Multi-SpacePhish: Extending the Evasion-space of Adversarial Attacks against Phishing Website Detectors using Machine Learning

YING YUAN, † University of Padua, Italy
GIOVANNI APRUZZESE, University of Liechtenstein, Liechtenstein
MAURO CONTI†, Delft University of Technology, Netherlands

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 *feasibility* 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 an 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. After that, 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 ($p = 0.22$).

Finally, as an additional contribution of this journal publication, we are the first to propose and empirically evaluate the intriguing case wherein an attacker introduces perturbations in multiple evasion-spaces *at the same time*. These new results show that simultaneously applying perturbations in the problem- and feature-space can cause a drop in the detection rate from 0.95 to 0.

Our contribution paves the way for a much-needed re-assessment of adversarial attacks against ML systems for cybersecurity.

CCS Concepts: • **Security and privacy → Phishing**; • **Computing methodologies → Machine learning**.

Additional Key Words and Phrases: websites, features, detection, webpage, deep learning, artificial intelligence

ACM Reference Format:
Ying Yuan, Giovanni Apruzzese, and Mauro Conti. 2018. Multi-SpacePhish: Extending the Evasion-space of Adversarial Attacks against Phishing Website Detectors using Machine Learning. In . ACM, New York, NY, USA, 50 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

After more than a decade of research [24] and thousands of papers [5], it is well-known that Machine Learning (ML) methods are vulnerable to "adversarial attacks"[1]. Specifically, by introducing imperceptible perturbations (down to a single pixel or byte [15, 88]) in the input data, it is possible to compromise the predictions made by an ML model. Such vulnerability, however, is more dangerous in settings that implicitly assume the presence of adversaries. A cat will not try to fool an ML model. An attacker, in contrast, will actively try to evade an ML detector—the focus of this paper.

[1]To embrace a recommendation of a recent work [13], we use the term "adversarial attacks" to denote "attacks reliant on adversarial perturbations".

Manuscript submitted to ACM

On the surface, the situation portrayed in research is vexing. The confirmed successes of ML [52] are leading to large-scale deployment of ML in production settings (e.g., [34, 81, 90]). At the same time, however, dozens of papers showcase adversarial attacks that can crack 'any' ML-based detector (e.g., [16, 61]). Although some papers propose countermeasures (e.g., [77]), they are quickly defeated (e.g., [31]), and typically decrease the baseline performance (e.g. [16, 35]). As a result, recent reports [38, 57] focusing on the integration of ML *in practice* reveal that: "I Never Thought About Securing My Machine Learning Systems" [26]. 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., [3, 4]) are endorsing the development of "trustworthy" ML systems; yet, any enhancement should be economically justified. No system is foolproof (ML-based or not [29]), 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 [66]. Real attackers have a cost/benefit mindset [99]: 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. [78] 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., [68]), then there is a risk of generating an adversarial example that is not physically realizable [92]. Apruzzese et al. [14], 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 threats" [24]. Only *after* proactively assessing the impact of such threats a suitable countermeasure can be developed—if required.

We aim to promote the development 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 [8, 53] – is an endless struggle. Blocklists can be easily evaded [91] and, to cope against adaptive attackers, some detectors are equipped with ML (e.g. [90]). Yet, as shown by Liang et al. [61], 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., [23, 40, 59, 85]) assumed threat models that hardly resemble a real scenario. Phishing, by nature, is meant to be cheap [54] and most attempts end up in failure [71]. It is unlikely[2] 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 fundamental 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 five contributions.

[2]It is unlikely, but *not impossible*. Hence, as recommended by Arp et al [20], it is positive that such cases have also been studied by prior work.

- 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 an 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 leads to higher cost).
- We combine and practically demonstrate the two previous contributions. We perform an extensive, reproducible, and statistically validated *evaluation of adversarial attacks* against state-of-the-art ML-PWD. By using diverse datasets, ML algorithms and features, we develop 18 ML-PWD (§5), each of which is assessed against 12 different evasion attacks built upon our threat model (§6).
- By analyzing the results (§7) of our evaluation: (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.
- As an additional contribution of this journal paper, we propose and empirically assess 6 new URL-related perturbations, as well as 37 new HTML-related perturbations that envision an attacker who can *operate in multiple spaces* (§8).

Our results highlight that more realistic attacks are not as disruptive as claimed by past works (§9) but their low-cost makes them a threat that induces statistically significant degradation. Intriguingly, however, some "cheap" perturbations can lead to devastating impacts.

Finally, our evaluation serves as a 'benchmark' for future studies: we provide the complete results in the Appendix, whereas the source-code and additional resources are publicly available at 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 [55], phishing attacks are still a rampant menace [53]: according to the FBI [2], 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 [41]), 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* [100]. Despite recent positive trends, however, such education is far from comprehensive: the latest "State of the Phish" report [8] 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.

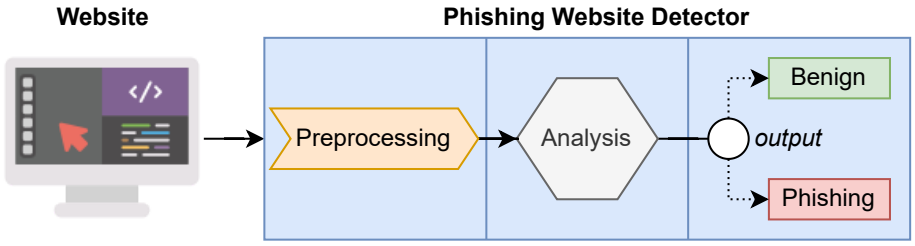


Fig. 1. **Exemplary PWD.** After preliminary preprocessing, a website is analyzed by a detector to determine its legitimacy.

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., [70, 79]). Such techniques have been improved and nowadays they even involve automatic updates with recent feeds (e.g., PhishTank [7]). 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 [10]. The latter is a problem: expert attackers are aware of blocklists and hence move their phishing 'hooks' from site to site, bypassing most PWD. As shown by Tian et al. [91], 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 [90]. 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" [52], 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* an 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, an 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.

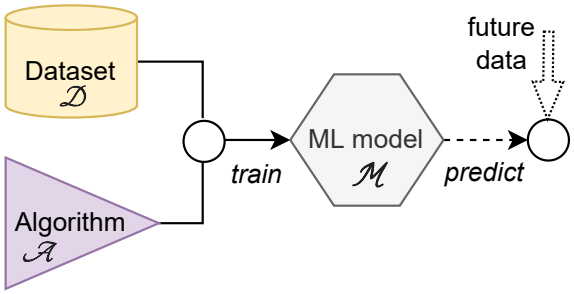


Fig. 2. **Machine Learning workflow.** By training $\mathcal{A}$ on $\mathcal{D}$, an ML model $\mathcal{M}$ is developed, which can be used to predict future data.

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. [91] show that an 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 [19].

Such advantages have been successfully leveraged by many research efforts (e.g., [69, 89]). 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* [97] or to its *representation*, e.g., by analyzing the actual image of a webpage as rendered by the browser [45], or by inspecting the HTML [50]. For example, Mohammad et al. [64] 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., [27]) – which can also be integrated into phishing *email* filters (e.g., [42]) – more recent proposals use combinations of features (e.g., [33, 95]); potentially, such features can be derived by querying third-party services (e.g., DNS servers [49]).

The cost-effectiveness of ML-PWD increased their adoption: even commercial browsers (e.g., Google Chrome [61]) integrate ML models in their phishing filters (which can be further enhanced via customized add-ons [90]); moreover, ML-PWD can also be deployed in corporate SIEM [47]. 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 [16].

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 [24, 32]. Attacks against ML exploit *adversarial samples*, which leverage perturbations to the input data of an ML model that induce predictions favorable to the attacker. Even imperceptible perturbations can mislead proficient ML models: for instance, Su et al. [88] modify a single pixel of an image to fool an object detector; whereas Apruzzese et al. [15] evade botnet detectors by extending the network communications with few junk bytes.

An adversarial attack is described with a *threat model*, which explains the relationship of a given *attacker* with the *defender's system*. The attacker has a *goal* and, by leveraging their *knowledge* and *capabilities*, they will adopt a specific *strategy* [24]. Common terms associated with the attacker's knowledge are *white-box* and *black-box*: the former denotes attackers who know everything about the defender; whereas the latter denotes attackers who know nothing [75, 103]. 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 [77] and broken in [31]). Even "certified" defenses [51] 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. [30]). 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* [86], 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., [16]). Even when countermeasures have a small impact (e.g., [35]), this is not negligible in cyber-detection: attacks are a "needle in a haystack" [91], and even a 1% increase in false positives is detrimental [96]. Therefore, ML engineers will not devise any protection mechanism unless the corresponding threat is shown to be dangerous in reality [57].

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 [88]); or vice versa, i.e., show that even oblivious attackers can thwart ML systems [75]. 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 the 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 [14]. In the context of ML-PWD, our paper is a first step in this direction: as we will show, evasion attempts evaluated in literature (§9), 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 an 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 an ML-PWD

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

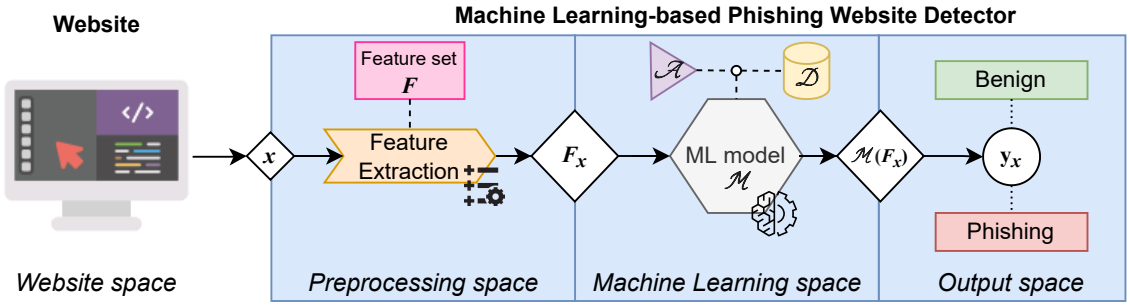


Fig. 3. Esattamente A website, x, is preprocessed into F_x. an ML model $\mathcal{M}$ analyzes such feature representation and predicts its ground truth as $\mathcal{M}(F_x) = y_x$.

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 an 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 an ML model $\mathcal{M}$ to provide an output favoring the attacker (§2.3). In our case, $\mathcal{M}$ is a (binary) classifier that analyzes F_x, hence we can express an adversarial attack as follows:

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

In other words, the objective is finding a perturbation ε that induces an 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*; and (ii) preserve the *phishing logic* of a webpage [74]. 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 [14]. Indeed, $\overline{F_x}$ must not violate any inter-feature dependencies or physical constraints. With respect to WsP, PsP are guaranteed to impact the feature representation of x; 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'

[3]Which depend on the semantics of websites, e.g., URLs cannot be 1 character long.

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 as $\bar{F}_x = F_x + \varepsilon$. As an example, an 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):

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

We remark that the effects of WsP can match those of PsP—which can also match those of MsP. For instance, an 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 considers perturbations in the "feature-space" (i.e., PsP or MsP). Let us explain why.

