Comput. Comm. Secur.*, 2016, pp. 1866–1867.

[51] Y. Shi and Y. E. Sagduyu, "Evasion and causative attacks with adversarial deep learning," in *Proc. IEEE Conf. Military Comm.*, 2017, pp. 243–248.

[52] Y. Vorobeychik and M. Kantarcioglu, "Adversarial machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–169, 2018.

[53] A. Warzyński and G. Kołaczek, "Intrusion detection systems vulnerability on adversarial examples," in *Proc. IEEE Conf. Innovations Intell. Syst. Appl.*, Jul. 2018, pp. 1–4.

[54] S. Chen, M. Xue, L. Fan, S. Hao, L. Xu, H. Zhu, and B. Li, "Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach," *Elsevier Comp. Secur.*, vol. 73, pp. 326–344, 2018.

[55] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proc. ACM Workshop Artif. Intel. Secur.*, Nov. 2017, pp. 27–38.

[56] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *Proc. Springer Europ. Symp. Res. Comput. Secur.*, 2017, pp. 62–79.

[57] I. Rosenberg, A. Shabtai, L. Rokach, and Y. Elovici, "Generic black-box end-to-end attack against state of the art api call based malware classifiers," in *Proc. Springer Int. Symp. Res. Attacks, Intrusions and Defenses*, 2018, pp. 490–510.

[58] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 506–519.

[59] H. Dang, Y. Huang, and E.-C. Chang, "Evading classifiers by morphing in the dark," in *Proc. ACM Conf. Comp. Commun. Secur.*, 2017, pp. 119–133.

[60] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proc. Netw. Distrib. Syst. Symp.*, 2016, pp. 21–24.

[61] B. Li and Y. Vorobeychik, "Feature cross-substitution in adversarial classification," in *Proc. Advances Neur. Inf. Process. Syst. Conf.*, 2014, pp. 2087–2095.

[62] C. Bai, Q. Han, G. Mezzour, F. Pierazzi, and V. Subrahmanian, "Dbank: Predictive behavioral analysis of recent android banking trojans," *IEEE T. Depend. Secur.*, 2019.

[63] T. Chakraborty, F. Pierazzi, and V. Subrahmanian, "Ec2: ensemble clustering and classification for predicting android malware families," *IEEE T. Depend. Secur.*, 2017.

[64] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! a case study on android malware detection," *IEEE T. Depend. Secure.*, 2017.

[65] G. Apruzzese and M. Colajanni, "Evading botnet detectors based on flows and random forest with adversarial samples," in *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, 2018, pp. 1–8.

[66] H. S. Anderson, J. Woodbridge, and B. Filar, "Deepdga: Adversarially-tuned domain generation and detection," in *Proc. ACM Workshop Artif. Intell. Secur.*, 2016, pp. 13–21.

[67] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "Enabling fair ml evaluations for security," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2018, pp. 2264–2266.

[68] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware c&c detection: A survey," *ACM Comput. Surv.*, vol. 49, no. 3, p. 59, 2016.

[69] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. 2016 IEEE Symp. Secur. Privacy*, 2016, pp. 582–597.

