SpacePhish: The Evasion-space of Adversarial Attacks against Phishing Website Detectors using Machine Learning

Giovanni Apruzzese*
giovanni.apruzzese@uni.li
Institute of Information Systems
University of Liechtenstein

Mauro Conti†
conti@unipd.it
Department of Computer Science
Delft University of Technology, NL

Ying Yuan*
ying.yuan@math.unipd.it
Department of Mathematics
† University of Padua, Italy

ABSTRACT

Existing literature on adversarial Machine Learning (ML) focuses either on showing attacks that break every ML model, or defenses that withstand most attacks. Unfortunately, little consideration is given to the actual *cost* of the attack or the defense. Moreover, adversarial samples are often crafted in the "feature-space", making the corresponding evaluations of questionable value. Simply put, the current situation does not allow to estimate the actual threat posed by adversarial attacks, leading to a lack of secure ML systems.

We aim to clarify such confusion in this paper. By considering the application of ML for Phishing Website Detection (PWD), we formalize the "evasion-space" in which an adversarial perturbation can be introduced to fool a ML-PWD—demonstrating that even perturbations in the "feature-space" are useful. Then, we propose a realistic threat model describing evasion attacks against ML-PWD that are cheap to stage, and hence intrinsically more attractive for real phishers. Finally, we perform the first statistically validated assessment of state-of-the-art ML-PWD against 12 evasion attacks. Our evaluation shows (i) the true efficacy of evasion attempts that are more likely to occur; and (ii) the impact of perturbations crafted in different evasion-spaces. Our realistic evasion attempts induce a statistically significant degradation (3–10% at p <0.05), and their cheap cost makes them a subtle threat. Notably, however, some ML-PWD are immune to our most realistic attacks (p=0.22). Our contribution paves the way for a much needed re-assessment of adversarial attacks against ML systems for cybersecurity.

KEYWORDS

Machine Learning, Phishing, Website, Adversarial Attacks

ACM Reference Format:
Giovanni Apruzzese, Mauro Conti, and Ying Yuan. 2022. SpacePhish: The Evasion-space of Adversarial Attacks against Phishing Website Detectors using Machine Learning. In *Annual Computer Security Applications Conference (ACSAC '22), December 5–9, 2022, Austin, TX, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3564625.3567980

ACSAC '22, December 5–9, 2022, Austin, TX, USA

ACM ISBN 978-1-4503-9759-9/22/12…$15.00
https://doi.org/10.1145/3564625.3567980

1 INTRODUCTION

After more than a decade of research [21] and thousands of papers [4], it is well-known that Machine Learning (ML) methods are vulnerable to adversarial attacks. Specifically, by introducing imperceptible perturbations (down to a single pixel or byte [13, 82]) in the input data, it is possible to compromise the predictions made by a ML model. Such vulnerability, however, is more dangerous in settings that implicitly assume the presence of adversaries. A cat will not try to fool a ML model. An attacker, in contrast, will actively try to evade a ML detector—the focus of this paper.

On the surface, the situation portrayed in research is vexing. The confirmed successes of ML [48] are leading to large-scale deployment of ML in production settings (e.g., [31, 76, 84]). At the same time, however, dozens of papers showcase adversarial attacks that can crack 'any' ML-based detector (e.g., [14, 57]). Although some papers propose countermeasures (e.g., [72]), they are quickly defeated (e.g., [28]), and typically decrease the baseline performance (e.g. [14, 32]). As a result, recent reports [35, 53] focusing on the integration of ML *in practice* reveal that: "I Never Thought About Securing My Machine Learning Systems" [23]. This is not surprising: if ML can be so easily broken, then why invest resources in increasing its security through –unreliable– defenses?

Sovereign entities (e.g., [2, 3]) are endorsing the development of "trustworthy" ML systems; yet, any enhancement should be economically justified. No system is foolproof (ML-based or not [26]), and guaranteeing protection against omnipotent attackers is an enticing but unattainable objective. In our case, a security system should increase the *cost* incurred by an attacker to achieve their goal [61]. Real attackers have a cost/benefit mindset [93]: they may try to evade a detector, but only if doing so yields positive returns. In reality, worst-case scenarios are an exception—not the norm.

Our paper is inspired by several recent works that pointed out some 'inconsistencies' in the adversarial attacks carried out by prior studies. Pierazzi et al. [73] observe that real attackers operate in the "problem-space", i.e., the perturbations they can introduce are subject to physical constraints. If such constraints are not met, and hence the perturbation is introduced in the "feature-space" (e.g., [63]), then there is a risk of generating an adversarial example that is not physically realizable [86]. Apruzzese et al. [12], however, highlight that even 'impossible' perturbations can be applied, but *only if* the attacker has internal access to the data-processing pipeline of the target system. Nonetheless, Biggio and Roli suggest that ML security should focus on "anticipating the most likely

* Equal Contribution. Contact Ying Yuan for correspondence.

threats" [21]. Only *after* proactively assessing the impact of such threats a suitable countermeasure can be developed—if required.

We aim to promote the development (and deployment) of secure ML systems. However, meeting Biggio and Roli's recommendation presents two tough challenges for research papers. First, it is necessary to devise a *realistic threat model* which portrays adversarial attacks that are not only physically realizable, but also economically viable. Devising such a threat model, however, requires a detailed security analysis of the *specific cyberthreat* addressed by the detector—while factoring the resources that attackers are willing to invest. Second, it is necessary to *evaluate the impact* of the attack by crafting the corresponding perturbations. Doing so is difficult if the threat model assumes an attacker operating in the problem-space, because such *perturbations must be applied on raw-data*, i.e., before any preprocessing occurs—which is hard to find.

In this paper, we tackle both of these challenges. In particular, we focus on ML-systems for Phishing Website Detection (PWD). Countering phishing – still a major threat today [7, 49] – is an endless struggle. Blocklists can be easily evaded [85], and to cope against adaptive attackers some detectors are equipped with ML (e.g. [84]). Yet, as shown by Liang et al. [57], even such ML-PWD can be "cracked" by oblivious attackers—if they invest enough effort to reverse engineer the entire ML-PWD. Indeed, we address ML-PWD because prior work (e.g., [20, 36, 55, 79]) assumed threat models that hardly resemble a real scenario. Phishing, by nature, is meant to be cheap [50] and most attempts end up in failure [66]. It is unlikely[1] that a phisher invests many resources *just to evade* ML-PWD: even if a website is not detected, the user may be 'hooked', but is not 'phished' yet. As a result, the state-of-the-art on adversarial ML for PWD is immature—from a pragmatic perspective.

Contribution and Organization. Let us explain how we aim to spearhead the security enhancements to ML-PWD. We begin by introducing the fundamentals concepts (PWD, ML, and adversarial ML) at the base of our paper in §2, which also serves as a motivation. Then, we make the following four contributions.

- We *formalize the evasion-space* of adversarial attacks against ML-PWD (§3), rooted in exhaustive analyses of a generic ML-PWD. Such evasion-space explains 'where' a perturbation can be introduced to fool a ML-PWD. Our formalization highlights that even adversarial samples created by direct feature manipulation can be realistic, *validating all the attacks performed by past work.*
- By using our formalization as a stepping stone, we propose a *realistic threat model* for evasion attacks against ML-PWD (§4). Our threat model is grounded on detailed security considerations from the viewpoint of a typical phisher, who is confined in the 'website-space'. Nevertheless, our model can be relaxed by assuming attackers with greater capabilities (which require a higher cost).
- We practically demonstrate the previous contributions (§5). We perform an extensive, reproducible, and statistically validated *evaluation of adversarial attacks* against state-of-the-art ML-PWD. By using diverse datasets, algorithms and features, we develop 18 ML-PWD and assess them against 12 different evasion attacks built upon our threat model.
- By analyzing the results of our evaluation (§6): (i) we *show the impact of attacks that are very likely to occur* against both baseline and adversarially robust ML-PWD; and (ii) we are the first to *fairly compare the effectiveness* of evasion attacks in the problem-space with those in the feature-space.

Our results highlight that more realistic attacks are not as disruptive as claimed by past works (§7), but their low-cost makes them a threat that induces statistically significant degradations. Finally, our evaluation serves as a 'benchmark' for future studies: we provide the complete results and source-code in a dedicated website: https://spacephish.github.io.

2 BACKGROUND AND MOTIVATION

Our paper lies at the intersection of Phishing Website Detection (PWD) and Machine Learning (ML) security. To set-up the stage for our contribution and motivate its necessity, we first summarize PWD (§2.1), and then explain the role of ML in PWD (§2.2). Finally, we provide an overview of the adversarial ML domain (§2.3).

2.1 Phishing Website Detection

Although having been studied for nearly two decades [51], phishing attacks are still a rampant menace [49]: according to the FBI [1], the number of reported phishing attempts has increased by 900% from 2018 to 2020 (26k up to 240k). Aside from the well-known risks to single users (e.g., fraud, credential theft [37]), phishing is still one of the most common vectors to penetrate an organization's perimeter. Intuitively, the best countermeasure to phishing is its prevention through proper *education* [94]. Despite recent positive trends, however, such education is far from comprehensive: the latest "State of the Phish" report [7] states that more than 33% of companies do not have any training program for their employees, and more than 50% only evaluate such education through simulations. As a result, there is still a need of IT solutions that mitigate the phishing threat by its early *detection*. In our case, this entails identifying a phishing website before a user lands on its webpage, therefore defusing the risk of falling victim to a phishing attack. We provide in Fig. 1 an exemplary architecture of a Phishing Website Detector (PWD).

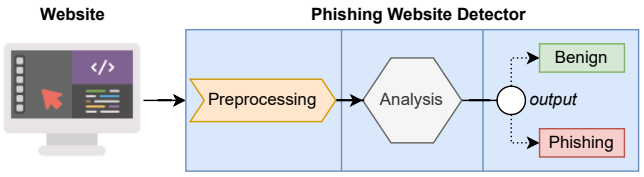


Fig. 1: Exemplary PWD. After preliminary preprocessing, a website is analyzed by a detector to determine its legitimacy.

Despite extensive efforts, PWD remains an open issue. This is due to the intrinsic limitations of the most common detection approaches reliant on *blocklisting* (e.g., [65, 74]). Such techniques have been improved and nowadays they even involve automatic updates with recent feeds (e.g., PhishTank [6]). However, blocklists are a double-edged sword: on the good side, they are very precise and are hence favored due to the low rate of false alarms; on the bad side, they are only effective against known phishing websites [9]. The latter is a problem: expert attackers are aware of blocklists and hence move their phishing 'hooks' from site to site, bypassing most

[1] It is unlikely, but *not impossible*. Hence, as recommended by Arp et al [17], it is positive that such cases have also been studied by prior work.

PWD. As shown by Tian et al. [85], such strategies can elude over 90% of popular blocklists for more than one month. To counter such *adaptive* attackers, much attention has been given to data-driven detection schemes—including those within the Machine Learning (ML) paradigm [84]. Indeed, ML allows to greatly enhance the detection capabilities of PWD. Let us explain why.

2.2 Machine Learning for PWD

The cornerstone of ML is having "machines that automatically learn from experience" [48], and such experience comes in the form of *data*. By applying a given ML *algorithm* $\mathcal{A}$, e.g. Random Forest (RF), to analyze a given *dataset* $\mathcal{D}$, it is possible to *train* a ML *model* $\mathcal{M}$ that is able to 'predict' previously unseen data. We provide a schematic of such workflow in Fig. 2. In the case of PWD, a ML model $\mathcal{M}$ can be deployed in a detector (e.g., in the hexagon in Fig. 1) to *infer* whether a given webpage is benign or phishing.

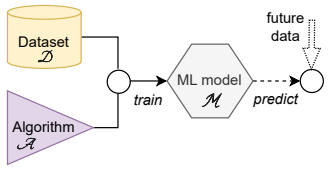


Fig. 2: Machine Learning workflow. By training $\mathcal{A}$ on $\mathcal{D}$, a ML model $\mathcal{M}$ is developed. Such $\mathcal{M}$ can be used to predict future data.

The main advantage of ML models is their intrinsic ability of noticing weak patterns in the data that are overlooked by a human, and then leveraging such patterns to devise 'flexible' detectors that can counter even adaptive attackers. As a matter of fact, Tian et al. [85] show that a ML model based on RF is effective even against "squatting" phishing websites—while retaining a low-rate of false alarms (only 3%). Moreover, acquiring suitable data (i.e., recent and labelled) for ML-PWD is not difficult—compared to other cyber-detection problems for which ML has been proposed [16].

Such advantages have been successfully leveraged by many research efforts (e.g., [64, 83]). Existing ML-empowered PWD can leverage different types of *information* (i.e., *features*) to perform their detection. Such information can pertain either to a website's *URL* [91] or to its *representation*, e.g., by analyzing the actual image of a webpage as rendered by the browser [41], or by inspecting the HTML [46]. For example, Mohammad et al. [60] observed that phishing websites usually have long URLs; and often contain many 'external' links (pointing to, e.g., the legitimate 'branded' website, or the server for storing the phished data), which can be inferred from the underlying HTML. Although some works use only URL-related features (e.g., [24]) – which can also be integrated in phishing *email* filters (e.g., [38]) – more recent proposals use combinations of features (e.g., [30, 89]); potentially, such features can be derived by querying third-party services (e.g., DNS servers [45]).