Context. By using Pierazzi et al. [78] notation, our definition of 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. [78] 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 necessarily weaknesses—therefore overturning some of the claims of Pierazzi et al. [78] in some cases, as their work assumes an implicit limitation of attacker capabilities that may not hold in practice. Indeed, our thesis is rooted in the following observation: the "inverse mapping problem" is irrelevant *if the attacker has write access to the ML-PWD.*

Practical 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

[4]Indeed, an ML model $\mathcal{M}$ is agnostic to the generation process of a given input.
[5]Of course MsP=PsP if there is no 'integrity check'.
[6]Our formalization is orthogonal to the one by Šrndic and Laskov. [105]: while [105] 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. [80]: while [80] affect the preprocessing phase *from outside* the ML system, our PsP affect such phase *from the inside*. Put simply, our formalization is the first of its kind in adversarial ML research.

integrating a client-side ML-PWD: such an attacker can interfere with any of the ML-PWD operations—especially if it is open-source (e.g., [48]). 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 [14]).

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

Consequences. Simply put, we *restore* the value (partially 'lost' after the publication of [78]) of the evaluations performed by prior work (§9). 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 an attacker is not subject to. Such different assumptions, however, implicitly raise the cost of the corresponding attack. For example. Corona et al. [33] craft perturbations in the ML-space: according to [78], the resulting perturbations are, hence, unreliable. However, by assuming that the attacker *can manipulate the ML-space*, then such adversarial examples (deemed unreliable by [78]) 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). Next, we provide some considerations (§4.3) that set-up the stage for the additional contribution of this paper (§4.4). Finally, we show how to apply WsP on *real* phishing webpages (§4.5).

4.1 Formal Definition

We define our threat model according to the following four criteria (well-known in adversarial ML [24]).

- *Goal.* The adversary wants to *evade* an ML-PWD that uses $\mathcal{M}$ (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 hence limited to perturbations in the website-space (i.e., WsP).
- *Strategy.* The adversary uses their knowledge of K to craft WsP that may lead to evasion (at the inference stage).

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 an ML-PWD used by email filters if the corresponding $\mathcal{M}$ analyzes URL-related information (e.g., [37, 42]).

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 [54]. 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 an 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 an 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 [8], most phishing attempts are prone to failure [71]—and the attackers are well aware of this fact. Of course, attackers can opt for more expensive spear-phishing campaigns [28] (which still have a success rate of barely 10% [46]); 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 [24] has ably demonstrated that white-box attacks can break most systems—including ML-PWD (e.g., [9, 40, 63, 87]).

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 [36]. 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 [18]). *Not interesting*, because such attacks *have been investigated before* [11, 82]. For instance, Liang et al. [61] demonstrated that attackers with access to client-side detectors can crack and evade the corresponding ML-PWD; doing this, however, required *more than 24 hours* of queries [61].

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

4.3 Technical 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 an 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 $K = \varnothing$. 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 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 [14]. Compromising such dataset would significantly raise the cost of the offensive campaign (as also highlighted in [62]). 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 [60]; furthermore, the impossibility of witnessing the output of $\mathcal{M}$ prevents enacting typical black-box strategies (e.g., [67]). Finally, as pointed out also by [13, 30, 104], achieving 'minimal' perturbations may be an unrealistic objective.

4.4 Extensions

Our threat model can be *extended* by relaxing some of its assumptions. Indeed, in its current formulation, our threat model envisions an attacker that is "weak" (and, hence, very likely to appear in reality). However, some adversaries may be willing to invest more resources to ensure that their attacks come to fruition (i.e., increasing the chances that their phishing webpages are misclassified by the ML-PWD, and hence displayed to the end-user). Abundant prior work in the adversarial ML domain considers attacks having different levels of *knowledge* (i.e., the so-called "black-box" and "white-box" [13]). However, given that our original formalization focuses on the attacker's *capabilities* (§3), we identify two types of extensions that portray a stronger attacker. Namely:

- *Deeper spaces.* An attacker who manages to obtain write-access to the ML-PWD (or part of its elements) can tamper with its internal functionalities, thereby realizing either PsP or MsP.
- *Mixed spaces.* If the attacker can obtain some control on either the Preprocessing- or Machine Learning-space, then – alongside being able to apply PsP or MsP – they are also able to apply WsP. Indeed, the attacker will *always* be able to manipulate the phishing webpage, since it is (by definition) under their complete control. Hence, an attacker who can inject PsP can also inject a WsP; furthermore, an attacker who can inject a MsP can also inject a PsP (since they can overlap), and can, of course, also inject a WsP.

We will empirically evaluate all the abovementioned cases in our evaluation (§7) 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). Moreover, we will also assess attacks entailing perturbations in different spaces (§8).

4.5 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 [49], 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 the

thresholds proposed in [64] (and also used in [49]). We observe that such methodology (and, hence, F) is also adopted by very recent works (e.g., [44, 84]). We provide more details in the next section (§5.1.3).

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

Attacker. The attacker expects the usage of an 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., [84]). 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[7] webpage shown in Fig. 4, 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:

- *Modify the HTML.* The attacker knows that phishing websites have many links that point to external domains[8] with respect to internal resources (which would require to invest more into webhosting). 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[9]. We provide a visual representation of such WsP in Fig. 5, showing a snippet of the HTML of the original phishing webpage (cf. Fig. 4); the red rectangles denote two exemplary 'perturbations', i.e., the introduction of (hidden) links pointing

[7] PhishTank reports such webpage to be a true and verified phishing (March 2022).
[8] E.g., phishing associated with AT&T will have many links pointing to the real AT&T.
[9] 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'">*.

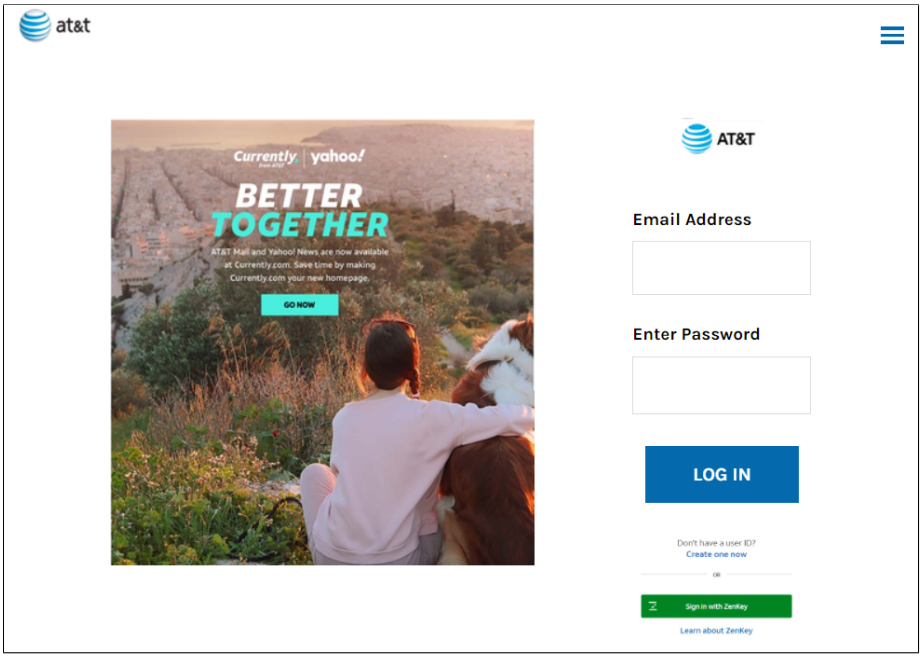


Fig. 4. **An exemplary (and true) Phishing website**, whose URL is https://www.63y3hfh-fj39f30-f30if0f-f392.weebly.com/.

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[10] influence the *HTML_objectRatio* (included in K).

- *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 shared in the wild[11].
- *Both of the above.* The attacker can perturb both the URL and HTML to induce perturbations of higher impact.

We observe that none of these WsP are guaranteed to evade the ML-PWD. Indeed, 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 exactly pinpoint what the attacker does). To reveal[12] the uncanny effects of such WsP, we assess them in §7.

[10] 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.
[11] The ML-PWD will be fooled if it does not visit all the redirections of the shortening service. Nevertheless, there are many ways to reduce the URL_length.
[12] **Remark.** 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 (§9). 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 $\mathcal{D}$

Fig. 5. **A perturbation ε in the website-space (WsP).** The original HTML (related to the website in Fig. 4) is modified by introducing *hidden link*(s). Such WsP will not be noticed by a user.

5 EVALUATION: EXPERIMENTAL SETUP AND TECHNICAL IMPLEMENTATION

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 [19].

We present our experimental testbed (§5.1), and then describe our technical implementation (§5.2). Finally, we measure the baseline performance of our ML-PWD (§5.3).

5.1 Testbed

We consider 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: δPhish and Zenodo [33, 95]. The reason is threefold.

- 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.

(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): all of these scenarios have already been investigated by past work (§9), and are hence outside our scope.

- Both datasets have been *used by the state of the art*. Prior research [33, 95] 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).

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

Dataset	#Benign	#Phish	*fpr*	*tpr*
δPhish [33]	5511	1012	0.01	0.98
Zenodo [95]	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.

5.1.2 ***ML Algorithms.*** We consider ML-PWD based on shallow and deep learning algorithms [16]. Our selection aims to provide a meaningful assessment of ML-PWD based on exemplary ML methods. 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 [61].
- Random Forests (*RF*). An ensemble technique, *RF* often outperforms other contenders for ML-PWD [91].
- Convolutional neural Network (*CN*). We consider this well-known deep learning technique [58] due to its demonstrated proficiency also in ML-PWD (e.g., [98]).

All of these algorithms support binary classification, making them appropriate for our ML-PWD.

5.1.3 ***Feature Sets.*** We consider ML-PWD that use three feature sets (F), all resembling the one described in our use-case (§4.5). 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 [33]. 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 an ML-PWD [72]: hence, a defender may develop an 'adversarially robust' detector that analyzes only the representation of a webpage. Sucha 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., [44, 49, 64, 84]), but also used in *practice*. Indeed, several leading security companies yearly organize MLSEC, an ML evasion competition [6]. In 2021 and 2022, MLSEC also involved evading ML-PWD *which specifically analyzed the HTML* [39] representation of a webpage—i.e., our F^r. We will also refer to MLSEC in our evaluation.

5.2 Technical Implementation

Let us describe how we combined all the elements described insofar to devise our "baseline" ML-PWD. We provide a schematic of our workflow in Fig. 6. Each source dataset (Zenodo and δPhish) 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. Finally, we ensure that such 9 ML-PWD maximize their performance (high tpr and low fpr, at least for F^c).