[70] A. Kantchelian, J. Tygar, and A. Joseph, "Evasion and hardening of tree ensemble classifiers," in *Int. Conf. Machin. Learn.*, 2016, pp. 2387–2396.
[71] B. Biggio, I. Corona, Z.-M. He, P. P. Chan, G. Giacinto, D. S. Yeung, and F. Roli, "One-and-a-half-class multiple classifier systems for secure learning against evasion attacks at test time," in *Proc. Springer Int. Workshop Multiple Classifier Syst.*, 2015, pp. 168–180.
[72] P. Russu, A. Demontis, B. Biggio, G. Fumera, and F. Roli, "Secure kernel machines against evasion attacks," in *Proc. ACM Workshop Artif. Intel. Secur.*, 2016, pp. 59–69.
[73] Z. He, J. Su, M. Hu, G. Wen, S. Xu, and F. Zhang, "Robust support vector machines against evasion attacks by random generated malicious samples," in *Proc. IEEE Int. Conf. Wavelet Anal. Pattern Recogn.*, 2017, pp. 243–247.
[74] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE T. Knowl. Data En.*, vol. 26, no. 4, pp. 984–996, 2013.
[75] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, and M. Colajanni, "Modeling realistic adversarial attacks against network intrusion detection systems," *ACM Digital Threats: Research and Practice*, 2021.
[76] B. Liang, M. Su, W. You, W. Shi, and G. Yang, "Cracking classifiers for evasion: a case study on the Google's phishing pages filter," in *Proc. Int. Conf. World Wide Web*, 2016, pp. 345–356.
[77] S. Marchal, J. Francois, R. State, and T. Engel, "Phishstorm: Detecting phishing with streaming analytics," *IEEE T. Netw. Serv. Manag.*, vol. 11, no. 4, pp. 458–471, 2014.
[78] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 625–642.
[79] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proc. ACM Workshop on Deep Learn. for Recommender Syst.*, Sep. 2016, pp. 7–10.
[80] R. B. Basnet and T. Doleck, "Towards developing a tool to detect phishing urls: a machine learning approach," in *Proc. IEEE Int. Conf. Comput. Intel. Commun. Techn.*, Feb. 2015, pp. 220–223.
[81] C. Smutz and A. Stavrou, "Malicious pdf detection using metadata and structural features," in *Proc. ACM Conf. Comput. Secur. Appl.*, Dec. 2012, pp. 239–248.
[82] G. Apruzzese, M. Colajanni, L. Ferretti, and M. Marchetti, "Addressing adversarial attacks against security systems based on machine learning," in *Proc. IEEE Int. Conf. Cyber Conflicts*, 2019, pp. 1–18.
[83] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *Proc. IEEE Symp. Secur. Priv.*, 2018, pp. 36–52.

Giovanni Apruzzese is an Assistant Professor within the Hilti Chair of Data and Application Security at the University of Liechtenstein since 2022, and was previously a Post-Doc at the same institution since 2020. He received the PhD Degree and the Master's Degree in Computer Engineering (summa cum laude) in 2020 and 2016 respectively at the Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Italy. In 2019 he spent 6 months as a Visiting Researcher at Dartmouth College (Hanover, NH, USA) under the supervision of Prof. V.S. Subrahmanian. His research interests involve all aspects of big data security analytics with a focus on machine learning, and his main expertise lies in the analysis of Network Intrusions, Phishing, and Adversarial Attacks.

Homepage: https://gioapru.github.io

V.S. Subrahmanian is the Walter P. Murphy Professor of Computer Science and Buffett Faculty Fellow in the Buffett Institute of Global Affairs at Northwestern University. He was previously the Dartmouth College Distinguished Professor in Cybersecurity, Technology, and Society and Director of the Institute for Security, Technology, and Society at Dartmouth. Before that, he was a Professor of Computer Science at the University of Maryland from 1989-2017 where he also served for 6+ years as Director of the University of Maryland's Institute for Advanced Computer Studies. Prof. Subrahmanian is an expert on big data analytics including methods to analyze text/geospatial/relational/social network data, learn behavioral models from the data, forecast actions, and influence behaviors with applications to cybersecurity and counterterrorism. He has written five books, edited ten, and published over 300 refereed articles. He is a Fellow of the American Association for the Advancement of Science and the Association for the Advancement of Artificial Intelligence and received numerous other honors and awards. His work has been featured in numerous outlets such as the Baltimore Sun, the Economist, Science, Nature, the Washington Post, American Public Media. He serves on the editorial boards of numerous journals including Science, the Board of Directors of the Development Gateway Foundation (set up by the World Bank), SentiMetrix, Inc., and on the Research Advisory Board of Tata Consultancy Services. He previously served on DARPA's Executive Advisory Council on Advanced Logistics and as an ad-hoc member of the US Air Force Science Advisory Board.

Homepage: https://vssubrah.github.io/