The cost-effectiveness of ML-PWD increased their adoption: even commercial browsers (e.g., Google Chrome [57]) integrate ML models in their phishing filters (which can be further enhanced via customized add-ons [84]); moreover, ML-PWD can also be deployed in corporate SIEM [43]. However, it is well-known that no security solution is foolproof: in our case, ML models can be thwarted by exploiting the so-called adversarial attacks [14].

2.3 Adversarial Attacks against ML

The increasing diffusion of ML led to question its security in adversarial environments, giving birth to "adversarial machine learning" research [21, 29]. Attacks against ML exploit *adversarial samples*, which leverage perturbations to the input data of a ML model that induce predictions favorable to the attacker. Even imperceptible perturbations can mislead proficient ML models: for instance, Su et al. [82] modify a single pixel of an image to fool an object detector; whereas Apruzzese et al. [13] evade botnet detectors by extending the network communications with few junk bytes.

An adversarial attack is typically described with a *threat model*, which explains the relationship of a given *attacker* with the *defender's system*. In particular, the attacker has a *goal* and, by leveraging their *knowledge* and *capabilities*, they will adopt a specific *strategy* [21]. Common terms associated with the attacker's knowledge are *white-box* and *black-box*: in the former, the attacker knows everything about the defender; whereas in the latter the attacker knows nothing [70, 97]. The capabilities describe how the attacker can interact with the target system, e.g., they: can influence only the *inference* or also the *training* stage of the ML model; can use the ML model as an "oracle" by inspecting the output to a given input; and can be subject to constraints on the creation of the adversarial perturbation (e.g., a limited amount of queries).

Despite thousands of papers focusing on this topic, a universal and pragmatic solution has not been found yet. Promising defenses are invalidated within the timespan of a few months (e.g. distillation was proposed in [72] and broken in [28]). Even "certified" defenses [47] can only work by assuming that the perturbation is bounded within some magnitude—which is not a constraint to which real attackers must abide (as pointed out by Carlini et al. [27]). From a pragmatic perspective, *any defense has a cost*: first, because it must be developed; second, because it can induce additional overhead. The latter is particularly relevant in cybersecurity, because it may decrease the performance of the ML model when no adversarial attack occurs. For instance, a well-known defense is *feature removal* [80], which entails developing ML models that do not analyze the features expected to be targeted by a perturbation. Doing this, however, leads to less information provided to the ML model, hence inducing performance degradation (e.g., [14]). Even when countermeasures have a small impact (e.g., [32]), this is not negligible in cyber-detection: attacks are a "needle in a haystack" [85], and even a 1% increase in false positives is detrimental [90]. Therefore, ML engineers will not devise any protection mechanism unless the corresponding threat is shown to be dangerous in reality [53].

The Problem. Unfortunately, research papers intrinsically impair the development of secure ML systems, because the aim is often to "outperform the state-of-the-art". In adversarial ML, this leads to papers that either showcase devastating attacks stemming from extremely powerful adversaries (i.e., white-box [82]); or viceversa, i.e., show that even oblivious attackers can thwart ML systems [70]. However, real 'adaptive' attackers (i.e., those that ML methods should be protected against) do not conform to these two extremes. Indeed, having complete knowledge of the target system requires a huge resource investment (especially if such system is devoted to cybersecurity), which may be better spent elsewhere; conversely, it is unlikely that opponents will launch attacks while knowing nothing of the defender. Hence, to provide *valuable* research, efforts on

adversarial ML should start focusing on the gray area within these two extremes—which implicitly are more likely to occur [12]. In the context of ML-PWD, our paper is a first step in this direction: as we will show, evasion attempts evaluated in literature (§7), despite being devastating, are costly to launch—even in black-box settings.

3 THE EVASION-SPACE OF ADVERSARIAL ATTACKS AGAINST ML-PWD

We aim to spearhead valuable research in adversarial attacks against ML-PWD. To this purpose, we first elucidate the internal functionalities of a ML-PWD (§3.1). Then, we propose our original formalization of the *evasion-space* of adversarial perturbations (§3.2). Finally, we explain why our contribution validates *all* prior work (§3.3).

3.1 Analysis of a ML-PWD

Let us connect the previously introduced concepts (cf. §2.1 and §2.2) and provide an overview of a generic ML-PWD in Fig. 3.

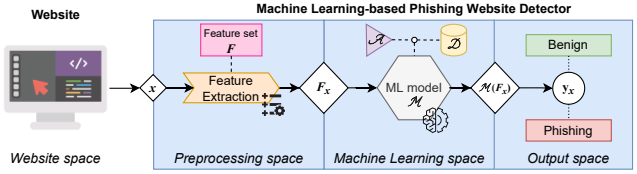


Fig. 3: Architecture of a ML-PWD. A website, x, is preprocessed into F_x. A ML model $\mathcal{M}$ analyzes such feature representation and predicts its ground truth as $\mathcal{M}(F_x) = y_x$.

A sample (i.e., a website), x, 'enters' the ML-PWD and is subject to some preprocessing aimed at transforming any input into a format accepted by the ML model—according to a given feature set, F. (We assume that x is not blocklisted.) The result of such preprocessing is the feature representation of the website x, i.e. F_x, which can now be analyzed by the ML model $\mathcal{M}$. We consider a ML model focused on *binary classification*. Hence, training $\mathcal{M}$ requires: a dataset, $\mathcal{D}$, whose samples are labelled as *benign* or *phishing*; and any ML algorithm, $\mathcal{A}$, supporting classification tasks (e.g., RF).

The ML model $\mathcal{M}$ predicts the ground truth of F_x as y_x, i.e., $\mathcal{M}(F_x) = y_x$. Hence, we can summarize the workflow of our ML-PWD through the following Expression:

$$x \rightarrow F_x \rightarrow \mathcal{M}(F_x) = y_x. \tag{1}$$

If x is a phishing (benign) webpage and y_x is also phishing (benign), then we have a true positive (true negative); otherwise, we have an incorrect classification (either a false positive or a false negative). We assume that $\mathcal{M}$ has been properly trained, so that its deployment performance yields a high true positive rate (tpr) while maintaining a low false positive rate (fpr)—under the assumption that no adversarial attack occurs.

3.2 Evasion Attacks against ML-PWD

Adversarial attacks exploit a perturbation, ε, that induces a ML model $\mathcal{M}$ to provide an output favoring the attacker (cf. §2.3). In our case, $\mathcal{M}$ is a (binary) classifier that analyzes F_x, hence we can express an adversarial attack through the following Expression:

$$\text{find } \varepsilon \text{ s.t. } \mathcal{M}(F_x) = y_x^{\varepsilon} \neq y_x. \tag{2}$$

In other words, the objective is finding a perturbation ε that induces a ML model $\mathcal{M}$ (that is assumed to work well) to misclassify a given sample x (i.e., $y_x^{\varepsilon} \neq y_x$). Because our focus is on *evasion* attacks, such misclassification entails having a positive (i.e., phishing) classified as a negative (i.e., benign). It is implicitly assumed that such ε must: (i) preserve the *ground truth*[2] (i.e., y_x^{ε} should be the same as y_x); and (ii) preserve the *phishing logic* of a webpage [69]. Such ε, however, can lead to different effects on y_x^{ε} depending on 'where' it is applied during the workflow described by Exp. 1. We describe such occurrence by formalizing the *evasion-space* of an attacker.

Evasion-Space. Let us observe Fig. 3. We can see that the figure is divided into four 'spaces', each allowing the introduction of a perturbation ε that can affect the output of the ML-PWD. Of course, a perturbation in the last space, i.e., the *output-space*, cannot be considered as an 'adversarial ML attack', because it will have no relationship with the ML model $\mathcal{M}$. Hence, the evasion-space of an attacker that wants to induce a misclassification by $\mathcal{M}$ is confined to the first three spaces. Let us analyze each of these.

(1) *Website-space Perturbations (WsP).* The entire detection workflow begins in the 'website-space', in which the website (i.e., x) is generated. Such space is accessible by any attacker, because they are in control of the generation process of their (phishing) website. As an example, the attacker can freely modify the URL or the representation of a website (subject to physical constraints[3]). Introducing a perturbation ε in this space (i.e., a WsP) yields an adversarial sample $\bar{x} = x + \varepsilon$, and the effects of such ε *can* affect all the operations performed by the ML-PWD (cf. Exp 1). We emphasize the word "can": this is because what happens *after* $\bar{x}$ enters the ML-PWD strictly depends on the implementation of such ML-PWD—which may, or may not, 'notice' the corresponding ε (e.g., $\mathcal{M}$ can analyze an F that is not influenced by ε).

(2) *Preprocessing-space Perturbations (PsP).* After x is acquired by the ML-PWD, it is first transformed into F_x. *An attacker with write access* to the 'preprocessing-space' can introduce a PsP ε that affects the *process* that yields the feature representation of a website, leading to $\overline{F_x} = F_x + \varepsilon$. For instance, a website x with an URL of 40 characters can be turned into a $\overline{F_x}$ that has the *URL_length* feature=20. Intuitively, attackers able to introduce PsP are powerful, but are still subject to constraints: before any F_x is sent to the ML model $\mathcal{M}$, such F_x is checked to ensure that it is not corrupted [12]. Indeed, $\overline{F_x}$ must not violate any inter-feature dependencies or physical constraints. With respect to WsP, PsP are guaranteed to be 'noticed' by the ML-PWD; however, they do not necessarily influence the predictions of $\mathcal{M}$: making a URL shorter may not be enough to fool the detection process.

(3) *ML-space Perturbations (MsP).* After the preprocessing, the feature representation of a website F_x enters the Machine Learning-space in order to be analyzed by $\mathcal{M}$. *If an attacker has write access* to this space, they can introduce an MsP, i.e., a perturbation ε that affects F_x *immediately before* it reaches $\mathcal{M}$. An MsP is the 'strongest' type of perturbation because it affects the F_x after all integrity checks[4] have been performed—potentially leading to corrupted values, or which have no relationship to any real x. We hence denote MsP

[2] E.g., changing a URL from "go0gle.com" to "google.com" is not a valid ε.
[3] Which depend on the semantics of websites, e.g., URLs cannot be 1 character long.
[4] Indeed, a ML model $\mathcal{M}$ is agnostic to the generation process of a given input.

as $\bar{F}_x = F_x + \varepsilon$. As an example, a MsP can yield a $\bar{F}_x$ having an *URL_length*=0. As such, MsP are very likely to induce uncanny responses by $\mathcal{M}$ (but do not guarantee evasion).

Summary and Cost. From Exp. 2, we observe that any perturbation ε should ultimately affect the feature representation F_x of a given sample x. Hence, the crux is determining 'where' such perturbation is introduced—which can happen in three spaces. We formally define adversarial attacks by means of introducing a perturbation in each of these spaces (i.e., WsP, PsP and MsP) through the following Expression (which extends Exp. 1):

$$\text{find } \varepsilon \text{ s.t.} \begin{cases} \bar{x} = x + \varepsilon \;\Rightarrow\; x \to \bar{x} \to F_{\bar{x}} \to \mathcal{M}(F_{\bar{x}}) = y_x^c \neq y_x & \text{WsP} \\ \overline{F_x} = F_x + \varepsilon \;\Rightarrow\; x \to \overline{F_x} \to \mathcal{M}(\overline{F_x}) = y_x^c \neq y_x & \text{PsP} \\ \bar{F}_x = F_x + \varepsilon \;\Rightarrow\; x \to F_x \to \bar{F}_x \to \mathcal{M}(\bar{F}_x) = y_x^c \neq y_x & \text{MsP} \end{cases} \tag{3}$$

We remark that the effects of WsP can match those of PsP—which can also match those of MsP. For instance, a MsP can yield a sample with an *URL_length* of 20 which – as long as it does not violate any inter-feature dependency – can represent a valid website (hence MsP=PsP)[5]; to obtain an equivalent WsP, the attacker would have to modify the actual URL and make it of exactly 20 characters (which is doable). Hence, in some cases, $F_{\bar{x}}$=$\overline{F_x}$=$\bar{F}_x$. As such, although some MsP cannot be crafted in the website-space, it is also unfair to consider all MsP (or PsP) as being not physically realizable. Finally, from a *cost* viewpoint, WsP≪PsP<MsP, because realizing MsP requires the attacker to have more control[6] on the ML-PWD (i.e., they must obtain write-access to deeper segments of the ML-PWD).

3.3 Validation of Previous Work

An important contribution of our evasion-space is that it *validates all past research* that consider perturbations in the "feature-space" (i.e., PsP or MsP). Let us explain why.