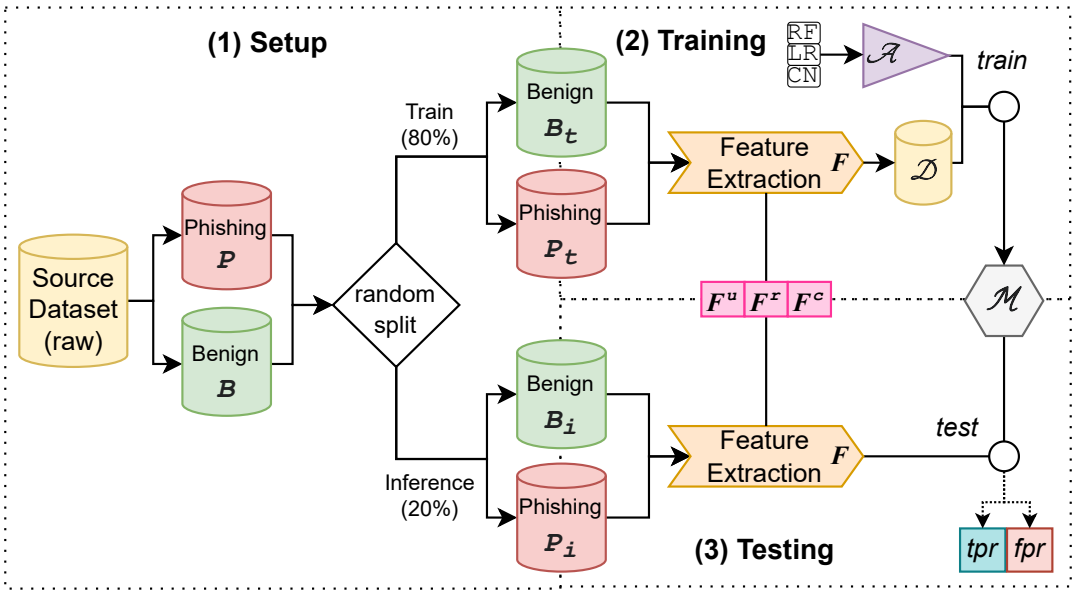


Fig. 6. **Experimental workflow.** Each dataset (containing benign, B, and phishing, P, samples) is randomly split into the training (B_t and P_t) and inference (B_i and P_i) partitions, used to train and test each ML-PWD. We use P_i as basis for our adversarial samples.

We now describe the implementation of our feature extractor (§5.2.1) and the development of our ML-PWD (§5.2.2).

5.2.1 ***Feature Extractor.*** An important part of our evaluation is represented by the feature extractor, for which we rely on the established guidelines provided in [64, 65] and still widely employed in recent literature (e.g., [49]). The underlying principle of such guidelines is to analyze several elements of a webpage (e.g., the length of its URL), and then use threshold-based mechanisms to determine whether such element is 'benign' or 'phishing' (e.g., a short URL is likely benign, whereas a long one is likely phishing). Any feature can have a value within [-1, 1], where -1 is 'benign' and 1 is 'phishing'. Our extractor generates all the features reported in Table 1. We explain some of them.

- (#1) *URL_length*. We compute the amount of character composing the entire URL. Strings shorter than 53 characters correspond to -1 (likely 'benign'), whereas longer strings correspond to +1 (likely 'phishing').
- (#28) *URL_pageRank*. We use OpenPageRank API to query the URL domain. The response shows the page ranks from 0 to 10: the corresponding feature is normalized between -1 (if the rank is 10) and +1 (if the rank is 0).

- (#4) *URL_short*. If the URL starts[13] with keywords related to popular shortening services (bit.ly, goo.gl, tinyurl, ad.fly) then this feature is set to +1, and to -1 otherwise.
- (#37) *HTML_objectRatio*. We capture all the objects embedded in the webpage, and compute the ratio of internal-to-external objects. An internal object either has its link starting with *../* or with the same 'root' as the website's URL. If the ratio is less than 0.15, then this feature is -1 (likely benign), and +1 otherwise (likely phishing).
- (#38) *HTML_metaScripts*. Same as #37, but for scripts, links and metas. If the ratio is more than 0.61, the feature value is +1 (likely phishing); if the ratio is less than 0.52, the feature value is -1 (likely benign); and 0 otherwise.
- (#45) *HTML_nullLnkWeb*. We check how many links are useless, i.e., they point to the exact same page (e.g., *href=#*). The count can be normalized between +1 (high number of useless links) and -1 (no useless links).
- (#51) *HTML_hiddenInput*. We check if there are any hidden *input* tags in the webpage. If there are, the feature value is +1 (likely phishing), and -1 otherwise (likely benign).
- (#52) *HTML_URLBrand*. We check (in the HTML) if the webpage *title* includes the brand name in the URL. If included, the feature value is -1 (benign); otherwise, is +1 (phishing).

(Our repository includes the source-code of our feature extractor.)

We use similar thresholds as those by Mohammad et al. [64, 65], and are the same used to create the popular UCI dataset [1]. To validate our choice of using the same thresholds (which play a crucial role in our evaluation), we find instructive to report the length of URLs contained in our chosen datasets, i.e., Zenodo and δPhish. The results are as follows: for Zenodo, there are 1500 URLs (out of 4000) which are *longer* than 54 characters; for δPhish, there are 1909 URLs (out of 6523) which are *longer* than 54 characters. Hence, such a threshold is still sensible for more recent datasets.

5.2.2 ***Development of the ML-PWD.*** We follow three phases (i.e., the three dotted squares in Fig. 6). Namely:

(1) *Setup*. The first phase is choosing a given source dataset (i.e., Zenodo or δPhish) and partition its samples into *benign* and *phishing* (B and P, respectively). Then, we perform a *random split* (to avoid bias) on each of these partitions by using a 80:20 ratio (common in related literature [11, 21]). In other words, we randomly select 80% of the samples in both B and P (i.e., B_t and P_t respectively), which will be used to train the ML model. The leftout samples, B_i and P_i (corresponding to 20% of B and P, respectively), are used to assess the inference performance of the resulting ML model. We will also use P_i as basis to craft our adversarial samples.

(2) *Training*. To train $\mathcal{M}$, we recall that the source data is in raw format. Hence, before obtaining the training dataset $\mathcal{D}$, the corresponding *training* partitions B_t and P_t must be transformed into their feature representation. Hence, we develop a feature extractor (described in §5.1.3) that is based on a given feature set F (either F^u, F^r, or F^c). Then, we preprocess both B_t and P_t to obtain the actual training data $\mathcal{D}$. At this point, we apply a given ML algorithm $\mathcal{A}$ (either *RF*, *LR* or *CN*) to such $\mathcal{D}$; the resulting ML model $\mathcal{M}$ (a binary classifier, which we fine tune via grid-search) will be the detection component of the considered ML-PWD.

(3) *Testing*. The last phase is measuring the performance of $\mathcal{M}$. In our case, an ML-PWD must exhibit both a high detection rate and a low false positive rate: indeed, no one is interested in detectors that block legitimate websites due to excessive false alarms. Hence, we preprocess the *inference* partitions B_i and P_i (by considering the proper F) and measure the fpr and tpr—in the absence of adversarial attacks.

[13] Our feature extractor is 'stateless'. Once it receives a sample, the only queries performed are those to some third-party services (e.g., PageRank API, DNS servers), which can be cached to save time. Our extractor, however, does not 'update' a sample: if, e.g., a URL uses a shortening service, then the extractor uses such 'shortened' URL as a basis, and if the HTML changes (due to some automatic script) then such change will not be captured. Such a choice makes sense because ML-PWD must be fast: a user does not want to wait seconds before visiting each website just because a phishing check is made. Moreover, our decision makes our extractor suitable also to ML-PWD that analyze *only* the URL, because the webpage will not be opened in the first place (which is common for phishing email filters) due to the high overhead.

The topmost priority is ensuring that $\mathcal{M}$ analyzing F^c achieve optimal performance: indeed, models using either F^u or F^r are expected to exhibit a lower performance as they are provided with less information; however, using F^u or F^r is expected to yield superior robustness in the presence of evasion attacks. (Our repository includes the best parameter configurations of each ML algorithm.)

Statistical Validation. To provide results that are devoid of experimental bias and also to serve as a reliable benchmark for future research, *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 fpr and tpr. Furthermore, all[14] such $\mathcal{M}$ are also assessed against all our considered attacks (which we will discuss in the next section). Such a large evaluation allows one to perform *statistically validated comparisons* by leveraging well-known techniques [19]. 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.

5.3 Baseline Performance

We report the performance of our ML-PWD (*in the absence* of adversarial attacks) in Table 3. This table shows that the best ML-PWD on both datasets use *RF*. We appreciate that the 'true' baseline ML-PWD (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[15]. 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.

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

Finally, 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.

[14] Overall, for our experiments we develop 900 $\mathcal{M}$ (given by: 2 source datasets * 50 random draws * 3 F * 3 $\mathcal{A}$).
[15] Focusing on the ML-PWD using F^r (which are similar to the real ML-PWD in MLSEC [6]), 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.

6 EVALUATION: ATTACKS (RATIONALE AND IMPLEMENTATION)

We now focus on our considered attacks. We begin by providing an extensive overview (§6.1), and then summarize the workflow for their empirical evaluation (§6.2). Finally, we describe their technical implementation (§6.3)

6.1 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 also affect features beyond the attacker's knowledge.)

- *Cheap Website Attacks* (WA) perfectly align with our threat model (and resemble the use-cases in §4.5). The perturbations are created in the website-space (WsP), realizing either WA^u, WA^r, or WA^c. Specifically for r (and c), we consider two semantically equivalent WsP: "add fake link" for δPhish, 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* ($\widehat{\text{WA}}$), 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., $\widehat{\text{WA}^u}$, $\widehat{\text{WA}^r}$, $\widehat{\text{WA}^c}$), 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 research. Indeed, we observe that all three remaining families ($\widehat{\text{WA}}$, 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 [92] 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 $\mathcal{M}$.

Effectiveness and Affordability. In terms of effectiveness, assuming the same targeted features, WA$<\widehat{\text{WA}}<$PA$\ll$MA (as confirmed by our results in §7.2). 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 $\widehat{\text{WA}}$) 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).
- $\widehat{\text{WA}}$: 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., [22]).
- 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: WA$\gg\widehat{\text{WA}}\gg$PA$>$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).

6.2 Evaluation Workflow

The procedure to assess the adversarial attacks involves three steps.

(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. To meet such condition, we isolate 100 samples from P_i that are detected successfully by the best[16] ML-PWD (typically using F^c) during one of our runs. Such samples are then used as a basis to craft the adversarial samples corresponding to each of the 12 considered types of evasion attacks.
(2) *Perturb*. We apply the perturbations as follows. For WA and $\widehat{\text{WA}}$, 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 (given by 12 attacks, each using 100 samples).
(3) *Evade*. The 1200 adversarial samples are sent to the 9 ML-PWD (for each dataset), and we measure the tpr again.

[16] This ensures that all ML-PWD are assessed against the *same* adversarial samples. We provide such samples in the source-code.

The expected result is that the tpr obtained on the adversarial samples (generated as a result of any of the 12 considered attacks) will be lower than the tpr on the original 100 phishing samples.

6.3 Attacks Implementation

Let us discuss how we implement our perturbations, and provide some insight as to which features are influenced as a result of our attacks. We recall that each attack family presents three variants, depending on which features the attacker is 'consciously' trying to affect. Namely: u, r and c, i.e., features involving the URL, the representation (HTML) or a combination thereof. All attacks are created by manipulating (phishing) samples taken from P_i. In particular, during our first trial, we isolate 100 samples from P_i that are correctly detected by the best ML-PWD: such samples are then used as the basis for all their adversarial variants (to ensure consistency). We will denote any of such samples as p.

We start by describing MA which are the easiest to implement. Then, we describe WA and $\widehat{\text{WA}}$. Finally, we describe PA, which are the most complex to implement because they must consider several implications (e.g., inter-feature dependencies). (Our repository includes the exact implementation of MA and PA, and also all the pre-processed variants of the samples generated via WA and $\widehat{\text{WA}}$.)

6.3.1 ***ML-space attacks.*** The attacks (i.e., MA) are the easiest to implement. Indeed, we simply follow the same procedure as done by most prior works (e.g., [33, 59]) that directly manipulate the feature representation F_p of a sample p right before it is analyzed by the ML-PWD. We do this without taking into account any inter-dependency between features and/or any physical property that the actual webpage must preserve: this is compliant with our assumption that the attacker has access to the ML-space. Specifically, for each MA we apply the following MsP:

- MA^u: The attacker targets URL-related features. Hence, we manipulate F_p by setting features based on F^u equal to -1, which denotes a value that is more likely associated with a benign sample. In particular, we set to -1 the features in Table 1 with the following numbers: (1-17,19-21,27,30-35)
- MA^r: Same as above, but the targeted features are within F^r. Hence, we set to -1 the features in Table 1 with the following numbers: (36-40,42-52,54-57)
- MA^c: We set to -1 all features involved in MA^u and MA^r.

We remark that the attacker is not aware of the feature importance (because it would require knowledge of $\mathcal{M}$). Hence, although some manipulations will likely 'move' F_p towards a benign webpage, it is not guaranteed that $\mathcal{M}$ will actually classify such F_p as benign: if the manipulated features are not important, then even MsP may have no effect (and such phenomenon *does* happen in our evaluation, e.g., the ML-PWD using *RF* with F^c on Zenodo against MA^r).

Of course, we could set *all* features to -1 (e.g., all F^u and F^r). Doing this, however, would obviously result in a perfect misclassification (and hence not interesting to show). Moreover, it would not be sensible *even for the attacker*. Indeed, MA assume no knowledge of $\mathcal{M}$ and of $\mathcal{D}$, meaning that an attacker may suspect the existence of a honeypot [83]. For instance, $\mathcal{D}$ may contain some samples with all features set to -1 (i.e., benign) that are labelled as phishing—for the sole purpose of defeating similar attacks in the ML-space. Hence, it is realistic to assume that even an attacker capable of MA would not exaggerate with their perturbations.

6.3.2 ***Website attacks.*** We recall that we performed two families of attacks in the website-space: WA and $\widehat{\text{WA}}$. The peculiarity of both of these attacks (both relying on WsP) is that the attacker does not have access to the ML-PWD. Hence, they are not able to manipulate F_p, and they are not even able to *observe* F_p.

- WA: These attacks resemble the pragmatic example (discussed in §4.5). Let us elaborate:
  - WA^u: We set the URL to a random string starting with "www.bit.ly/", followed by 7 randomly chosen characters (which what this popular URL shortener does).

  - WA^r: For δPhish, we change the HTML by adding 50 invisible internal links (i.e., having the same root domain of the website);[17] for Zenodo, we wrap all links within an "onclick", i.e., we change: *<a href='link'>* into *<a onclick="this.href='link'">*.[18]
  - WA^c: We do both of the above for each dataset.
- $\widehat{\text{WA}}$: These attacks envision an attacker that knows how the feature extractor within the ML-PWD operates (see §5.1.3). Such knowledge can be acquired, e.g., if the attacker has (or is) an insider that provided them with such intelligence. However, the attacker is still confined in the website-space, and hence can only apply WsP (to generate $\bar{p}$). For a meaningful comparison, we assume an attacker who is aware of how the features targeted in WA are "extracted" within the ML-PWD. Hence, we craft each $\widehat{\text{WA}}$ as follows:
  - $\widehat{\text{WA}^u}$: The attacker, having knowledge of the extractor, knows that by using an URL shortener they will affect all features related to the URL (i.e., F^u); furthermore, they know the threshold (53) that makes an URL to be considered as 'benign'. Such length is well above that of an URL generated via any shortening service. As such, these attacks are an exact replica as $\widehat{\text{WA}^u}$ (the only difference is that the attacker of $\widehat{\text{WA}^u}$ is more confident than the one in WA^u).
  - $\widehat{\text{WA}^r}$: The attacker manipulates the HTML in the same was as in WA^c. However, the attacker also knows the threshold (0.15) of internal-to-external links that yields a benign value of the *HTML_objectRatio* feature. Hence, the WsP manipulate the HTML of each p by introducing as many links (or wrappings) as necessary to meet such threshold.
  - $\widehat{\text{WA}^c}$: The attacker does both of the above.

  We stress that the attacker cannot observe $F_{\bar{p}}$. Indeed, doing this would require the attacker to completely replicate the feature extractor, which is costly, and may not even be possible (some third-party services may require subscriptions to be used). As such, the attacker is aware of how to craft WsP that are more likely noticed by the ML-PWD, but evasion is not guaranteed.

6.3.3 ***Preprocessing attacks.*** These attacks are the hardest to realize *from a research perspective* and *in a fair way*.

Challenges. The underlying principle of PsP (the backbone of PA) is affecting the preprocessing space of the ML-PWD. Technically, since we are the developers of our own feature-extractor (i.e., the component of the ML-PWD devoted to data preprocessing), we could simply directly manipulate our own extractor, i.e., by introducing a 'backdoor'. However, doing this would prevent a fair generalization of our results: for instance, it is possible to develop another feature extractor, having the same functionality but whose operations are executed in a different order. Hence, to ensure a more fair evaluation, we apply the perturbations *at the end* of the preprocessing phase, but we do so by anticipating how a perturbation in the website-space (a WsP) could affect the preprocessing-space, thereby turning a WsP into a "physically realizable" PsP. To this purpose, we *assume the viewpoint of an attacker*. For instance, we ask ourselves: "if an attacker wants to affect URL features by using an URL shortener, how would the feature extractor react?".

Scenario. In PA the attacker *knows* and *can interfere* (through PsP) with the feature extraction process of the targeted ML-PWD. However, the attacker is *not* aware of what happens next: the ML-space and the output-space are both inaccessible by the attacker (from both a *read* and *write* perspective). Hence, once the PsP has been applied and $\overline{F_p}$ is generated, the attacker cannot influence $\overline{F_p}$ any longer. For each PA we do the following:

[17] The exact string we inject is: "* can not see*", which is the second string shown in our pragmatic example (§4.5).
[18] This WsP, if applied to textual link, would remove the underline of such a link, therefore being visible to a user; however, it is possible to make it invisible by editing the CSS properties. Our feature extractor is agnostic of such properties, so we do not do this: the results would be equivalent.

- PA^u: we anticipate an attack that targets URL features, and specifically *URL_length*, by using an URL shortener. Hence, we can foresee that operations (in the website-space) can lead to alterations of *all* the features involved with the URL (i.e., F^u). For instance, doing this would make weird characters (if present) disappear from the URL. However, doing this would induce alterations also to F^r. For instance, some objects originally considered to be 'internal' would become 'external'. Hence, we implement PA^u by setting the following features (from Table 1) to -1: (1-3,5,6,8,10-16,22,23,25,26,28-30), whereas the following features are set to +1: (4,27,36-38,41,44,48,52,54,56).
- PA^r: we anticipate an attack that targets features related to the representation of a website—in our case the HTML, and specifically the *HTML_objectRatio* feature. We foresee that an attacker can interfere with such a feature in many ways, for instance by removing links, adding new ones, or changing those already contained in the webpage. All such changes will affect many features, such as the *HTML_freqDom*: because populating the HTML with (fake) internal links would change the 'frequent domains' included in the HTML. Such changes can also affect the links in the footer of the webpage (*HTML_nullLnkFooter*); or the anchors (*HTML_anchors*); but also others. We implement PA^r by setting the following features (from Table 1) to -1: (36–38,41,51,54,56,57); whereas we set (39,40) to 1 and 46 to 0.
- PA^c: they are a combination of the two above. We expect the attacker to use a URL shortener, and also interfere with the *HTML_objectRatio*. However, we cannot simply set the features to the same values as PA^r and PA^u, because one of the two will prevail. In our case, shortening the URL will be 'stronger', because the URL will change (to that of the URL shortener) and hence the internal objects will become 'external'. Hence, we implement PA^c by setting the following features (from Table 1) to -1: (1-3,5,6,8,10-16,22,23,25,26,28-30), whereas the following features are set to +1: (4,27,36-38,41,44,48,52,54,56).

We remark that our PsP may not yield an $\overline{F_{\bar{p}}}$ that is a perfect match with a $F_{\bar{p}}$ generated via WsP (i.e., those of $\widehat{\text{WA}}$). Indeed, some inconsistencies may be present—likely due to 'inaccurate' anticipations from our (i.e., the attacker's) side. Such inconsistencies are sensible. An attacker with access to the preprocessing-space can theoretically *replicate* the entire feature extractor, and use it to exactly pinpoint how to generate PsP that are an exact match with WsP (i.e., $\overline{F_{\bar{p}}}$=$F_{\bar{p}}$). However, doing this would be *very expensive*. Furthermore, it would *defeat the purpose* of using PsP: the attacker does not want that $\overline{F_{\bar{p}}}$=$F_{\bar{p}}$, rather, they want a PsP that is 'stronger'; otherwise, why use PsP in the first place?

7 RESULTS AND DISCUSSION

We present the results of our evaluation. We aim at answering two questions:

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

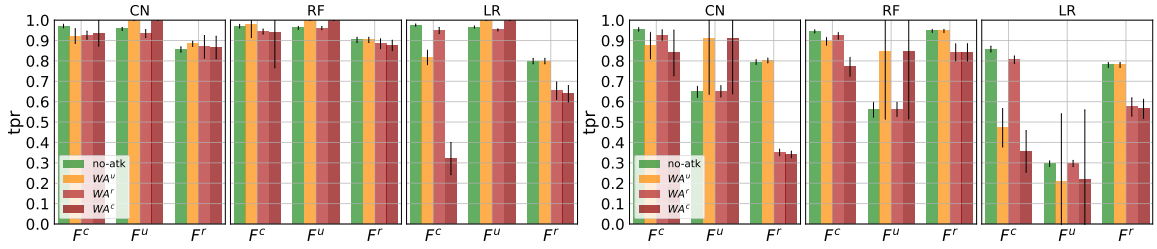
We also perform a proof-of-concept experiment on a competition-grade ML-PWD (§7.4). Finally, we discuss our evaluation and potential for future work in §7.3. We report our full 'benchmark' results in Appendix A.

7.1 Effectiveness of the most likely attacks (WA)

Let us focus our attention on the most likely attacks. We report in Figs. 7 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 the ML-PWD using F^c (leftmost group of bars in each plot), since they are the ML-PWD with the best performance in the absence of attacks (cf. Table 3).

- On δPhish (Fig. 7b), 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,



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

Fig. 7. **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.

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. 7a). 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.