Context. By using Pierazzi et al. [73] notation, our WsP can be seen as perturbations in the "problem-space"; whereas PsP and MsP are perturbations in the "feature-space". The main thesis of Pierazzi et al. [73] is that evaluations carried out in the feature space are unreliable due to the "inverse mapping problem": some changes in the feature representation of a sample (i.e., F_x) may not be physically realizable when manipulating the original sample (i.e., x)—therefore exposing the "weakness of previous evasion approaches."

Intuition. Our original formalization elucidates that the "weaknesses" of past work are not, in fact, weaknesses—therefore overturning some of the claims of Pierazzi et al. [73]. Our thesis is rooted in the following observation: the "inverse mapping problem" is irrelevant *if the attacker has write access to the ML-PWD.*

Explanation. Any attacker is able to craft WsP by manipulating their own phishing webpages (to some degree). In contrast, *reliably* realizing PsP and MsP can only be done by assuming an attacker that can manipulate the corresponding space (i.e., either the preprocessing- or the ML-space). Achieving this in practice presents a high barrier of entry—but *it is not impossible*. For instance, consider the case of an attacker who has compromised a given device integrating a client-side ML-PWD: such attacker can interfere with any of the ML-PWD operations—especially if it is open-source (e.g., [44]). Of course, realizing PsP or MsP if the ML-PWD is deployed in an organization-wide intrusion detection system is harder, but not unfeasible (as pointed out by [12]).

Takeaway: Our formalization validates all evasion attacks against ML-PWD previously evaluated through perturbations in any internal 'space' of the ML-PWD. This requires to *change the attacker's assumptions*, implicitly increasing the cost of the attack.

Consequences. Simply put, we *restore* the value (partially 'lost' after the publication of [73]) of the evaluations performed by prior work (§7). By assuming that the considered attacker can access a given space of the ML-PWD (either for PsP or MsP), then there is no risk of falling into the "inverse mapping problem"—because it is a constraint that such attacker is not subject to. Such different assumptions, however, implicitly raise the cost of the corresponding attack. For example. Corona et al. [30] craft perturbations in the ML-space: according to [73], the resulting perturbations are, hence, unreliable. However, by assuming that the attacker *can manipulate the ML-space*, then such adversarial examples (deemed unreliable by [73]) would become realistic (thanks to our contribution).

4 PROPOSED REALISTIC THREAT MODEL

We use our evasion-space formalization to devise our proposed adversarial ML threat model—describing attractive strategies for real phishers. We first provide its definition (§4.1), and then support its realisticness via security analyses (§4.2). In Appendix B we show how to apply WsP on *real* phishing webpages.

4.1 Formal Definition

We define our threat model according to the following four criteria (well-known in adversarial ML [21]).

Goal. The adversary wants to *evade* a ML-PWD that uses $\mathcal{M}$ as a detection method (i.e., the attacker wants to satisfy Exp. 2).

Knowledge. The adversary has *limited knowledge* of the target system, the ML-PWD. They know nothing about: the ML model $\mathcal{M}$, its training data $\mathcal{D}$, and its underlying ML algorithm $\mathcal{A}$ (except that it supports binary classification). However, the adversary knows a subset of the feature set F analyzed by $\mathcal{M}$. Let $K \subseteq F$ be such a subset. The adversary is also aware that the ML-PWD will likely detect phishing websites if no evasion attempt is made (otherwise, there would be no reason to do so). Finally, the adversary implicitly knows that no blocklist includes their phishing webpages (otherwise, the attacker would be *forced* to manipulate the URL).

Capability. The adversary has *no access* to the ML-PWD. They cannot use the ML-PWD as an "oracle" (i.e., inspect the output to a given input); and they are therefore confined to perturbations in the website-space (i.e., WsP).

Strategy. The adversary uses their knowledge of K to craft WsP that may result in successful evasion attacks *at inference time.*

We observe that our threat model is *general* because no specific set of features (F) or ML model $\mathcal{M}$ (and hence $\mathcal{D}$ and $\mathcal{A}$) is provided. Therefore, our threat model can cover any ML-PWD that resembles the one in Fig. 3. Potentially, it can even be a ML-PWD used by email filters if the corresponding $\mathcal{M}$ analyzes URL-related information

[5] Of course MsP=PsP if there is no 'integrity check'.

[6] Our formalization is orthogonal to the one by Šrndic and Laskov. [98]: while [98] focus on the attacker's *knowledge* ("what does the attacker know about the ML system?"), we focus on the *capabilities* (i.e., "where can the attacker introduce a perturbation affecting the ML system?"). Moreover, our PsP are semantically different than the "adversarial preprocessing" by Quiring et al. [75]: while [75] affect the preprocessing phase *from outside* the ML system, our PsP affect such phase *from the inside.*

(e.g., [34, 38]). Furthermore, our threat model can be *extended*. We will do so in our evaluation (§5), in which we compare the effects of attacks using WsP against those entailing PsP and MsP (by assuming the same knowledge, i.e., limited to K).

4.2 Security Analysis

Let us analyze our threat model and explain why it portrays a *realistic* attacker—especially if compared to typical 'white-/black-box' adversarial scenarios (cf. §2.3). We intend to justify that our threat model describes attacks that are *interesting* to investigate, and hence *valuable* for the security of ML-PWD.

Phishing in a nutshell. We start by focusing the attention on the intrinsic nature of phishing. Indeed, phishing attempts – and especially those involving phishing websites – are 'cheap' in nature [50]. Considering that real attackers operate with a cost-benefit mindset, it is unlikely that such attackers will invest extensive resources just to have their webpages evade a ML-PWD. Firstly, because such evasion will be temporary (as soon as the webpage is reported in a blocklist, any adversarial attack will be useless); secondly, because, even if a website evades a ML-PWD, the phishing attempt is not guaranteed to succeed (a user still has to input its sensitive data). Indeed, despite the exponential proliferation of phishing [7], most phishing attempts are prone to failure [66]—and the attackers are well aware of this fact. Of course, attackers can opt for more expensive spear-phishing campaigns [25] (which still have a success rate of barely 10% [42]), but in this case they will likely design entirely new phishing webpages—and not rely on cheap perturbations on pre-existing samples.

Limited Knowledge. Our attacker knows *something* (i.e., K) about the ML-PWD, but they are not omniscient—hence, our threat model can be considered as a gray-box scenario. Such 'box', however, is the entire ML-PWD, i.e., the blue rectangle in Fig. 3. Our scenario is *more interesting* to investigate than white-box scenarios. The reason is simple: ours is *more likely* to occur, because 'phishers' with complete knowledge of the entire ML-PWD are extremely unlikely. Furthermore, extensive adversarial ML literature [21] has ably demonstrated that white-box attacks can break most systems—including ML-PWD (e.g., [8, 36, 59, 81]).

Realistic Capabilities. Our 'standard' attacker has no access to the ML-PWD, which is a realistic assumption. For instance, the attacker can share a phishing website via social media, but without knowing which device (and, hence, ML-PWD) is being used by potential victims to open such website. Therefore, the attacker cannot reliably use $\mathcal{M}$ as an oracle. They could opt for querying a surrogate ML-PWD to reverse-engineer its functionalities and then leverage the transferability of adversarial attacks [33]. However, such 'black-box' scenario is both (i) unlikely to occur; and (ii) ultimately not interesting to consider for a research paper. *Unlikely*, because it would *defeat the purpose* of phishing attacks: reverse-engineering operations require a huge resource investment—which can be invalidated via a simple re-training of $\mathcal{M}$ (a common cybersecurity practice [15]). *Not interesting*, because such attacks *have been investigated before* [10, 77]. For instance, Liang et al. [57] clearly demonstrated that attackers with access to client-side detectors can successfully crack and evade the corresponding ML-PWD; doing this, however, required *more than 24 hours* of constant queries [57].

Takeaway: Phishing attempts have an intrinsic low rate of success. Attackers that aim to evade a ML-PWD will favor 'cheap' tactics—which can be represented by our proposed threat model.

Consideration. Attacking ML-PWD through (potentially unreliable) WsP is not the only way to 'realistically' evade ML-PWD. This is clearly evidenced by prior work—whose validity is restored thanks to our evasion-space formalization. However, our proposed 'cheap' attacks (through WsP) have never been investigated before in adversarial ML literature on PWD (§7). We hence set out to proactively assess the impact of feasible WsP on state-of-the-art ML-PWD; and comparing such impact to 'less realistic' (hence, less likely to occur) attacks performed through PsP and MsP. Therefore, our evaluation will also consider such worst-case scenarios. We stress, however, that our threat model shall not envision attackers who: (i) can observe or manipulate $\mathcal{D}$ (for poisoning attacks); (ii) can observe the output-space (for black-box attacks); (iii) have full knowledge of the ML-PWD (for white-box attacks).

5 EVALUATION

As a constructive step forward, we assess the robustness of 18 ML-PWD against 12 evasion attacks—all based on our threat model, but performed in different evasion spaces. We have three goals:

- assess state-of-the-art ML-PWD against *feasible attacks*;
- compare perturbations introduced in *distinct evasion-spaces*;
- provide a *statistically validated benchmark* for future studies.

Achieving all such goals is challenging in research. Indeed, *crafting* perturbations in the three distinct spaces (i.e., WsP, PsP, MsP) requires: (i) datasets containing raw-data (for WsP), which are difficult to find; (ii) devising custom feature extractors (for developing the ML-PWD); as well as (iii) foreseeing the effects of WsP on such extractor (for PsP). Furthermore, to derive statistically sound conclusions, we must repeat our experiments multiple times [16].

We describe our experimental setup (§5.1), and then summarize our evaluation workflow (§5.2). More details are in Appendix D.

5.1 Experimental Setup

We consider a total of 18 ML-PWD, which vary depending on the *source dataset* (2), the *ML algorithm* (3), and the *feature set* (3) used to develop the corresponding ML model. Such a wide array allows one to draw more generalizable conclusions.

5.1.1 ***Source Datasets.*** We rely on two datasets for ML-PWD: δ`phish` and `Zenodo` [30, 89]. Our choice is based on three reasons.

- Both datasets include *raw information* of each sample (specifically, its URL and its HTML). This is necessary because most of our attacks leverage WsP, for which we must modify the raw webpage, i.e., before its features are extracted.
- Both datasets have been *used by the state-of-the-art*. Prior research [30, 89] has demonstrated the utility of both datasets for ML-PWD, allowing for fair and significant comparisons.
- They enable experimental *reproducibility*. Indeed, collecting ad-hoc data through public feeds (e.g., AlexaTop/PhishTank) prevents fair future comparisons: phishing webpages are taken down quickly, and it is not possible to retrieve the full information of webpages 'blocklisted' years before.

We provide an overview of our datasets in Table 2, which shows the number of samples (benign and phish) and the performance (tpr and fpr) achieved by their creators (in the absence of evasion).

5.1.2 ***ML Algorithms.*** We consider ML-PWD based on shallow and deep learning algorithms [14] for binary classification. Our selection aims to provide a meaningful assessment of exemplary ML-PWD based on exemplary ML methods. In particular, we consider:

- Logistic Regression (*LR*). One of the simplest ML algorithms, we consider *LR* because it was (assumed to be) used by the ML-PWD embedded in Google Chrome [57].
- Random Forests (*RF*). An ensemble technique, *RF* often outperforms other contenders in phishing detection tasks [85].
- Convolutional neural Network (*CN*). We consider this well-known deep learning technique [54] due to its demonstrated proficiency also in ML-PWD (e.g., [92]).

5.1.3 ***Feature Sets.*** We consider ML-PWD that use three feature sets (F), all resembling the one described in our use-case (Appendix B). Specifically, our ML-PWD analyze one of the following:

- URL-only (F^u), i.e., the first 35 features in Table 1.
- Representation-only (F^r), i.e., the last 22 features in Table 1.
- Combined (F^c), corresponding to all features in Table 1.

Rationale. Analyzing more information (i.e., larger feature sets, such as F^c) leads to superior detection performance—as shown, e.g., in [30]. However, in some cases this may not be possible: for instance, phishing *email* filters may make their decisions only by analyzing the URL (cf. §2.2). Nevertheless, modifying the URL is one of the easiest ways to trick a ML-PWD [67]: hence, a defender may develop an 'adversarially robust' detector that analyzes only the representation of a webpage. Such detector will have a lower performance (w.r.t. F^c) in non-adversarial scenarios, but will counter evasion attacks that manipulate the URL (cf. §2.3).

Observation. Our feature sets are not only popular in research (e.g., [40, 45, 60, 78]), but also used in *practice*. Indeed, several leading security companies yearly organize MLSEC, an ML evasion competition [5]. In 2021 and 2022, MLSEC also involved evading ML-PWD *which specifically analyzed the HTML* representation of a webpage—i.e., our F^r. We will also refer to MLSEC in our evaluation.