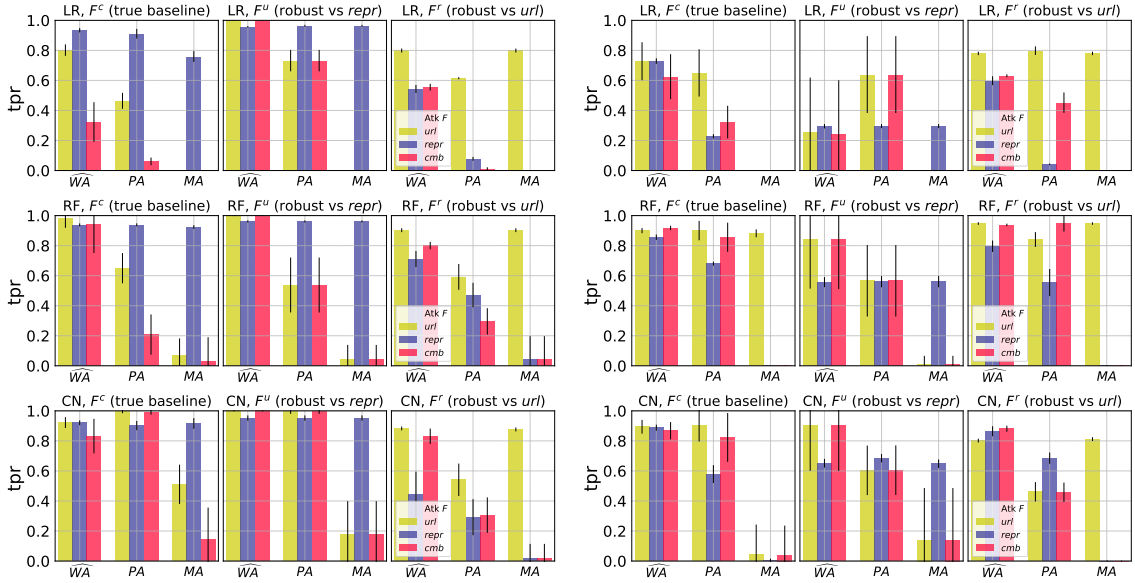
Robust Baselines (F^u, F^r). The robust baselines are, in general, reliable against WA. The ML-PWD using F^u counter WA^r (and viceversa), because the tpr is exactly the same as the original one. Notably, however, ML-PWD using F^r (similar to the ML-PWD of[19] MLSEC [6]) are affected by WA^r: the *LR* 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:** Our realistic attacks in the website-space (WA^c) can evade 5 (out of 6) ML-PWD. The performance degradation is small, but statistically significant. Due to their cheap cost, WA^c are a threat to state-of-the-art ML-PWD.

7.2 Comparing the evasion-space ($\widehat{\text{WA}}$, 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. 8.

[19]We also successfully attacked the competition-grade ML-PWD of 7.4 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 of the 9 advanced attacks (i.e., $\widehat{\text{WA}}$, PA, MA) across the 50 trials. Colors denote the targeted features (u, r, c).

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

Fig. 8. **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).

The 'true' baselines (using F^c, i.e., the leftmost plots in Fig. 8) 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 $\widehat{\text{WA}}$ 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 $\widehat{\text{WA}}$. What is even more surprising is that such *CN* significantly outperforms the *RF* against PA, but *also* against MA. Such finding could inspire deployment of ML-PWD using deep learning on Zenodo—despite being inferior to *RF* in the no-atk (Table 3) and against WA^c (§7.1).

We note that $\widehat{\text{WA}^u}$ perfectly match WA^u, which makes sense as they involve exactly the same WsP (cf. §6). 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 δPhish) we observe that the effectiveness of $\widehat{\text{WA}}$ and PA tend to be similar. Such a 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 [92].

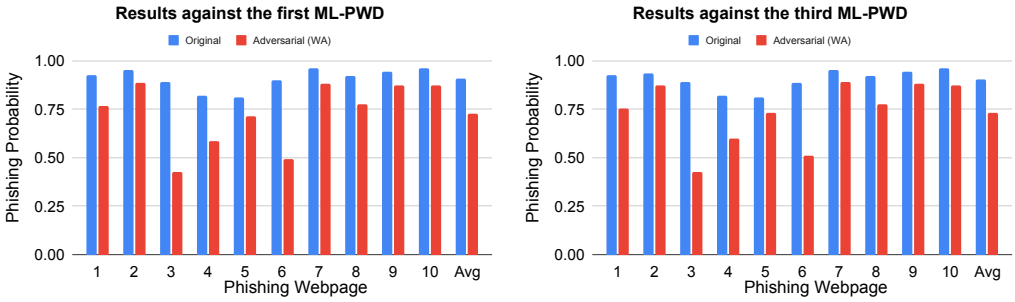
Let us compare our attacks with those considered by δPhish creators. Specifically, the attacks in [33] manipulate increasingly higher amounts of features (up to 10), and all ultimately evade target ML-PWD (which analyzes the HTML). Such a finding is confirmed by our results on the ML-PWD analyzing F^r on δPhish 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.*

7.3 Discussion

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 [24]). 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 (§7.1). 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 [8], and in such big numbers even a 1% difference can be the separation between a compromised and secure system [18]. 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., [33, 61]) 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., [36]): 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.

Future Work. 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., [56]). Additional defenses can also be considered (e.g., adversarial training [73, 94]). For instance, we did not consider ML-PWD that analyze the visual representation of a webpage (e.g., [9, 63]): such attacks would resemble those conducted in computer vision, which are well-known to be effective (e.g., [76, 93]). 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.



(a) Impact of WA^r on the *first* ML-PWD.

(b) Impact of WA^r on the *third* ML-PWD.

Fig. 9. **MLSEC results.** Effectiveness of the most likely attacks (WA^r on δPhish) against the ML-PWD of MLSEC [6].

7.4 Proof-of-concept: attacks against a competition-grade ML-PWD

To further prove the impact of our 'cheap' attacks (i.e., WA), we tested them on a real ML-PWD that is used in a well-known Machine Learning Security Evasion Competition (MLSEC [6]). Such competition is held yearly, and is organized by leading tech-companies that provide cybersecurity services reliant on ML methods. The 2022 edition of MLSEC envisions a challenge in which participants are asked to *evade* ML-PWD. We took this opportunity to assess whether our attacks had any impact against such 'competition-grade' ML-PWD. (**Short story**: they do. A demonstrative video can be found in the homepage of our website—which also includes the source-code).

7.4.1 ***Challenge.*** Participants of the phishing evasion challenge are given 10 'phishing' webpages, which are provided in their raw HTML form. The purpose of the challenge is to manipulate such webpages so that (i) they render exactly as the originals, and (ii) they evade an ML-PWD. Specifically, the organizers provide 8 different ML-PWD, which the participants can use as a black-box: by sending an input (i.e., the HTML of a phishing webpage), they are given an output (i.e., the probability that such webpage is malicious—according to the specific ML-PWD). Such ML-PWDs only analyze the HTML of the webpage (which must render exactly as the original). Put simply: the objective of the challenge is to tweak the HTML of the 10 webpages with imperceptible modifications that decrease the confidence of the 8 ML-PWD.

7.4.2 ***Method.*** Of course, the setting described above perfectly describes the black-box scenarios envisioned in adversarial ML papers: query the detector, and use the response as a guide to craft a more evasive phishing webpage. Our primary attacks (WA), however, are query-less. Because we are aware that the target ML-PWD analyzes the HTML (recall that this is an assumption of our threat model), we then craft our 'adversarial' phishing webpages by using exactly the same WA^r used in our paper for δPhish: we add 50 invisible internal links. We apply these WsP to all the 10 webpages *provided by the organizers of the challenge*, and then test whether they had any impact to the *real* ML-PWD involved in the challenge.

7.4.3 ***Results.*** By taking into account *all* webpages against *all* ML-PWD, our attacks induced a drop of 3.4% in the confidence of the ML-PWD, indicating that our WsP had some effect. However, while some ML-PWD were not very affected, others incurred a significant drop. Specifically, we focus our attention on the first and third ML-PWD provided by the organizers of MLSEC. The results of our proof-of-concept experiments are shown in Figs. 9. These graphs show phishing probability (y-axis) given as output by the corresponding ML-PWD for each of the 10 webpages of the challenge (x-axis). We report two bars: the blue bar are the results of the original webpages, whereas the red bars are the results after applying our WsP.

7.4.4 ***Analysis.*** These two detectors were significantly less certain after our WsP, with an average confidence drop of 17.5%. We observe that in most cases, the confidences were still above 0.5 (i.e., the webpages would still be classified as 'phishing'). A more detailed look, however, reveals that **these detectors were completely fooled by some webpages** (i.e., their confidence dropped to below 0.5). We report:

- Page #3: from 0.90 down to 0.43 for the 1st and 3rd detectors.
- Page #6: from 0.90 down to 0.49 for the 1st detector.

We also attempted the same WA^r by changing the number of fake links, and also by considering a different string[20]. When applied to, e.g., webpage #3, adding 280 links dropped the confidence to below 0.2; whereas adding a slightly different string (the first one shown in our pragmatic example in §4.5) 280 times, the confidence dropped to 0.2 for

[20]We also considered the 'wrapping' WsP for Zenodo: the effects were negligible—probably because these ML-PWD factored such links into their 'count' (i.e., the attacker made a wrong guess). See Appendix B

the first and third detector, and to 0.49 for the seventh detector. The seventh detector was also fooled by adding such alternative string 50 times to webpage #4, causing a confidence of 0.46 (down from 0.68). The source-code is available in our repository, and the experiments are entirely reproducible. Interestingly, these results align with those shown in our primary evaluation: our *query-less* WA attacks cannot bypass any ML-PWD, but in some cases *they can induce a miss-classification*.

8 ADDITIONAL EXPERIMENTS: SAME-SPACE AND MIXED-SPACE

We *expand* the evaluation carried out for our ACSAC'22 paper [17] with additional experiments. Our goal is twofold:

- assessing other types of perturbations (either WsP, PsP or MsP) *in the same space*;
- consider a "stronger" attacker that applies *multiple* perturbations also in *different* spaces (cf. §4.4).

We first describe (§8.1) and empirically evaluate (§8.2) the attacks entailing perturbations in the "same-space". Then, we describe (§8.3) and evaluate (§8.4) the "multi-space" attacks.

8.1 Same-space Attacks: Description

In this section, we elaborate on new attacks in the same evasion space involving our WsP, PsP, and MsP. Building upon the attacks considered in the main evaluation (§6), we introduce additional perturbations. The motivation behind this extension is to present a more comprehensive range of use cases—all of which are likely to happen, since they are well within the attacker's capabilities (who will never have complete knowledge of the target PWD). Therefore, we explore novel perturbations of the HTML (§8.1.1) and URL (§8.1.2), as well as introduce new variations of MsP, PsP, and WsP. Altogether, the details of the new specific attacks are provided in Table 4.

8.1.1 ***HTML.*** As we know (§2), the HTML reflects the visual appearance of a webpage—therefore, changes to the HTML can lead to differences in the way the webpage is presented to its users.[21] Some of them may be noticed by users (e.g., alterations of the background), while others may not change the appearance at all (e.g., the hidden links considered in our pragmatic use-case §4.5). Here, we consider a wide-array of HTML-related perturbations, and scrutinize which are more likely to evade the detection of PWD. Practically, we propose a total of 37 new HTML-related perturbations—of which, 24 are WsP (i.e., new WA^r), which can be divided into the three following categories:

1) *iWsP* (invisible WsP), which denote perturbations that are inserted into the webpages but remain invisible to users. This means that the webpage appears unchanged before and after the perturbation insertion.
2) *eWsP* (elusive WsP), which introduces slight changes to the appearance of the webpage. While these changes may require some effort to be noticed by users, they are still discernible upon careful observation.
3) *rWsP* (recognizable WsP), which result in changes that are clearly visible to users. These modifications have a more pronounced impact on the webpage's appearance, making them readily noticeable.

The remaining 13 HTML-related perturbations are PsP and MsP (i.e., new PA^r and MA^r). Both of which require write-access to the ML-PWD. PA^r can bypass some of the checks of ML-PWD. Moreover, in MA^r, attackers may solely focus on evading ML-PWD: as a result, some MA^r might violate the fundamental rules of HTML.

8.1.2 ***URL.*** Domain and path are two essential components of URL, and most of our URL features in Table 1 are extracted from them. In this section, we implemented 6 types of perturbations that specifically target the URL. The low-level implementation of these attacks, referred to as $u\mathrm{WA}^u$, are discussed in Table 4. We do not consider URL-related perturbations that affect other spaces (i.e., PsP or MsP).

[21] We recall that our threat model does not assume that the perturbations are "imperceptible" to humans. This is because, in a real scenario, phishing is effective because humans are distracted. Hence, even if the webpage changes, the phishing attack can still be successful.

Table 4. **Same-space attacks details.** We show the low-level implementation of the (new) attacks entailing multiple perturbations in the same evasion space. The last group focuses on perturbations affecting the URL (i.e., u); the others focus on the HTML (i.e., r).

Category	Perturbation	Description
iWA^r	*addInLnk*	insert internal links
	replOnc	$\langle a\ href = `link\ '\rangle$ to $\langle a\ onclick = ``this.href = `link\ ''\rangle$
	delHidIt	delete hidden items from HTML
	addHidP	add hidden large page
	replJS	replace $\langle a\ href = `\#'\rangle$ with $\langle a\ href = `javascript : void(0)'\rangle$
	replRet	replace '\n' with whitespace
	htEsc	escape the whole body content, and write "document.write(unescape(' '))" to HTML
	htEncd	encode HTML with base64
	replPass	replace $\langle input\ type = `password'\rangle$ with $\langle input\ type = `text'\ \rangle$
	replOnfoc	replace $\langle input\ type = `password'/`email'\rangle$ with $\langle input\ onfocus = ``this.type = `password'/`email''\rangle$
	addSusLnk	add suspicious links $\langle a\rangle$, e.g., $\langle a\ href = `\#skip'\rangle$
eWA^r	*addImgBot*	insert 20 small local images to the webpage bottom
	modFntTyp	modify the font type italic
	addTps	randomly insert few typos into HTML text
	modCpy	modify copyright
	addIcn	add local icon
	delSusLnk	delete suspicious links
	delSusFrm	delete suspicious form (i.e., with empty or external 'action' links)
	modTtl	randomly modify the title
	delCpy	delete copyright information from HTML
rWA^r	*modBgimg*	change the background image
	modBgClr	randomly change the background color
	modFntClr	randomly modify the font color
	modFntSiz	modify the body font size to 0
PA^r	*delTxt*	delete all text from HTML
	delFrm	remove forms
	delSpn	remove all span
	delTtl	remove title
	addLngTxt	add long visible text to HTML
	delFtr	remove footer
	replSusFtrLnk	replace suspicious links of footer with internal links
MA^r	*brTg*	break the tag $\langle html\rangle$
	delHt	remove the whole html
	delHd	delete the whole $\langle head\rangle$ except style
	delBdy	delete the whole $\langle body\rangle$
	brTgs	break tags
	hmg	replace characters with homographic letters
uWA^u	*replChar*	replace the characters in the domain with visually similar characters
	sepWrd	randomly insert whitespaces within the domain to separate the individual word
	delChar	delete one character from the domain
	swpChar	randomly swap two adjacent characters in domain
	addChar	randomly insert an additional character into the domain
	atkPth	also conducted operations of swap, delete, or insert randomly within the path of the URL

8.2 Same-space Attacks: Evaluation

We now assess the impact of the abovementioned perturbations. For HTML perturbations (§8.2.1), we consider the effects both on the ML-PWD we developed by using the δPhish and Zenodo datasets, as well as by those provided by MLSEC (we carried out these experiments in December 2022, when the MLSEC API was still open for research

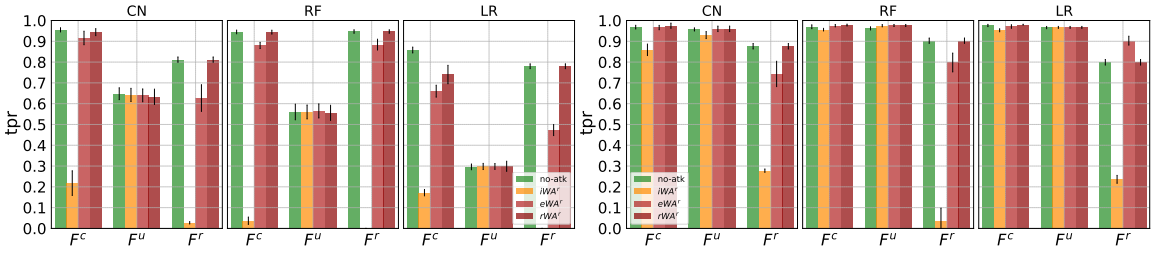
purposes). For the URL perturbations (§8.2.2) we consider only the ML-PWD trained on δPhish and Zenodo because those provided by MLSEC do not consider the URL in their analyses.

8.2.1 ***Impact of HTML perturbations.*** We begin by considering δPhish, Zenodo, and then focus on MLSEC.

δPhish and Zenodo. In Figs. 10, we present the *tpr* achieved by ML-PWD trained on δPhish and Zenodo. We evaluate the performance of these ML-PWD against iWA^r, eWA^r and rWA^r (represented by yellow and red bars)[22]. To provide a comparison, we also include the *tpr* achieved by the same ML-PWD on the original set of samples, depicted by the leftmost green bar labelled as "no-atk". These results aim to address two key questions:

- Will different WsP have different impacts on ML-PWD and how?
- What kind of WA is more likely to evade the ML-PWD trained on δPhish and Zenodo?

As shown in Fig. 10a, the iWA^r perturbation emerges as the most impactful attack, leading to a significant reduction (reduced by 0.68–0.95) in the *tpr* of F^r- and F^c-based ML-PWD trained on δPhish. Specifically, the *tpr* of RF-PWD trained on F^c drops from 0.945 to 0.037, and the *tpr* of RF-PWD trained on F^r decreases from 0.947 to 0. In comparison, the influence of eWA^r and rWA^r is relatively smaller. However, eWA^r still causes a notable drop in the *tpr* of F^r-based LR-PWD, reducing it from 0.78 to 0.47. On the other hand, rWsP has minimal impact on PWD (only F^c-based LR-PWD's *tpr* decreased by 0.12). A similar trend is observed in Fig. 10b for the influence on Zenodo, where iWA^r remains the most effective attack. Additionally, eWA^r affects ML-PWD to a greater extent (except for F^r-based LR-PWD) compared to rWA^r. These findings demonstrate that iWA^r poses the greatest challenge to ML-PWD of δPhish and Zenodo, significantly reducing their detection performance. eWA^r also has a notable impact, while rWA^r has a relatively minor effect on most ML-PWD (except for the ML-PWD using LR to analyze F^r).



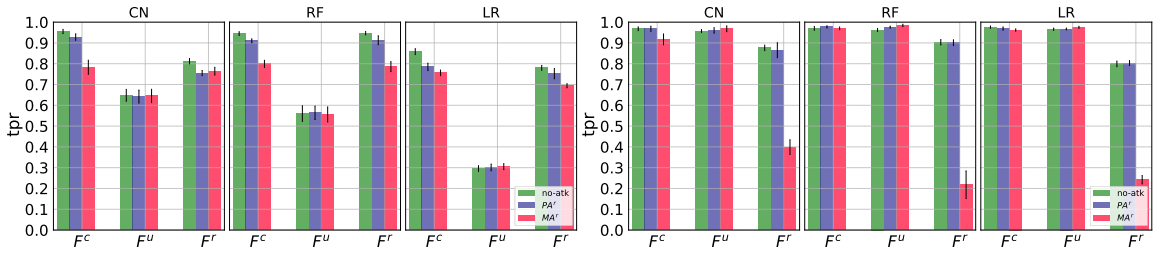
(a) Impact of iWA^r, eWA^r and rWA^r) δPhish. (b) Impact of iWA^r, eWA^r and rWA^r) on Zenodo.

Fig. 10. **Effectiveness of the most likely (new) 'same-space' attacks** WA^r. The three plots in each subfigure represent the algorithm used by a specific ML-PWD. Each plot has bars divided in three groups, each bar denotes 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.

Figs. 11 represents the impact of new PA^r and MA^r on ML-PWD trained on δPhish and Zenodo. In this context, PA^r refers to *delFrm* (i.e., remove forms from the webpage), while MA^r denotes applying perturbation *hmg* to HTML (i.e., inserting typos to the HTML, both tags and text). Comparing with the *tpr* of 'no-atk', it is evident that both PA^r and MA^r have negative impact on the *tpr* of ML-PWD trained on δPhish and *Zenodo*. Specifally, PA^r reduced the *tpr* of all F^c- and F^r-based ML-PWD on δPhish, with small decreases ranging from 0.01 to 0.08. On the other hand, MA^r had a more pronounced effect compared to PA^r, successfully reducing the *tpr* of F^r-based ML-PWD by 0.1–0.17. Nevertheless, WA^r is still the most effective attack compared with them.

MLSEC. We have summarized the impact of the new HTML attacks on MLSEC in Table 5. These attacks are the same HTML attacks used in δPhish and Zenodo. Our findings reveal several interesting phenomena in the evaluation:

[22]Our figures only present the most effective WsP, i.e., iWA^r denotes *addHidP*, eWA^r stands for *addImgBot*, and rWA^r represent *modFntClr*.



(a) Impact of new PA^r and MA^r on δPhish. (b) Impact of new PA^r and MA^r on Zenodo.

Fig. 11. **Effectiveness of new attacks PA^r and MA^r.** The three plots in each subfigure represent the algorithm used by a specific ML-PWD. Each plot has bars divided into three groups, each denoting a specific F used by the ML-PWD. The green bars show the tpr on the original samples, the blue bars represent tpr against PA^r and the red bars in the rightmost show the tpr against MA^r.