5.1.4 ***Considered Attacks.*** In our evaluation, we assess the robustness of each of the 18 ML-PWD against a total of 12 evasion attacks, which vary depending on the attacker's knowledge (i.e., K), capabilities (i.e., the evasion-space) and strategy (i.e., the features 'targeted'). In particular, we consider two macro-families of attacks:

- **Cheap (Website) Attacks** (WA), corresponding exactly to our threat model and described in our case-study (in Appendix B). The adversary has no access to the ML-PWD, and can only apply WsP (which may not be effective).
- **Advanced Attacks**, where we relax some of the assumptions of our threat model to describe a more powerful attacker[7]. We consider three families: $\widehat{\text{WA}}$, wherein the attacker uses WsP, but knows a portion of the low-level implementation of the feature extractor; PA, wherein the attacker has write-access to (parts of) the *preprocessing-space*, and applies PsP; and MA, wherein the attacker has write-access to the *ML-space* and will apply MsP (a worst-case scenario).

Each of these four attack families (i.e., WA, $\widehat{\text{WA}}$, PA, MA) comes in three variants—depending on the features known (and targeted) by the attacker (i.e., u, r, c). For instance, WA^r is a WA in which the attacker tries to affect (through WsP) features related to the HTML representation of the webpage. Despite all our perturbations being ultimately 'blind' (the attacker will never be able to observe their effect), we can expect that MA will have a greater impact than WA on the ML-PWD. However such impact is compensated by the higher entry barrier for MA (see §3.2). More details, including a high-level estimate of the affordability of our attacks, are in Appendix D.

[7]These attacks are solely for research: their implicit higher cost w.r.t. WA may discourage real phishers from launching them (although they are not completely impossible).

5.2 Workflow and Statistical Validation

Each source dataset (`Zenodo` and δ`phish`) represents a different setting—which we use to extract the corresponding training and inference partitions for our ML-PWD. Such ML-PWD are based on one among three ML algorithms, encompassing either shallow (*LR* and *RF*) or deep learning (*CN*) classifiers. Each of these classifiers presents three variants, depending on the analyzed features (F^u, F^r, or F^c), yielding a total of 9 'baseline' ML-PWD per source dataset. After ensuring that such 9 ML-PWD maximize their performance (high tpr and low fpr, at least for F^c), we assess their robustness against *all* the 12 proposed evasion attacks. Such attacks come in four families (WA, $\widehat{\text{WA}}$, PA, MA) depending on the knowledge and capabilities of the opponent, and each family presents three variants denoting the specific strategy, i.e., which features are 'targeted' by the attacker (either u, r, or c). We consider ML-PWD using F^c to be the 'true' baselines (likely highest performance in the absence of evasion attempts); whereas those using either F^u or F^r can be considered as 'robust' baselines (i.e., using F^u protects against attacks targeting F^r, and viceversa).[8] Such workflow is shown in Fig. 6.

To provide results that are devoid of experimental bias and also to serve as a reliable benchmark for future researches, *we repeat all the abovementioned operations 50 times.* This means that each source dataset is randomly sampled 50 times, each resulting in a different training partition $\mathcal{D}$ and, hence, a different $\mathcal{M}$. Such $\mathcal{M}$ is, in turn, assessed on different data (i.e., different inference partitions), yielding different tpr and fpr, and is also subject to the 12 evasion attacks (all using different malicious samples as basis).

Such a large[9] evaluation allows one to perform *statistically validated comparisons* by leveraging well-known techniques [16]. We will do this to infer whether some attacks induce a performance degradation that is statistically significant. To the best of our knowledge, we are the first to use statistical tests to validate the impact of adversarial attacks against ML-PWD.

6 RESULTS AND DISCUSSION

We present the results of our evaluation by focusing on our evasion attacks. Specifically, our results aim at answering two questions:

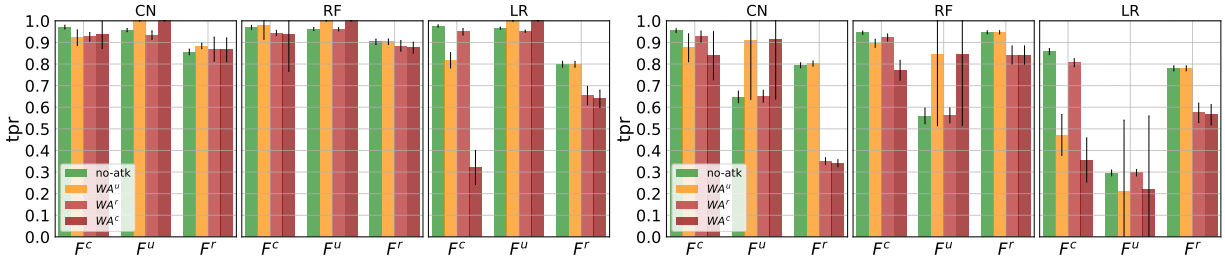
- (§6.1) how dangerous are the most likely attacks (i.e., WA)?
- (§6.2) what is the effectiveness of attacks carried out in different evasion spaces (i.e., $\widehat{\text{WA}}$, PA, MA)?

We discuss our evaluation and potential for future work in §6.3. Our Artifact includes the full 'benchmark' results.

Preliminary assessment. Our results *in the absence* of adversarial attacks, reported in Table 3, show that the best ML-PWD on both datasets use *RF*. We appreciate that the 'true' baseline ML-PWD

[8]Of course, the attacker expects the target ML-PWD to be using F^c.

[9]Overall, for our experiments we develop 900 $\mathcal{M}$ (given by: 2 source datasets * 50 random draws * 3 F * 3 $\mathcal{A}$), each assessed against 1200 adversarial examples.



(a) Impact of WA on the ML-PWD trained on Zenodo.

(b) Impact of WA on the ML-PWD trained on δPhish.

Fig. 4: Effectiveness of the most likely attacks (WA). The three plots in each subfigure represent the algorithm used by a specific ML-PWD. Each plot has bars divided in three groups, each denoting a specific F used by the ML-PWD. The green bars show the tpr on the original samples, while the others show the tpr against a specific variant of WA.

(using F^c) exhibit similar results as the state-of-the-art (cf. Table 2). In contrast, the 'robust' baselines (using either F^r or F^u) are slightly inferior[10]. For instance, on Zenodo, the *RF* using F^u has almost the same performance as F^c, but the one using F^r has 5% less tpr and 2% more fpr; whereas on δphish, the *RF* using F^u has 50% less tpr (but similar fpr), while the one using F^r has 0.5% more fpr, but only 3% less tpr. Such degradation is the **cost** of using defenses based on *feature removal* on the considered ML-PWD. The expected benefit, however, is a superior resilience to evasion attempts.

6.1 Effectiveness of the most likely attacks (WA)

Let us focus the attention on the most likely attacks. We report in Figs. 4 the tpr achieved by all our ML-PWD against all our WA attacks (red bars), and compare it with the tpr (*no-atk*, shown in green bars) achieved by the same ML-PWD on the original set of samples used as basis for WA. Some intriguing phenomena occur.

True Baseline (F^c). We first consider ML-PWD using F^c (leftmost group of bars in each plot), as they are the 'true' baseline.

- On δphish (Fig. 4b), all ML-PWD are affected by the 'strongest' cheap attack, i.e., WA^c. Specifically, the ML-PWD using *LR* is completely defeated (from 0.86 tpr down to 0.36); in contrast, those using *CN* or *RF* suffer a smaller, but still significant drop (from nearly 0.95 down to ~0.8). Notably, the *CN* despite being worse than the *RF* in non-adversarial settings (cf. Table 3), appears to be slightly more robust.
- The situation is different on Zenodo (Fig. 4a). Here, while the *LR* is still defeated, the *CN* and *RF* appear not to be very affected by WA^c. However, considering that both *CN* and *RF* exhibit very high performance in non-adversarial settings (cf. Table 3), it is crucial to determine whether WA^c poses a real threat to such ML-PWD. To this purpose, we carry out a Welch t-test, which we can do thanks to our large amount of trials. We set our null hypothesis as "WA^c and *no-atk* are equal". The findings are valuable: against *RF*, the *p-value* is 0.221; whereas against *CN*, the *p-value* is 0.002. By using the common statistical significance threshold of 0.05, we can hence provide the following answer: the *RF is not affected* by WA^c, whereas the *CN is affected* by WA^c.

The latter finding is intriguing, because it suggests that *shallow learning methods can be more resilient* than deep learning ones for PWD—against our proposed attacks. Finally, we also observe that WA^r clearly defeat *LR* on both datasets, whereas the impact on *RF* and *CN* is significant on δPhish, but small on Zenodo.

Robust Baselines (F^u, F^r). The robust baselines are, in general, reliable against WA. The ML-PWD using F^u counter WA^r (and viceversa), because the tpr is exactly the same as the original one. Notably, however, ML-PWD using F^r (similar to the ML-PWD of[11] MLSEC [5]) are affected by WA^r: the *LR* is clearly defeated on both datasets, whereas *RF* suffers a 10% and 3% drop on δphish and Zenodo, respectively. Nevertheless, we observe a fascinating phenomenon: in some cases, the tpr under attack *is higher* than in *no-atk*; e.g., on δphish the *RF* analyzing F^u has its tpr to increase from 0.56 to ~0.84 against both WA^u and WA^c. Such phenomenon occurs because the attacker (in any variant of WA) does not know 'what to do' to reliably evade the ML-PWD: the attacker guesses some WsP, which can have no impact, or even make the website closer to a 'malicious' one (from the viewpoint of $\mathcal{M}$).

Takeaway: The realistic attacks in the website-space (WA^c) can evade five (out of six) ML-PWD. Despite being small, the performance degradation is statistically significant: hence, due to their cheap cost, WA^c represent a threat to state-of-the-art ML-PWD.

6.2 Comparing the evasion-space ($\widehat{\mathsf{WA}}$, PA, MA)

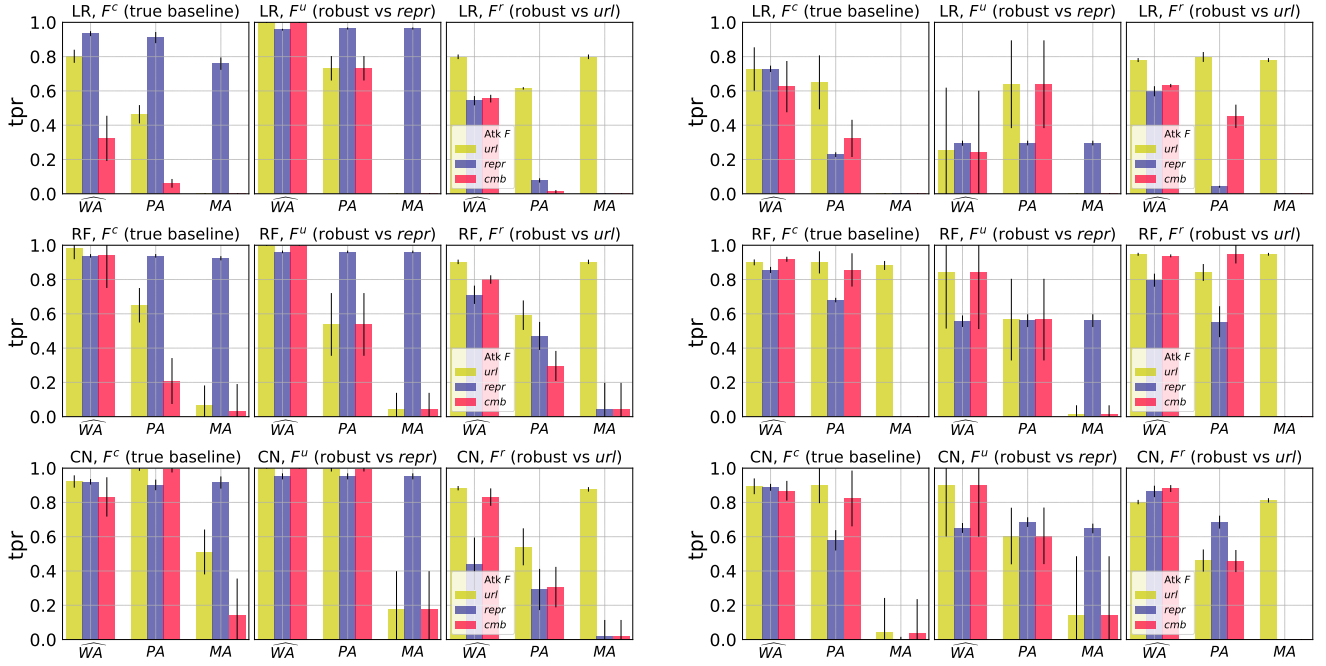
We now focus on comparing the effectiveness of attacks that aim at influencing the same features (i.e., either u, r, c), but whose perturbations are introduced in different spaces (i.e., either WsP, PsP, or MsP). We visualize such results in Fig. 5.