- Among the attacks evaluated, iWA^r emerges as the most potent attack, significantly degrading the performance of PWD of MLSEC. The confidence of models $m0$ and $m2$ drop from nearly 0.9 to 0.02, indicating a stark decrease in their ability to accurately detect malicious webpages. However, it is worth noting that other attacks also have a substantial impact on degrading the detection capability of PWD. For instance, PA^r reduce the confidence of $m2$ from 0.9 to 0.61, while MA^r results in a decrease of 0.76 in the confidence of $m6$.
- Comparing to eWA^r and rWA^r, iWA^r has a greater influence on $m0$–$m3$, leading to a decrease in their confidence by 0.35–0.89. However, for PWD $m4$–$m7$, iWA^r does not decrease their confidence but slightly increases them by 0.01. On the other hand, rWA^r reduces their confidence by 0.1 (from nearly 0.8 to 0.7), while eWA^r results in a confidence reduction of 0.2 for PWD $m4$ and $m6$. This phenomenon can be considered reasonable since PWD employed in MLSEC are black-box models which may consist of multiple types of PWD. It implies that the impact of perturbations may vary depending on the specific model characteristics and vulnerabilities. Hence, it is important to note that the goal of this study is not to propose a generalized perturbation set that works for all PWD, but rather to investigate the impact and effectiveness of cheap perturbations on PWD in practice.
- It is observed that rWA^r has a more widespread impact as it influences all seven PWD on MLSEC, resulting in a reduction of confidence by 0.1 across the board.
- Both PA^r and MA^r are effective attacks that successfully evade the detection of PWD in MLSEC. In particular, MA^r proves to be a potent attack, as it evades five (out of eight) PWD, causing their confidence score to drop below 0.5. Additionally, the confidence scores of seven PWD decrease to approximately 0.65 from initial values of around 0.85. These findings highlight the impact of PA^r and MA^r on the performance of PWD on MLSEC.

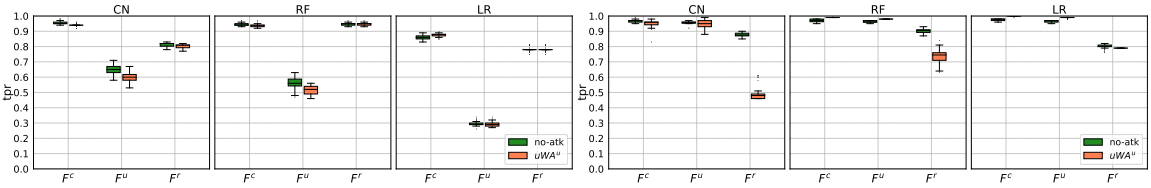
Takeaway: Applying iWsP does not change the webpage's appearance but yields highly evasive samples. In contrast, the application of rWsP results in obvious changes to the webpage's appearance but it has a relatively minor impact on the detection. MA^r has a more pronounced effect compared to PA^r. Intriguingly, some WA^r are more evasive than MA^r and PA^r.

8.2.2 ***Impact of URL perturbations.*** The impact of $u\mathrm{WA}^u$ is illustrated in Figs. 12. Fig. 12a reveal the changes when performing *atkPth* on ML-PWD trained on δPhish. Green boxes represent the tpr of 'no-atk' (i.e., baseline), while the orange boxes indicate the impact of $u\mathrm{WA}^u$. Comparing the medians of each box plot, the median line of orange boxes is lower than Green boxes for F^u-based ML-PWD, indicating that $u\mathrm{WA}^u$ can degrade ML-PWD's tpr. In contrast, this type of $u\mathrm{WA}^u$ does not decrease tpr of F^u-based CN-PWD trained on Zenodo (as shown in Table 21 in Appendix C).

Table 5. New attack's impact on MLSEC (HTML perturbations)

$\mathcal{A}$	no-atk	iWA^r	eWA^r	rWA^r	PA^r	MA^r
$m0$	0.91±0.052	0.02±0.011	0.65±0.185	0.81±0.116	0.91±0.052	0.90±0.062
$m1$	0.87±0.071	0.52±0.161	0.87±0.085	0.78±0.100	0.67±0.262	0.31±0.051
$m2$	0.90±0.051	0.02±0.011	0.65±0.185	0.85±0.087	0.61±0.390	0.88±0.096
$m3$	0.88±0.070	0.51±0.172	0.87±0.079	0.81±0.091	0.66±0.271	0.26±0.080
$m4$	0.82±0.106	0.83±0.123	0.64±0.199	0.73±0.112	0.57±0.372	0.80±0.121
$m5$	0.81±0.120	0.82±0.136	0.85±0.107	0.70±0.103	0.64±0.280	0.39±0.166
$m6$	0.83±0.108	0.84±0.116	0.64±0.198	0.73±0.111	0.56±0.373	0.07±0.076
$m7$	0.82±0.121	0.83±0.127	0.85±0.106	0.70±0.097	0.64±0.279	0.36±0.129

However, it is significantly reduces the performance of F^r-based ML-PWD. This is because some HTML features require extracting information from both URL and HTML (e.g., *HTML_URLBrand*: which checks (in the HTML) if the webpage title includes the brand name that appeared in the URL). Therefore, either URL perturbations or HTML perturbations can possibly affect the F^r-based ML-PWD. Furthermore, as shown in Fig. 12b, another $u\text{WA}^u$, *sepWrd*, also clearly decreases the tpr of F^r-based ML-PWD. Simply put, $u\text{WA}^u$ affect the ML-PWD's performance.



(a) Impact of new $u\text{WA}^u$ on ML-PWD trained on δPhish (b) Impact of new $u\text{WA}^u$ on ML-PWD trained on Zenodo

Fig. 12. **Effectiveness of new attacks $u\text{WA}^u$.** The three plots in each subfigure represent the algorithm used by a specific ML-PWD. Each plot has box divided into three groups, each denoting a specific F used by the ML-PWD. The green box shows the tpr on the original samples, while the orange box show the tpr against $u\text{WA}^u$.

8.3 Multi-space Attacks: Description

Insofar, we have always considered perturbations applied in a single space. However, as mentioned in §4.4, an attacker who can apply PsP or MsP (which require write-access to the ML-PWD) can also apply WsP (which only require access to the phishing webpage—which the attacker owns). These "multi-space" attacks are worth considering because they are trivial to implement for an attacker—assuming that such an attacker can already apply PsP and/or MsP (we recall that, from a cost viewpoint, WsP≪PsP<MsP). Therefore, we introduce 66 types of multi-space attacks (the complete details are in Appendix C). These attacks span across all the defined evasion spaces (§3): Website space, Preprocessing space, and Machine Learning space. In particular, we consider "accessible" attacks (which combine WA and PA), as well as stronger ones (which entail MA and PA). We also consider "double" attacks, entailing multiple perturbations *in the same space* (e.g., WsP+WsP). We expect that multi-space attacks, which exploit vulnerabilities/weaknesses in different stages of the detection process, lead to more evasive samples (at least w.r.t. corresponding single-space attacks).

8.4 Multi-space Attacks: Evaluation

We evaluate the evasion capabilities of our new mixed-space attacks on the ML-PWD trained on the δPhish, Zenodo, as well as those provided by MLSEC.[23] We begin by considering attacks entailing two perturbations *in the same space*, i.e., PsP+PsP (§8.4.1) and WsP+WsP (§8.4.2); then, we consider attacks entailing two perturbations *in different spaces*, i.e., PsP+WsP (§8.4.3) and PsP+MsP (§8.4.4).

8.4.1 ***Double-PsP.***

- **δPhish** and **Zenodo**. Table 16 and Table 26 demonstrate the impact of 8 kinds of PA^r+PA^r on ML-PWD trained on δPhish and Zenodo. Even though not all of them significantly impact the PWD of δPhish. While not all combinations significantly affect the PWD, there are notable influences observed. For instance, when the combination attack occurs (specifically, the perturbation *delSpn_delTtl*), the tpr of LR-PWD based on F^c and F^r drops by 0.1 and 0.16, respectively. Additionally, the tpr of F^r-based LR-PWD down from 0.8 to 0.58, and CN-PWD's drops from 0.86 to 0.64 after being subjected to PA^r+PA^r. In contrast, F^u-based PWD is not affected, and most of F^c-based PWD remain unchanged. That is because our PA^r+PA^r combinations specifically target HTML, and F^u is the core component when crafting the F^c-based ML-PWD.
- **MLSEC**. In the case of MLSEC's PWD, Table 32 indicates that all cheap PA^r+PA^r combination attacks proposed can decrease the performance of PWD, resulting in the confidence score dropped by [0.01–0.32].

8.4.2 ***Double-WsP.***

- **δPhish** and **Zenodo**. As shown in Table 18, the combination attack WA^r+WA^r did not reduce the tpr of ML-PWD trained on δPhish. In fact, in some cases, the tpr increased to 1.0, such as the tpr of F^r-based CN-PWD increased from 0.79 to 1. Similarly, '*replOnfoc_replRet*' did not affect the ML-PWD of Zenodo, as shown in Table 24). However, it is importance to note that under the influence of '*htEsc_replRet*', the tpr of F^r-based LR-PWD reduced to 0.55 from 0.8. Moreover, '*htEncd_replRet*' reduced tpr of F^r-based CN-, LR- and RF-PWD to 0. These findings suggest that while some combinations of WA^r+WA^r attacks may not result in a significant reduction in the tpr of ML-PWD, specific combinations can still have an impact on the detection performance, leading to a decrease in the tpr. The effectiveness and impact of these combinations may vary depending on the specific ML-PWD and the nature of the attacks employed.
- **MLSEC**. On the contrary, WA^r+WA^r proves to be a powerful weapon for disrupting PWD of MLSEC. As indicated in Table 31, the combination attack '*replOnfoc_replRet*' defeated all detectors, leading to a significant decrease in their confidence scores by [0.12–0.58]. Moreover, four PWD have their confidence scores reduced below 0.5, indicating a successful evasion. Furthermore, the attack '*htEsc_replRet*' evades four detectors, resulting in a substantial reduction in their confidence scores to 0.03 or near 0.15. Additionally, the attack '*htEncd_replRet*' successfully bypasses four detectors and notably decreases the confidence score of model $m0$ from 0.91 to 0.08. These findings demonstrate the effectiveness and potency of WA^r+WA^r combination attacks in evading detection and undermining the confidence of PWD in MLSEC. The combination of multiple WA^r proves to be highly disruptive, highlighting the need for robust defense mechanisms against such attacks.

> **Takeaway:** The simplest "Double-WsP" can evade PWD, but their effectiveness varies against different PWD.

[23] Since MLSEC only analyzes the HTML, we do not consider mixed-space attacks entailing perturbations of the URL.