The 'true' baselines (using F^c, i.e., the leftmost plots in Fig. 5) are defeated by MA. However, there are some notable exceptions: on Zenodo, the *RF* and *CN* are resilient to MA^r (this is because the HTML features have little importance for F^c). In contrast, on δphish, *RF* can withstand MA^u. The 'robust' baselines counter the corresponding MA, but unsurprisingly suffer against the others.

In general, PA tend to have a larger impact than $\widehat{\mathsf{WA}}$ against the 'true' baselines. However, this is not always true: we find enlightening that the *CN* on Zenodo is more robust to PA than to $\widehat{\mathsf{WA}}$. What is even more surprising is that such *CN* significantly outperforms the *RF* against PA, but *also* against MA. Such finding could inspire

[10]Focusing on the ML-PWD using F^r (which are similar to the real ML-PWD in MLSEC [5]), we appreciate that *RF* achieves a remarkable 0.935 tpr and 0.01 fpr (averaged on both datasets), making such ML-PWD a valid baseline.

[11]We also successfully attacked the competition-grade ML-PWD of [5] with WA^r, achieving similar results than the one shown in our custom-built ML-PWD. A demonstrative video (of 140s) can be found at the homepage of our website.



(a) `Zenodo`. Each plot reports the tpr resulting from the 9 advanced attacks (i.e., $\widehat{\text{WA}}$, PA, MA) across the 50 trials. Colors denote the targeted features (u, r, c).

(b) δ`phish`. Each plot reports the tpr resulting from the 9 advanced attacks (i.e., $\widehat{\text{WA}}$, PA, MA) across the 50 trials. Colors denote the targeted features (u, r, c).

Fig. 5: Comparison of attacks carried out in different evasion-spaces. Each subfigure refers to a specific dataset, and presents 9 plots. Such plots are organized in three rows and three columns. Rows denote a specific ML algorithm (*LR*, *RF*, *CN*). Columns denote a specific feature set: the 'true' baseline (using F^c) is on the left; the others are the 'robust' baselines (using F^u or F^r).

deployment of ML-PWD using deep learning on `Zenodo`—despite being inferior to *RF* in the no-atk (Table 3) and against WA^c (§6.1).

We note that $\widehat{\text{WA}^u}$ perfectly match WA^u, which makes sense as they involve exactly the same WsP (cf. Appendix D). We can also see some discrepancies between $\widehat{\text{WA}}$ and PA: as a matter of fact, our anticipation of the preprocessing-space (i.e., the PsP of PA) did not exactly match what truly happened in the website-space . However, in some cases (e.g., the *RF* using F^c and F^r on δ`phish`) we observe that the effectiveness of $\widehat{\text{WA}}$ and PA tend to be similar. Such crucial finding demonstrates that perturbations applied directly to F_x (which we use for PA) can induce the same effects as those applied to x (which we use for $\widehat{\text{WA}}$). In other words: if properly crafted, then even perturbations in the "feature-space" can resemble adversarial examples that are physically realizable [86].

Let us compare our attacks with those considered by δ`phish` creators. Specifically, the attacks in [30] manipulate increasingly higher amounts of features (up to 10), and all ultimately evade target ML-PWD (which analyzes the HTML). Such finding is confirmed by our results on the ML-PWD analyzing F^r on δ`phish` against MA^r, which all misclassify the adversarial samples. However, *if the perturbations are applied in different spaces* (i.e., PsP or WsP), *then the ML-PWD is significantly less affected.*

6.3 Discussion and Future Work

Our evaluation is a proof-of-concept, and we do not claim that *all* ML-PWD will respond in the same way as ours, and neither we claim novelty in the 'generic' method used to to evade PWD (attackers have been manipulating the HTML or URL for decades [21]). Indeed, our goal was to validate our primary contribution (whose focus is on machine learning) by performing a fair comparison of attacks (each having a different *cost*) in diverse evasion-spaces.

Warning on WA. A legitimate observation is that our cheap attacks, despite affecting most ML-PWD, have a small impact—even if statistically significant (§6.1). Such results, however, must not induce conclusions such as "these attacks are not interesting" or (worse) "these attacks can be overlooked in the security lifecycle". Indeed, *the main threat of* WA *is represented by the cheap cost*: thousands of phishing websites are created every day [7], and in such big numbers even a 1% difference can be the separation between a compromised and secure system [15]. Our goal is not to propose devastating attacks that bypass any ML-PWD; rather, we focus on those attacks that are more likely to occur in reality. As a matter of fact, WAs *can be automatized and implemented within seconds and few lines of code*; in contrast, the advanced attacks (including those of past work, e.g., [30, 57]) require to compromise or reverse-engineer the ML-PWD (§3.1). The *cost* of an attack should also account for the *effort* required for its implementation. Most related literature focuses on measuring 'queries' (e.g., [33]): our WA do not require any query. Nonetheless, we invite future work to explore metrics to estimate the cost of attacks in terms of human effort.

Extensions. The main purpose of our evaluation is to highlight how state-of-the-art ML-PWD respond to diverse evasion attacks. There are, however, millions of ways to do the above. For instance, the attacks can target different features (and in different ways) than the ones considered in our evaluation (i.e., u, r, c); the ML-PWD can analyze different features, which can be generated via different

preprocessing mechanisms (e.g., [52]). Additional defenses can also be considered (e.g., adversarial training [68, 88]). For instance, we did not consider ML-PWD that analyze the visual representation of a webpage (e.g., [8, 59]): such attacks would resemble those conducted in computer vision, which are well-known to be effective (e.g., [71, 87]). Nevertheless, our threat model is agnostic of the data-type, so we endorse future work to also consider ML-PWD analyzing images. Finally, our evasion-space formalization can be applied even to settings beyond phishing (e.g., malware), which may entail attackers more likely to use PsP or MsP.

7 RELATED WORK

Countering phishing is a long-standing security problem, which can be considered as a subfield of cyberthreat detection—a research area that is being increasingly investigated also by adversarial ML literature [14]. We focus on the detection of phishing *websites*. Papers that consider phishing in social networks [22], darkweb [95], phone calls [39], or emails [34] are complementary to our work—although our findings can also apply to phishing email filters if they analyze the URLs included in the body text (e.g., [38]). Our focus is on attacks *against* ML-PWD. For instance, Tian et al. [85] evade PWD that use common blacklists, and their main proposal is to use ML as a detection engine to counter such "squatting" phishing websites. Hence, non-ML-PWD (e.g., [96]) are outside our scope.

Let us compare our paper with existing works on evasion attacks against ML-PWD. We provide an overview in Table 4, highlighting the main differences of our paper with the state-of-the-art. Only half of related papers craft their attacks in the problem-space—which requires modifying the raw webpage. Unfortunately, most publicly available datasets do not allow similar procedures. A viable alternative is composing ad-hoc dataset through public feeds as done, e.g., by [36] and [77] (the latter only for URL-based ML-PWD). All these papers, however, do not release the actual dataset, preventing reproducibility and hence introducing experimental bias. The authors of [81] share their dataset, but while the *malicious* websites are provided with complete information (i.e., URL and HTML), the *benign* websites are provided only with their URL—hence preventing complete reproducibility of attacks in the problem-space against ML-PWD inspecting the HTML. The latter is a well-known issue in related literature [69], which does not affect our paper because our entire evaluation is reproducible. Notably, Aleroud et al. [11] evaluate attacks both in the problem and feature-space, but on *different* datasets, preventing a fair comparison. Indeed, they evade one ML-PWD trained on `PhishStorm` (which only includes raw URLs) with attacks in the problem space; and another ML-PWD trained on `UCI` (which is provided as pre-computed features) through feature space attacks. Hence, it is not possible to compare these two settings. A similar issue affects also [10], which consider 4 datasets, each having a different F. Therefore, no prior work *compared the impact* of attacks carried out in distinct evasion-spaces—to the best of our knowledge. Not many papers consider adversarially robust ML-PWD, and only half consider both SL and DL algorithms—which our evaluation shows to respond differently against adversarial examples (cf. §6.2). It is concerning that few papers overlook the importance of statistically significant comparisons. The most remarkable effort is [79] which only performs 10 trials (we do 50), which are not enough to compute precise statistical tests.

Nevertheless, most prior work assume stronger attackers than those envisioned in our threat model (cf. §4). Indeed, past threat models portray *black-box* attackers who can freely inspect the output-space and query the ML-PWD (e.g., [10, 57, 77]); or *white-box* attackers who perfectly know the target ML model $\mathcal{M}$, such as its configuration, its training data $\mathcal{D}$, or the feature importance (e.g., [8, 36, 59]). The only papers considering attackers that are closer to our threat model are [55, 67] and [8]. However, the ML-PWD considered in [8] is specific for *images*, which are tough to implement (cf. §6.3) and also implicitly resembles a ML system for computer vision—a task well-investigated in adversarial ML literature [21]. In contrast, the ML-PWD considered in [55] and [67] is similar to ours, but the adversarial samples are randomly created in the feature space, hence requiring an attacker with write-access to the internal ML-PWD workflow. Such an assumption is not unrealistic, but very unlikely in the context of phishing (cf. §4.2).

8 CONCLUSIONS

This paper aims to provide a constructive step towards developing ML systems that are secure against adversarial attacks.

Specifically, we focus on the detection of phishing websites, which represent a widespread menace to information systems. Such context entails attackers that actively try to evade 'static' detection mechanisms via crafty, but ultimately simple tactics. Machine learning is a reliable tool to catch such phishers, but ML is also prone to evasion. However, realizing the evasion attempts considered by most past work requires a huge resource investment—which contradicts the very nature of phishing. To provide valuable research for ML security, the emphasis should be on attacks that are more likely to occur in the wild. We set this goal as our primary objective.

After dissecting the architecture of ML-PWD, we propose an original interpretation of attacks against ML systems by formalizing the EVASION-SPACE of adversarial perturbations. We then carry out a large evaluation of evasion attacks exploiting diverse 'spaces', focusing on those requiring less resources to be staged in reality.

TAKEAWAY: The findings of our paper are useful to both research and practice in the domain of adversarial ML.

- Our *evasion-space* formalization allows **researchers** to evaluate adversarial ML attacks without the risk of falling into the "unrealizable" perturbation trap (as long as the corresponding cost is factored in).
- Our *results* raise an alarm for **practitioners**: some ML-PWD can be evaded with simple tactics that do not rely on gradient computations, days of bruteforcing, or extensive intelligence gathering campaigns.

Acknowledgement. We thank the anonymous reviewers and our shepherd Amin Kharraz; the organizers of MLSEC for releasing their ML-PWD for research; and the Hilti corporation for funding.

REFERENCES

[1] 2020. *Interet Crime Report.* Technical Report. Federal Bureau of Investigation. https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf
[2] 2020. *On Artificial Intelligence - A European approach to excellence and trust.* Technical Report. European Commission. 27 pages. https://ec.europa.eu/info/sites/info/files/commission-white-paper-artificial-intelligence-feb2020_en.pdf
[3] 2021. *S&T Artificial Intelligence and Machine Learning Strategic Plan.* Technical Report. US Department of Homeland Security. 24 pages. https://www.dhs.gov/sites/default/files/publications/21_0730_st_ai_ml_strategic_plan_2021.pdf

[4] 2022. All Adversarial Examples Papers. https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html.
[5] 2022. Machine Learning Security Evasion Competition. https://mlsec.io/.
[6] 2022. PhishTank. https://phishtank.org/.
[7] 2022. *State of the Phish 2022.* Technical Report. ProofPoint. https://www.proofpoint.com/it/resources/threat-reports/state-of-phish
[8] Sahar Abdelnabi, Katharina Krombholz, and Mario Fritz. 2020. VisualPhishNet: Zero-day phishing website detection by visual similarity. In *Proc. ACM SIGSAC Conf. Comp. Commun. Secur.*
[9] Bhupendra Acharya and Phani Vadrevu. 2021. {PhishPrint}: Evading Phishing Detection Crawlers by Prior Profiling. In *USENIX Security Symposium.*
[10] Rayah Al-Qurashi, Ahmed AlEroud, Ahmad A Saifan, Mohammad Alsmadi, and Izzat Alsmadi. 2021. Generating Optimal Attack Paths in Generative Adversarial Phishing. In *Proc. IEEE Int. Conf. Intell. Secur. Inf.*
[11] Ahmed AlEroud and George Karabatis. 2020. Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks. In *Proc. Int. Workshop Secur. Privacy Anal.*
[12] Giovanni Apruzzese, Mauro Andreolini, Luca Ferretti, Mirco Marchetti, and Michele Colajanni. 2021. Modeling Realistic Adversarial Attacks against Network Intrusion Detection Systems. *ACM Digital Threats: Res. and Practice* (2021).
[13] Giovanni Apruzzese and Michele Colajanni. 2018. Evading botnet detectors based on flows and Random Forest with adversarial samples. In *Proc. IEEE Int. Symp. Netw. Comput. Appl.* 1–8.
[14] Giovanni Apruzzese, Michele Colajanni, Luca Ferretti, and Mirco Marchetti. 2019. Addressing Adversarial Attacks against Security Systems based on Machine Learning. In *Proc. IEEE Int. Conf. Cyber Conflicts.* 1–18.
[15] Giovanni Apruzzese, Pavel Laskov, Edgardo Montes de Oca, Wissam Mallouli, Luis Burdalo Rapa, Athanasios Vasileios Grammatopoulos, and Fabio Di Franco. 2022. The Role of Machine Learning in Cybersecurity. *ACM Digital Threats: Research and Practice* (2022).
[16] Giovanni Apruzzese, Pavel Laskov, and Aliya Tastemirova. 2022. SoK: The Impact of Unlabelled Data in Cyberthreat Detection. In *Proc. IEEE EuroS&P.*
[17] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and Don'ts of Machine Learning in Computer Security. In *USENIX Secur. Symp.*
[18] Trinh Nguyen Bac, Phan The Duy, and Van-Hau Pham. 2021. PWDGAN: Generating Adversarial Malicious URL Examples for Deceiving Black-Box Phishing Website Detector using GANs. In *Proc. IEEE Int. Conf. Machin. Learn. Appl. Netw.*
[19] Eugene Bagdasaryan and Vitaly Shmatikov. 2021. Blind backdoors in deep learning models. In *USENIX Sec. Symp.*
[20] Alejandro Correa Bahnsen, Ivan Torroledo, Luis David Camacho, and Sergio Villegas. 2018. DeepPhish: simulating malicious AI. In *Proc. APWG Symp. Elec. Crime Res.*
[21] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Elsevier Pattern Recogn.* 84 (2018), 317–331.
[22] Khalid Binsaeed, Gianluca Stringhini, and Ahmed E Youssef. 2020. Detecting Spam in Twitter Microblogging Services: A Novel Machine Learning Approach based on Domain Popularity. *Int. J. Adv. Comput. Sci. Appl* (2020).
[23] Franziska Boenisch, Verena Battis, Nicolas Buchmann, and Maija Poikela. 2021. "I Never Thought About Securing My Machine Learning Systems": A Study of Security and Privacy Awareness of Machine Learning Practitioners. In *ACM Mensch und Computer.*
[24] Andrei Butnaru, Alexios Mylonas, and Nikolaos Pitropakis. 2021. Towards lightweight URL-based phishing detection. *Future internet* (2021).
[25] Deanna D Caputo, Shari Lawrence Pfleeger, Jesse D Freeman, and M Eric Johnson. 2013. Going spear phishing: Exploring embedded training and awareness. *IEEE Security & Privacy* (2013).
[26] Nicholas Carlini. 2021. Poisoning the Unlabeled Dataset of {Semi-Supervised} Learning. In *USENIX Secur. Symp.*
[27] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. 2019. On evaluating adversarial robustness. *arXiv:1902.06705* (2019).
[28] Nicholas Carlini and David Wagner. 2016. Defensive distillation is not robust to adversarial examples. *arXiv:1607.04311* (2016).
[29] Lingwei Chen, Yanfang Ye, and Thirimachos Bourlai. 2017. Adversarial machine learning in malware detection: Arms race between evasion attack and defense. In *Europ. Intell. Secur. Inf. Conf.*
[30] Igino Corona, Battista Biggio, Matteo Contini, Luca Piras, Roberto Corda, Mauro Mereu, Guido Mureddu, Davide Ariu, and Fabio Roli. 2017. Deltaphish: Detecting phishing webpages in compromised websites. In *Proc. Springer Europ. Symp. Res. Comput. Secur.* 370–388.
[31] Darktrace. 2020. *Machine Learning in the Age of Cyber AI.* Technical Report. https://www.darktrace.com/es/resources/wp-machine-learning.pdf
[32] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. 2017. Yes, machine learning can be more secure! A case study on android malware detection. *IEEE T. Dependable Secure Comp.* (2017).
[33] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. 2019. Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks. In *Proc. USENIX Secur. Symp.* 321–338.
[34] Sevtap Duman, Kubra Kalkan-Cakmakci, Manuel Egele, William Robertson, and Engin Kirda. 2016. Emailprofiler: Spearphishing filtering with header and stylometric features of emails. In *IEEE Ann. Comp. Software Appl. Conf.*
[35] Simone Fischer-Hübner, Cristina Alcaraz, Afonso Ferreira, Carmen Fernandez-Gago, Javier Lopez, Evangelos Markatos, Lejla Islami, and Mahdi Akil. 2021. Stakeholder perspectives and requirements on cybersecurity in Europe. *Elsevier J. Inf. Secur. Appl.* (2021).
[36] Gilad Gressel, Niranjan Hegde, Archana Sreekumar, and Michael Darling. 2021. Feature Importance Guided Attack: A Model Agnostic Adversarial Attack. *arXiv:2106.14815* (2021).
[37] Zhen Guo, Jin-Hee Cho, Ray Chen, Srijan Sengupta, Michin Hong, and Tanushree Mitra. 2022. SAFER: Social Capital-Based Friend Recommendation to Defend against Phishing Attacks. In *Int. AAAI Conf. Web and Social Media.*
[38] Brij B Gupta, Krishna Yadav, Imran Razzak, Konstantinos Psannis, Arcangelo Castiglione, and Xiaojun Chang. 2021. A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment. *Elsevier Comp. Commun.* (2021).
[39] Payas Gupta, Roberto Perdisci, and Mustaque Ahamad. 2018. Towards measuring the role of phone numbers in twitter-advertised spam. In *ACM AsiaCCS.*
[40] Abdelhakim Hannousse and Salima Yahiouche. 2021. Towards benchmark datasets for machine learning based website phishing detection: An experimental study. *Elsevier Eng. Appl. Artifi. Intell.* (2021).
[41] Shuichiro Haruta, Fumitaka Yamazaki, Hiromu Asahina, and Iwao Sasase. 2019. A Novel Visual Similarity-based Phishing Detection Scheme using Hue Information with Auto Updating Database. In *Proc. IEEE Asia-Pacific Conf. Commun.*
[42] Grant Ho, Asaf Cidon, Lior Gavish, Marco Schweighauser, Vern Paxson, Stefan Savage, Geoffrey M Voelker, and David Wagner. 2019. Detecting and characterizing lateral phishing at scale. In *Proc. USENIX Security Symp.*
[43] David Howell. 2015. Building better data protection with SIEM. *Elsevier Computer Fraud & Security* (2015).
[44] Mohith Gowda HR, Adithya MV, et al. 2020. Development of anti-phishing browser based on random forest and rule of extraction framework. *Cybersecurity* (2020).
[45] Ankit Kumar Jain and Brij B Gupta. 2018. Towards detection of phishing websites on client-side using machine learning based approach. *Telecom. Syst.* (2018).
[46] Ankit Kumar Jain and Brij B Gupta. 2019. A machine learning based approach for phishing detection using hyperlinks information. *J. Ambient Intell. Human. Comp.* (2019).
[47] Jinyuan Jia, Binghui Wang, Xiaoyu Cao, Hongbin Liu, and Neil Zhenqiang Gong. 2022. Almost tight l0-norm certified robustness of top-k predictions against adversarial perturbations. *Int. Conf. Learn. Repr.* (2022).
[48] Michael I Jordan and Tom M Mitchell. 2015. Machine Learning: Trends, Perspectives, and Prospects. *Science* 349, 6245 (2015), 255–260.
[49] Houssain Kettani and Polly Wainwright. 2019. On the Top Threats to Cyber Systems. In *Proc. IEEE Int. Conf. Inf. Comp. Tech.* 175–179.
[50] Doowon Kim, Haehyun Cho, Yonghwi Kwon, Adam Doupé, Sooel Son, Gail-Joon Ahn, and Tudor Dumitras. 2021. Security Analysis on Practices of Certificate Authorities in the HTTPS Phishing Ecosystem. In *Proc. ACM AsiaCCS.*
[51] Engin Kirda and Christopher Kruegel. 2005. Protecting users against phishing attacks with antiphish. In *IEEE Annual Int. Comp. Soft. Appl. Conf.*
[52] Brian Kondracki, Babak Amin Azad, Oleksii Starov, and Nick Nikiforakis. 2021. Catching Transparent Phish: Analyzing and Detecting MITM Phishing Toolkits. In *ACM Conf. Comp. Commun. Secur.*
[53] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. 2020. Adversarial machine learning-industry perspectives. In *IEEE Secur. Privacy Workshops.*
[54] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436.
[55] Jehyun Lee, Pingxiao Ye, Ruofan Liu, Dinil Mon Divakaran, and Mun Choon Chan. 2020. Building robust phishing detection system: an empirical analysis. *Proc. Netw. Distrib. Syst. Symp. – MADWeb Workshop* (2020).
[56] Qizhang Li, Yiwen Guo, and Hao Chen. 2020. Practical no-box adversarial attacks against DNNs. *NeurIPS* 33 (2020), 12849–12860.
[57] Bin Liang, Miaoqiang Su, Wei You, Wenchang Shi, and Gang Yang. 2016. Cracking classifiers for evasion: a case study on the Google's Phishing pages filter. In *Proceedings of the 25th International Conference on World Wide Web.* 345–356.
[58] Cong Liao, Haoti Zhong, Sencun Zhu, and Anna Squicciarini. 2018. Server-based manipulation attacks against machine learning models. In *ACM Conf. Data and Application Security and Privacy.*
[59] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. 2021. Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages. In *Proc. USENIX Secur. Symp.*

[60] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. 2014. Intelligent rule-based phishing websites classification. *IET Inf. Secur.* (2014).
[61] Tyler Moore. 2010. The economics of cybersecurity: Principles and policy options. *Elsevier Int. J. Critical Infrastructure Protection* (2010).
[62] Jiaming Mu, Binghui Wang, Qi Li, Kun Sun, Mingwei Xu, and Zhuotao Liu. 2021. A hard label black-box adversarial attack against graph neural networks. In *ACM SIGSAC Conf. Comp. Commun. Secur.*
[63] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. 2021. Defeating {DNN-Based} Traffic Analysis Systems in {Real-Time} With Blind Adversarial Perturbations. In *USENIX Security Symposium.*
[64] Amirreza Niakanlahiji, Bei-Tseng Chu, and Ehab Al-Shaer. 2018. PhishMon: A Machine Learning Framework for Detecting Phishing Webpages. In *Proc. IEEE Int. Conf. Intel. Secur. Inf.*
[65] Adam Oest, Yeganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, and Adam Doupé. 2020. PhishTime: Continuous Longitudinal Measurement of the Effectiveness of Anti-phishing Blacklists. In *USENIX Sec.*
[66] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. 2020. Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In *Proc. USENIX Secur. Symp.*
[67] Alexander O'Mara, Izzat Alsmadi, and Ahmed AlEroud. 2021. Generative Adversarial Analysis of Phishing Attacks on Static and Dynamic Content of Webpages. In *Proc. IEEE Int. Conf. Parallel Distrib. Proc. Appl.*
[68] Ren Pang, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2020. AdvMind: Inferring adversary intent of black-box attacks. In *ACM Int. Conf. Knowl. Discov. Data Mining.*
[69] Thomas Kobber Panum, Kaspar Hageman, René Rydhof Hansen, and Jens Myrup Pedersen. 2020. Towards adversarial phishing detection. In *Proc. USENIX Workshop Cyber Security Exp. Test.*
[70] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proc. ACM Conf. Comput. Commun. Secur.* 506–519.
[71] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. 2018. SoK: Security and Privacy in Machine Learning. In *Proc. IEEE Europ. Symp. Secur. Privacy.* 399–414.
[72] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symp. Secur. Privacy.*
[73] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. 2020. Intriguing Properties of Adversarial ML Attacks in the Problem Space. In *IEEE Symp. Secur. Privacy.*
[74] Pawan Prakash, Manish Kumar, Ramana Rao Kompella, and Minaxi Gupta. 2010. Phishnet: predictive blacklisting to detect phishing attacks. In *IEEE InfoCOM.*
[75] Erwin Quiring, David Klein, Daniel Arp, Martin Johns, and Konrad Rieck. 2020. Adversarial Preprocessing: Understanding and Preventing {Image-Scaling} Attacks in Machine Learning. In *USENIX Secur. Symp.*
[76] Hemant Rathore, Swati Agarwal, Sanjay K Sahay, and Mohit Sewak. 2018. Malware detection using machine learning and deep learning. In *Springer Int. Conf. Big Data Analytics.* 402–411.
[77] Bushra Sabir, M Ali Babar, and Raj Gaire. 2020. An evasion attack against ML-based phishing URL detectors. *arXiv:2005.08454* (2020).
[78] Suhas R Sharma, Rahul Parthasarathy, and Prasad B Honnavalli. 2020. A Feature Selection Comparative Study for Web Phishing Datasets. In *Proc. IEEE Int. Conf. Elec. Comp. Commun. Tech.*
[79] Hossein Shirazi, Bruhadeshwar Bezawada, Indrakshi Ray, and Charles Anderson. 2019. Adversarial sampling attacks against phishing detection. In *IFIP Ann. Conf. Data Appl. Secur. Privacy.*
[80] Charles Smutz and Angelos Stavrou. 2012. Malicious PDF detection using metadata and structural features. In *Proc. ACM Ann. Comp. Secur. Appl. Conf.*
[81] Fu Song, Yusi Lei, Sen Chen, Lingling Fan, and Yang Liu. 2021. Advanced evasion attacks and mitigations on practical ML-based phishing website classifiers. *Int. J. Intell. Syst.* (2021).
[82] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE T. Evol. Comput.* (2019).
[83] Choon Lin Tan, Kang Leng Chiew, KokSheik Wong, et al. 2016. PhishWHO: Phishing webpage detection via identity keywords extraction and target domain name finder. *Elsevier Decis. Support Syst.* (2016).
[84] Lizhen Tang and Qusay H Mahmoud. 2021. A survey of machine learning-based solutions for phishing website detection. *Machine Learning and Knowledge Extraction* (2021).
[85] Ke Tian, Steve TK Jan, Hang Hu, Danfeng Yao, and Gang Wang. 2018. Needle in a haystack: Tracking down elite phishing domains in the wild. In *Proc, ACM Internet Measurement Conf.*
[86] Liang Tong, Bo Li, Chen Hajaj, Chaowei Xiao, Ning Zhang, and Yevgeniy Vorobeychik. 2019. Improving robustness of ML classifiers against realizable evasion attacks using conserved features. In *USENIX Secur. Symp.*
[87] Florian Tramèr, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, and Dan Boneh. 2019. Adversarial: Perceptual ad blocking meets adversarial machine learning. In *ACM Conf. Comp. Commun. Secur.*
[88] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble adversarial training: Attacks and defenses. In *Proc. Int. Conf. Learning Representations.*
[89] Bram Van Dooremaal, Pavlo Burda, Luca Allodi, and Nicola Zannone. 2021. Combining Text and Visual Features to Improve the Identification of Cloned Webpages for Early Phishing Detection. In *Proc. ACM Int. Conf. Availability, Reliability, Secur.*
[90] Kalyan Veeramachaneni, Ignacio Arnaldo, Vamsi Korrapati, Constantinos Bassias, and Ke Li. 2016. AIˆ 2: training a big data machine to defend. In *Proc. IEEE Int. Conf. Big Data Secur. Cloud.*
[91] Rakesh Verma and Keith Dyer. 2015. On the character of phishing URLs: Accurate and robust statistical learning classifiers. In *Proc. ACM CODASPY.*
[92] Wei Wei, Qiao Ke, Jakub Nowak, Marcin Korytkowski, Rafał Scherer, and Marcin Woźniak. 2020. Accurate and fast URL phishing detector: a convolutional neural network approach. *Elsevier Comp. Netw.* (2020).
[93] Kelce S Wilson and Müge Ayse Kiy. 2014. Some fundamental Cybersecurity concepts. *IEEE Access* (2014).
[94] Aiping Xiong, Robert W Proctor, Weining Yang, and Ninghui Li. 2019. Embedding training within warnings improves skills of identifying phishing webpages. *Human Factors* (2019).
[95] Changhoon Yoon, Kwanwoo Kim, Yongdae Kim, Seungwon Shin, and Sooel Son. 2019. Doppelgängers on the dark web: A large-scale assessment on phishing hidden web services. In *The World Wide Web Conference.*
[96] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, et al. 2021. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *IEEE Symp. Security and Privacy.*
[97] Baolin Zheng, Peipei Jiang, Qian Wang, Qi Li, Chao Shen, Cong Wang, Yunjie Ge, Qingyang Teng, and Shenyi Zhang. 2021. Black-box adversarial attacks on commercial speech platforms with minimal information. In *ACM CCS.*
[98] Nedim Å rndic and Pavel Laskov. 2014. Practical evasion of a learning-based classifier: A case study. In *Proc. IEEE Symp. Secur. Privacy.* 197–211.

A SUPPLEMENTARY TABLES AND FIGURES

We report in Table 1 the complete list of features of the ML-PWD considered in our paper. Table 2 shows some essential information on our datasets; Table 3 reports the baseline performance of our ML-PWD (developed through the workflow shown in Fig. 6); and Table 4 shows the related works discussed in §7.

Table 1: Features F of the considered ML-PWD.

#	Feature Name	#	Feature Name	#	Feature Name
1	URL_length	20	URL_shrtWordPath	39	HTML_commPage
2	URL_hasIPaddr	21	URL_lngWordURL	40	HTML_commPageFoot
3	URL_redirect	22	URL_DNS	41	HTML_SFH
4	URL_short	23	URL_domAge	42	HTML_popUp
5	URL_subdomains	24	URL_abnormal	43	HTML_rightClick
6	URL_atSymbol	25	URL_ports	44	HTML_domCopyright
7	URL_fakeHTTPS	26	URL_SSL	45	HTML_nullLnkWeb
8	URL_dash	27	URL_statisticRe	46	HTML_nullLnkFooter
9	URL_dataURI	28	URL_pageRank	47	HTML_brokenLnk
10	URL_commonTerms	29	URL_regLen	48	HTML_loginForm
11	URL_numerical	30	URL_checkGI	49	HTML_hiddenDiv
12	URL_pathExtend	31	URL_avgWordPath	50	HTML_hiddenButton
13	URL_punyCode	32	URL_avgWordHost	51	HTML_hiddenInput
14	URL_sensitiveWrd	33	URL_avgWordURL	52	HTML_URLBrand
15	URL_TLDinPath	34	URL_lngWordPath	53	HTML_iframe
16	URL_TLDinSub	35	URL_lngWordHost	54	HTML_favicon
17	URL_totalWords	36	HTML_freqDom	55	HTML_statBar
18	URL_shrtWordURL	37	HTML_objectRatio	56	HTML_css
19	URL_shrtWordHost	38	HTML_metaScripts	57	HTML_anchors

All features in Table 1 are used by both the ML-PWD targeted in our pragmatic use-case (cf. §B), as well as by the 'true baselines' ML-PWD (i.e., those analyzing F^c) used in our evaluation (cf. §5.1.3); in contrast, the 'robust' ML-PWD (i.e., those analyzing either F^u or F^r) consider subsets of the features in Table 1 (see §5.1.3).

Table 2: Statistics and state-of-the-art of our datasets.

Dataset	#Benign	#Phish	*fpr*	*tpr*
δphish [30]	5511	1012	0.01	0.98
Zenodo [89]	2000	2000	0.08	0.99

We mention that the original Zenodo contains 100k phishing, and almost 4M benign webpages. To make our evaluation "humanly feasible," we randomly sample 4000 webpages from Zenodo, equally split between benign and phishing. In such a way, we can analyze the response of ML-PWD having diverse *balancing*: while Zenodo is perfectly balanced, δPhish has significantly more benign samples.

Table 3: Performance in non-adversarial settings, reported as the average (and std. dev.) tpr and fpr over the 50 trials.

$\mathcal{A}$	F	Zenodo		δphish	
		tpr	fpr	tpr	fpr
CN	F^u	0.96±0.008	0.021±0.0077	0.55±0.030	0.037±0.0076
	F^r	0.88±0.018	0.155±0.0165	0.81±0.019	0.008±0.0020
	F^c	0.97±0.006	0.018±0.0088	0.93±0.013	0.005±0.0025
RF	F^u	0.98±0.004	0.007±0.0055	0.45±0.022	0.003±0.0014
	F^r	0.93±0.013	0.025±0.0118	0.94±0.016	0.006±0.0025
	F^c	**0.98**±0.006	**0.007**±0.0046	**0.97**±0.007	**0.001**±0.0011
LR	F^u	0.95±0.009	0.037±0.0100	0.24±0.017	0.011±0.0026
	F^r	0.82±0.017	0.144±0.0171	0.74±0.025	0.018±0.0036
	F^c	0.96±0.007	0.025±0.0077	0.81±0.020	0.013±0.0037

By comparing Table 3 with Table 2, we appreciate that our ML-PWD using F^c achieve comparable performance as prior work (even after our subsampling on Zenodo), confirming their relevance as baseline. Our repository includes the 4K pages we used for Zenodo.

Table 4: Adversarial attacks against ML-PWD. For each paper, we report: the *evasion space* (for simplicity we consider problem and feature-space); which *features* (F) are analyzed by the ML-PWD; the ML *algorithms* used by the ML-PWD (SL or DL); if some *defense* is evaluated; how many *datasets* are used (and if they are reproducible); and if the experiments are repeated for *statistical validation.*

Paper (1st Author)	Year	Evasion space	ML-PWD types (F)	ML Algorithms	Defense	Datasets (reprod.)	Stat. Val.
Liang [57]	2016	Problem	F^c	SL	✗	1 (✗)	✗
Corona [30]	2017	Feature	F^r, F^c	SL	✓	1 (✓)	✗
Bahnsen [20]	2018	Problem	F^u	DL	✗	1 (✗)	✗
Shirazi [79]	2019	Feature	F^c	SL	✗	4 (✓)	✓*
Sabir [77]	2020	Problem	F^u	SL, DL	✓	1 (✗)	✗
Lee [55]	2020	Feature	F^c	SL	✓	1 (✓)	✗
Abdelnabi [8]	2020	Problem	F^r	DL	✓	1 (✓)	✗
Aleroud [11]	2020	Both	F^u	SL	✗	2 (✓)	✗
Song [81]	2021	Problem	F^c	SL	✓	1 (✓*)	✗
Bac [18]	2021	Feature	F^u	SL, DL	✗	1 (✗)	✗
Lin [59]	2021	Feature	F^c	DL	✓	1 (✓)	✗
O'Mara [67]	2021	Feature	F^r	SL	✗	1 (✓)	✗
Al-Qurashi [10]	2021	Feature	F^u, F^c	SL, DL	✗	4 (✓)	✗
Gressel [36]	2021	Feature	F^c	SL, DL	✓	1 (✗)	✗
Ours		Both	F^u, F^r, F^c	DL, SL	✓	2 (✓)	✓

B PRAGMATIC USE-CASE

Let us showcase *how* an attacker can physically realize WsP leading to adversarial samples. We intend to demonstrate that WsP "can be done", and hence represent a (likely) threat that must be considered in a proactive development lifecycle of ML-PWD.

Target System. We consider the ML-PWD proposed in [45], whose architecture aligns with the one in Fig. 3. The corresponding $\mathcal{M}$ is a RF classifier trained on a dataset created ad-hoc through public feeds. The complete feature set F analyzed by $\mathcal{M}$ is reported in Table 1, which includes features related to both the URL and the representation of the website (based on the HTML). The ML-PWD extracts such features by inspecting the raw webpage according to

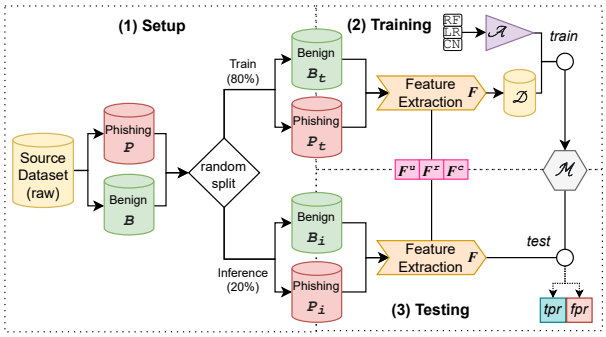


Fig. 6: Experimental workflow. Each source dataset (containing benign, B, and phishing, P, samples) is randomly split into the training (B_t and P_t) and inference (B_i and P_i) partitions, used to train and test each ML-PWD. We use P_i as basis for our adversarial samples.

the thresholds proposed in [60] (and also used in [45]). We observe that such methodology (and, hence, F) is also adopted by very recent works (e.g., [40, 78]). We provide more details in the Artifact.

Attacker. The attacker expects the usage of a ML-PWD, but they are agnostic of anything about the ML model $\mathcal{M}$, i.e., they are oblivious of the ML algorithm (i.e., RF) and its training data. The attacker, however, follows the state-of-the-art and hence knows the most popular feature sets used by ML-PWD (e.g., [78]). In particular, the attacker correctly guesses that the ML-PWD analyzes features related to both the URL and the representation of the webpage, and specifically the URL length and the objects embedded in the HTML. Formally: K=(URL_length, $HTML\_objectRatio$). The attacker, however, does not know the *exact* functionality of the feature extractor, the complete feature set F, and which features are more important for the final classification (the latter requires knowledge of $\mathcal{M}$). To provide a concrete example, we assume that the attacker owns the phishing[12] webpage shown in Fig. 7, whose URL is "https://www.63y3hfh-fj39f30-f30if0f-f392.weebly.com/".

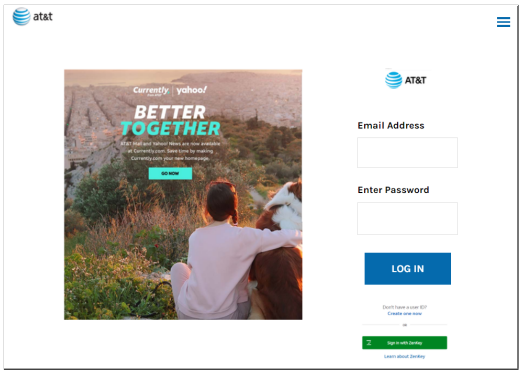


Fig. 7: An exemplary (and true) Phishing website, whose URL is https://www.63y3hfh-fj39f30-f30if0f-f392.weebly.com/.

Real Perturbations. To craft perturbations in the website-space (i.e., WsP) that affect $K \subset F$, the attacker can do the following.

(1) *Modify the HTML.* The attacker knows that phishing websites have many links that point to external domains[13] with respect to

[12] PhishTank reports such webpage to be a true and verified phishing (March 2022).
[13] E.g., phishing associated with AT&T will have many links pointing to the real AT&T.

internal resources (which would require to invest more into web-hosting). Hence, the attacker can introduce (in the HTML) a high number of 'fake links' that point to non-existent internal resources, which will affect the ratio of internal-to-external objects (making it more even). Such fake links, however, are can be made invisible (by exploiting some CSS properties) to users, who will not notice any difference[14]. We provide a visual representation of such WsP in Fig. 8, showing a snippet of the HTML of the original phishing webpage (cf. Fig. 7); the red rectangles denote two exemplary 'perturbations', i.e., the introduction of (hidden) links pointing to an internal resource (which may not exist). Note that such WsP does not break the website's functionality, and can be cheaply introduced anywhere (and many times) in the source HTML. Similar WsP are feasible and will[15] influence the *HTML_objectRatio* (included in K).

(2) *Modify the URL.* The attacker knows that long URLs are suspicious. So the attacker can, e.g., use a URL-shortening service (e.g., bit.ly) to alter the length of the phishing URL. In our case, the original URL (of 52 characters) can be shrunk to "https://bit.ly/3MZHjt7" (of 14 characters), thereby resulting in a completely different URL. Such a WsP will affect many features analyzed by $\mathcal{M}$ (cf. Table 1). Such features are not included in K, and hence their modifications are beyond the attacker's knowledge. The shrunk URL can then be distributed by the attacker in the wild[16].

(3) *Both of the above.* The attacker can easily perturb both the URL and HTML to induce perturbations of higher impact.

Observation. None of these WsP are guaranteed to evade the ML-PWD: A short URL is not necessarily benign, and having a non-suspicious ratio of internal-to-external objects is also not a strict requirement for being a benign webpage. The WsP could even be useless in the first place, e.g., the original URL could be already 'short'. Indeed, our attacker is not aware of what happens inside the ML-PWD. The problem, however, is that such uncertainty is shared by both the attacker (who cannot observe the ML-PWD) and the defender (who cannot pinpoint what the attacker does). To reveal the uncanny effects of such WsP, we assess them in §6

C THREAT MODEL: CONSIDERATIONS

Let us enhance our threat model with four considerations.

(1) The attacker can easily acquire a rough idea of the feature set F analyzed by the ML-PWD. For instance, the descriptions of many state-of-the-art solutions are openly accessible. However, it is unlikely that the attacker knows the *exact* feature set F: the actual implementation of a ML-PWD (including the feature extractor) can – or, rather, should! – differ from the publicly available information. This is why we consider an attacker that only knows $K \subseteq F$.

(2) We note that it is also possible that $K=\emptyset$. In this case, the attacker expects the ML-PWD to analyze some features that are *not* actually analyzed by $\mathcal{M}$ (for instance, the attacker can modify the URL, but nothing about the URL is analyzed by $\mathcal{M}$). This can

Fig. 8: A perturbation ε in the website-space (WsP). The original HTML (related to the website in Fig. 7) is modified by introducing *hidden link*(s). Such WsP will not be noticed by a user.

happen, e.g., against an 'adversarially robust' ML-PWD that leverages the well-known *feature removal* strategy (cf. §2.3). As a result, WsP targeting such K will likely result in a negligible impact. Furthermore, it is also possible that some features in K simply *cannot* be influenced by an attacker operating in the website-space (e.g., features that depend on third-party sources, such as DNS logs).

(3) Since our attacker cannot access the ML-PWD, they cannot observe the output-space and, thus, cannot optimize their perturbations to find the best WsP that guarantees evasion; and cannot even verify whether their WsP evade the ML-PWD or not. The attacker is, however, not subject to strict boundaries on WsP (§3.2).

(4) Our threat model considers attacks at *inference*-time (i.e., after $\mathcal{M}$ has been deployed in the PWD). This is because the dataset used to devise ML-based security systems is typically well-protected [12]. Compromising such dataset would significantly raise the cost of the offensive campaign (as also highlighted in [58]). Therefore, phishers are unlikely to launch attacks at training-time.

The last two are significant: lack of access (and, hence, knowledge) on the training set prevents from achieving the no-box attacks of [56]; furthermore, the impossibility of witnessing the output of $\mathcal{M}$ prevents enacting typical black-box strategies (e.g., [62]).

D EXPERIMENTS: CONSIDERED ATTACKS

In our paper, we consider a total of 12 evasion attacks, divided in four families. One of these families is an *exact replica* of our 'standard' threat model. The remaining three families, however, are *extensions* of our threat model, which assume more 'advanced' adversaries who have superior knowledge and/or capabilities.

Two of our families involve WsP (WA and $\widehat{\text{WA}}$), but assume attackers with different knowledge; whereas the remaining two families involve either PsP or MsP (PA and MA). Each family has three variants depending on the features 'targeted' by the attacker, i.e., either those related to the URL, the HTML, or a combination of both (u, r, or c). For WsP, the underlying 'attacked' features are always the same for all variants, which are assumed to be known by the attacker: u is always the *URL_length*; for r is the *HTML_objectRatio*; and for c they are both of these. (Do note that our WsP will affect also features beyond the attacker's knowledge.)

- *Cheap Website Attacks* (WA) perfectly align with our threat model (ans resemble the use-cases in Appendix B). The perturbations are created in the website-space (WsP), realizing

[14] N.b.: complete 'invisibility' is not a strict requirement. Some WsP can be 'spotted' by a detailed analysis, but users may not notice them while still being phished. E.g., a link can be deleted; or a WsP can wrap: *<a href='link'>* into *<a onclick="this.href='link'">*.
[15] In theory, similar WsP could be detected by analyzing whether a given link is valid or not. Doing so, however, would pose an extremely high overhead: it requires checking every single link for every webpage that is analyzed by the ML-PWD.
[16] The ML-PWD will be fooled if it is *stateless* and does not visit all the redirections of the shortening service. Nevertheless, there are many ways to reduce the URL_length.

either WA^u, WA^r, or WA^c. Specifically for r (and c), we consider two semantically equivalent WsP: "add fake link" for δ`Phish`, and "link wrapping" for `Zenodo`. Such WsP attempt to balance the object ratio: the former by adding (invisible) links to (fake) internal objects, whereas the latter by eluding the preprocessing mechanism—thereby having a link not being counted among the total links shown in a webpage.

- *Advanced Website Attacks* ($\widehat{\mathsf{WA}}$), which envision a more knowledgeable attacker than WA. The attacker knows how the feature extractor within the ML-PWD operates (i.e., they know the specific thresholds used to compute some features). The attacker – who is still confined in the website-space – will hence craft more sophisticated WsP because they know how to generate an adversarial sample that is more likely to influence the ML-PWD. Thus, the attacker will modify either the URL, the HTML, or both (i.e., $\widehat{\mathsf{WA}^u}$, $\widehat{\mathsf{WA}^r}$, $\widehat{\mathsf{WA}^c}$), but in more elaborate ways—e.g., by ensuring that the *HTML_objectRatio* exactly resembles the one of a 'benign' sample; or by making an URL to be 'long enough' to be considered short.
- *Preprocessing Attacks* (PA), which are an extension of our threat model, and assume an even stronger attacker that is able to access the preprocessing stage of the ML-PWD, and hence introduce PsP. Such an attacker is capable of direct feature manipulation—subject to integrity checks (i.e., the result must reflect a "physically realizable" webpage). Since the attacker does not know anything about the actual $\mathcal{M}$, the attacker must still guess their PsP. Such PsP will target features based on either u, r, c (i.e., PA^u, PA^r, PA^c) by accounting for inter-dependencies between other features.
- *ML-space attacks* (MA), representing a worst-case scenario. The attacker can access the ML-space of the ML-PWD, and can hence freely manipulate the entire feature representation of their webpage through MsP. However, the attacker is still oblivious of $\mathcal{M}$, and must hence still guess their WsP. Thus, the MsP applied by the attacker completely 'flip' many features related to u, r, c (i.e., MA^u, MA^r, MA^c).

Motivation. We consider these 12 attacks for three reasons. First, to assess the effects of diverse *evasion attacks at increasing 'cost'*. For instance, the simplicity of WA makes them the most likely to occur; whereas MA can be disruptive, but are very expensive (from the attacker's viewpoint). Second, to study the response of ML-PWD to WsP targeting the same features (WA^r), but in different ways (one per dataset), leading to alterations of *different features beyond the attacker's knowledge*. Third. to highlight the *effects of potential 'pitfalls'* of related researches. Indeed, we observe that all three remaining families ($\widehat{\mathsf{WA}}$, PA, MA) envision attackers with similar knowledge which they use to target *similar features*. Such peculiarity allows comparing attacks carried out in different 'spaces.' A particular focus is on PA, for which we apply PsP by *anticipating* how a WsP can yield a physically realizable [86] PsP. Put differently, our evaluation shows what happens if the perturbations are applied without taking into account *all* preprocessing operations that transform a given x into the F_x analyzed by $\mathcal{M}$.

Implementation. We follow three steps: isolate, perturb, evade. We refer to the Artifact and source-code for the low-level details.

(1) *Isolate.* Our threat model envisions evasion attacks that occur during inference, hence our adversarial samples are generated from those in P_i. Furthermore, we recall that the attacker expects the ML-PWD to be effective against 'regular' malicious samples (cf. §4.1). To meet such condition, we isolate 100 samples from P_i that are detected successfully by the best ML-PWD (typically using F^c). Such samples are then used as basis to craft all the adversarial samples (through WsP, PsP or MsP) of our evaluation—thereby ensuring that *all detectors are assessed against the exact same adversarial samples* (which is necessary for a fair comparison).
(2) *Perturb.* We apply the perturbations as follows. For WA and $\widehat{\mathsf{WA}}$, we craft the corresponding WsP, apply them to each of the 100 samples from P_i, and then preprocess such samples by using the feature extractor. For PA and MA, we first preprocess the 100 samples with the feature extractor, and then apply the corresponding PsP or MsP. Overall, these operations result in 1200 adversarial samples (12*100).
(3) *Evade.* The 1200 adversarial samples are then sent to all the 9 ML-PWD (for each dataset), and we measure the tpr again.

We expect the tpr on the adversarial samples (generated by any of our 12 considered attacks) to be lower than the tpr on the originals.

Effectiveness and Affordability. In terms of effectiveness, assuming the same targeted features, $\mathsf{WA} < \widehat{\mathsf{WA}} < \mathsf{PA} \ll \mathsf{MA}$ (§6.2). This is justified by the higher investment required by the attacker, who must either perform extensive intelligence gathering campaigns (to understand the exact feature extractor for $\widehat{\mathsf{WA}}$) or gain write-access to the ML-PWD (for PA and MA). Let us provide a high-level summary of the requirements to implement all our attacks—all of which are *query-less* and rely on *blind* perturbations.

- WA: they require as little as a dozen lines of elementary code, and a very rough understanding of how ML-PWD operate (which can be done, e.g., by reading research papers).
- $\widehat{\mathsf{WA}}$: they also require a few lines of code to implement. However, determining the exact thresholds requires a detailed intelligence gathering campaign (or many queries to reverse-engineer the ML-PWD, if it is client-side).
- PA: they require a compromise of the ML-PWD. For example, introducing a special 'backdoor' rule that "if a given URL is visited, then do not compute its length and return that the URL is *short*". Doing this is costly, but it is not unfeasible if the feature extractor is open-source (e.g., [19]).
- MA: they also require a compromise of the ML-PWD. In this case, the 'backdoor' is introduced *after* all features have been computed—and irrespective of their relationships. Hence. the cost is very high: the ML model is likely to be tailored for a specific environment, thereby increasing the difficulty of successfully introducing such backdoors in one of the deepest segments of the ML-PWD.

Hence, in terms of affordability: $\mathsf{WA} \gg \widehat{\mathsf{WA}} \gg \mathsf{PA} > \mathsf{MA}$ (i.e., the relationship is the reverse of the effectiveness). For this reason, in our evaluation we will put a greater emphasis on WA, because 'cheaper' attacks are more likely to occur *in the wild*: while WA can be associated with "horizontal phishing" (the majority), the others are tailored for "spear phishing" (the minority).