8.4.3 *Mixed: PsP and WsP.*

- δ**Phish** and **Zenodo**. As presented in Tables 15, we analyze the impact of 52 attacks across the Preprocessing space and Website space of δPhish. These attacks have a detrimental effect on the detection performance of ML-PWD, particularly those based on F^r. Among these attacks, the combination attacks involving '*addHidP*' demonstrate the most significant impact on the ML-PWD. For instance, the attack '*addLngTxt_addHidP*' mentioned in Table 15a reduce the *tpr* of F^r-based ML-PWD from 0.79, 0.95 and 0.78 to 0.03, 0 and 0 respectively. This indicates a drastic reduction in the ability of the ML-PWD to detect and classify phishing instances. Similar situation is observed in ML-PWD of Zenodo, as illustrated in Table 25, the combination attack of PA^r+WA^r demonstrates a decrease in the *tpr* of ML-PWD trained on Zenodo. Notably, the attack '*delFtr_addHidP*' leads to a significant reduction in the *tpr* of F^r-based RF-PWD, dropping from 0.9 to 0.15. Furthermore, when encountering attack '*delSpn_addHidP*', the *tpr* decreases to 0.03. Other PA^r+WA^r combination attacks also prove effective in bypassing the detection of ML-PWD of Zenodo. For example, the attack '*delFtr_replPass*' results in a similar drop, and '*delFtr_addSusLnk*' reduces the *tpr* by 0.4–0.65. These findings highlight the susceptibility of ML-PWD trained on Zenodo to PA^r+WA^r attacks.
- **MLSEC**. We executed 53 kinds of PA^r+WA^r on MLSEC's PWD and evaluated their impact, which is reported in Tables 30. All of these combination attacks affected the decision of PWD, with 51 (i.e., except '*delSpn_modBgClr*' and '*delFtr_modBgClr*') out of 53 attacks noticeably degrading the confidence of at least one PWD. One particular attack, '*delFrm_addHidP*' minimizes the confidence of all PWD. Specifically, the confidence of $m0$ and $m2$ dropped from 0.9 to 0.01, while the confidence of other PWDs decreased by 0.16–0.5. This substantial reduction caused by this cheap attack is both shocking and expected, as this combination attack simultaneously considers the "feature space" and "problem space", i.e., both the high-level definitions of adversarial perturbations [78].

Takeaway: Compared to other attacks mixing evasion spaces, it is evident that PA^r+WA^r possess greater destructive power and have a substantial impact on the *tpr* of ML-PWD. These attacks are particularly potent because they traverse both the 'feature-space' (e.g., Preprocessing space) and 'problem-space' (e.g., Website space).

8.4.4 *Mixed: PsP and MsP.*

- δ**Phish** and **Zenodo**. We showcase 3 combination PA^r+MA^r attacks target ML-PWD trained on δPhish and Zenodo in Table 17 and Table 23, respectively. It is worth noting that these combination attacks are difficult to achieve and require high costs, as attackers must obtain write-access to deeper segments of the ML-PWD. Interestingly, despite the high cost associated with these attacks, they do not consistently and effectively disrupt ML-PWD, except for the attack '*delFtr_brTgs*' which reduces the *tpr* of F^r-based CN-PWD from 0.86 to 0.64.
- **MLSEC**. As depicted in Table 33, the combination attack PA^r+MA^r decreases the performance of all considered ML-PWD, but the impact is relatively minor. The largest impact is observed with '*delSpn_brTgs*' and '*delFtr_brTgs*'. These attacks lead to a decrease in the confidence of $m0$ by 0.16 and 0.13, respectively.

Takeaway: Costly attacks (which require both MsP and PsP) do not always possess formidable evasion capabilities. Intriguingly, they may only slightly affect certain detectors or have no impact on others.

9 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 [16]. We focus on the detection of

phishing *websites*. Papers that consider phishing in social networks [25], darkweb [101], phone calls [43], or emails [37] 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., [42]). Our focus is on attacks *against* ML-PWD. For instance, Tian et al. [91] 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., [102]) are outside our scope.

Let us compare our paper with existing works on evasion attacks against ML-PWD. We provide an overview in Table 6, 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 [40] and [82] (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 [87] 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 [74], which does not affect our paper because our entire evaluation is reproducible. Notably, Aleroud et al. [12] 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 [11], which considers 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. §7.2). It is concerning that few papers overlook the importance of statistically significant comparisons. The most remarkable effort is [85] which only performs 10 trials (we do 50), which are not enough to compute precise statistical tests.

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., [11, 61, 82]); 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., [9, 40, 63]). The only papers considering attackers that are closer to our threat model are [59, 72] and [9]. However, the ML-PWD considered in [9] is specific for *images*, which are tough to implement (cf. §7.3) and also implicitly resembles an ML system for computer vision—a task well-investigated in adversarial ML literature [24]. In contrast, the ML-PWD considered in [59] and [72] 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. §4.3).

10 CONCLUSIONS

We aim 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

Table 6. **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 [61]	2016	Problem	F^c	SL	✗	1 (✗)	✗
Corona [33]	2017	Feature	F^r, F^c	SL	✓	1 (✓)	✗
Bahnsen [23]	2018	Problem	F^u	DL	✗	1 (✗)	✗
Shirazi [85]	2019	Feature	F^c	SL	✗	4 (✓)	✓*
Sabir [82]	2020	Problem	F^u	SL, DL	✓	1 (✗)	✗
Lee [59]	2020	Feature	F^c	SL	✓	1 (✓)	✗
Abdelnabi [9]	2020	Problem	F^r	DL	✓	1 (✓)	✗
Aleroud [12]	2020	Both	F^u	SL	✗	2 (✓)	✗
Song [87]	2021	Problem	F^c	SL	✓	1 (✓*)	✗
Bac [21]	2021	Feature	F^u	SL, DL	✗	1 (✗)	✗
Lin [63]	2021	Feature	F^c	DL	✓	1 (✓)	✗
O'Mara [72]	2021	Feature	F^r	SL	✗	1 (✓)	✗
Al-Qurashi [11]	2021	Feature	F^u, F^c	SL, DL	✗	4 (✓)	✗
Gressel [40]	2021	Feature	F^c	SL, DL	✓	1 (✗)	✗
Ours		Both	F^u, F^r, F^c	DL, SL	✓	2 (✓)	✓

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 adversarial ML domain.

- 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 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.

REFERENCES

[1] 2015. UCI Phishing Websites Dataset. https://archive.ics.uci.edu/ml/datasets/phishing+websites.

[2] 2020. *Interet Crime Report.* Technical Report. Federal Bureau of Investigation. https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf

[3] 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

[4] 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

[5] 2022. All Adversarial Examples Papers. https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html.

[6] 2022. Machine Learning Security Evasion Competition. https://mlsec.io/.

[7] 2022. PhishTank. https://phishtank.org/.

[8] 2022. *State of the Phish 2022.* Technical Report. ProofPoint. https://www.proofpoint.com/it/resources/threat-reports/state-of-phish

[9] Sahar Abdelnabi, Katharina Krombholz, and Mario Fritz. 2020. VisualPhishNet: Zero-day phishing website detection by visual similarity. In *Proc. of CCS.*

[10] Bhupendra Acharya and Phani Vadrevu. 2021. {PhishPrint}: Evading Phishing Detection Crawlers by Prior Profiling. In *Proc. of USENIX Security.*

[11] Rayah Al-Qurashi, Ahmed AlEroud, Ahmad A Saifan, Mohammad Alsmadi, and Izzat Alsmadi. 2021. Generating Optimal Attack Paths in Generative Adversarial Phishing. In *Proc. of ISI.*

[12] Ahmed AlEroud and George Karabatis. 2020. Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks. In *Proc. of CODASPY.*

[13] Giovanni Apruzzese, Hyrum S Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, and Kevin Roundy. 2023. "Real Attackers Don't Compute Gradients": Bridging the Gap Between Adversarial ML Research and Practice. In *Proc. of SaTML.*

[14] Giovanni Apruzzese, Mauro Andreolini, Luca Ferretti, Mirco Marchetti, and Michele Colajanni. 2022. Modeling Realistic Adversarial Attacks against Network Intrusion Detection Systems. *ACM Digital Threats: Research and Practice* 3 (2022), 1–19.

[15] Giovanni Apruzzese and Michele Colajanni. 2018. Evading botnet detectors based on flows and Random Forest with adversarial samples. In *Proc. of NCA.*

[16] Giovanni Apruzzese, Michele Colajanni, Luca Ferretti, and Mirco Marchetti. 2019. Addressing Adversarial Attacks against Security Systems based on Machine Learning. In *Proc. of CyCon.*

[17] Giovanni Apruzzese, Mauro Conti, and Ying Yuan. 2022. SpacePhish: The Evasion-space of Adversarial Attacks against Phishing Website Detectors using Machine Learning. In *Proc. of ACSAC.*

[18] 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).

[19] Giovanni Apruzzese, Pavel Laskov, and Aliya Tastemirova. 2022. SoK: The Impact of Unlabelled Data in Cyberthreat Detection. In *Proc. IEEE EuroS&P.*

[20] 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.*

[21] 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. of ICMLANT.*

[22] Eugene Bagdasaryan and Vitaly Shmatikov. 2021. Blind backdoors in deep learning models. In *Proc. of USENIX Security.*

[23] Alejandro Correa Bahnsen, Ivan Torroledo, Luis David Camacho, and Sergio Villegas. 2018. DeepPhish: simulating malicious AI. In *Proc. of eCrime.*

[24] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proc. of CCS.*

[25] 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* 11 (2020).

[26] 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 *Proc. of MuC.*

[27] Andrei Butnaru, Alexios Mylonas, and Nikolaos Pitropakis. 2021. Towards lightweight URL-based phishing detection. *Future internet* (2021).

[28] 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* 12 (2013), 28–38.

[29] Nicholas Carlini. 2021. Poisoning the Unlabeled Dataset of {Semi-Supervised} Learning. In *Proc. of USENIX Security.*

[30] 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).

[31] Nicholas Carlini and David Wagner. 2016. Defensive distillation is not robust to adversarial examples. *arXiv:1607.04311* (2016).

[32] Lingwei Chen, Yanfang Ye, and Thirimachos Bourlai. 2017. Adversarial machine learning in malware detection: Arms race between evasion attack and defense. In *Proc. of EISIC.*

[33] 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. of ESORICS.*

[34] Darktrace. 2020. *Machine Learning in the Age of Cyber AI.* Technical Report. https://www.darktrace.com/es/resources/wp-machine-learning.pdf

[35] 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 TDSC.* (2017).

[36] 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. of USENIX Security.*

[37] Sevtap Duman, Kubra Kalkan-Cakmakci, Manuel Egele, William Robertson, and Engin Kirda. 2016. Emailprofiler: Spearphishing filtering with header and stylometric features of emails. In *Proc. of COMPSAC.*

[38] 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).

[39] Yang Gao, Benjamin M Ampel, and Sagar Samtani. 2023. Evading Anti-Phishing Models: A Field Note Documenting an Experience in the Machine Learning Security Evasion Competition 2022. *ACM Digital Threats: Research and Practice* (2023).

[40] Gilad Gressel, Niranjan Hegde, Archana Sreekumar, and Michael Darling. 2021. Feature Importance Guided Attack: A Model Agnostic Adversarial Attack. *arXiv:2106.14815* (2021).

[41] 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 *Proc. of ICWSM.*

[42] 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).

[43] Payas Gupta, Roberto Perdisci, and Mustaque Ahamad. 2018. Towards measuring the role of phone numbers in twitter-advertised spam. In *Proc. of AsiaCCS*.

[44] 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).

[45] 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. of APCC*.

[46] 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.*

[47] David Howell. 2015. Building better data protection with SIEM. *Elsevier Computer Fraud & Security* (2015).

[48] Mohith Gowda HR, Adithya MV, et al. 2020. Development of anti-phishing browser based on random forest and rule of extraction framework. *Cybersecurity* (2020).

[49] Ankit Kumar Jain and Brij B Gupta. 2018. Towards detection of phishing websites on client-side using machine learning based approach. *Telecom. Syst.* (2018).

[50] 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).

[51] 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).

[52] Michael I Jordan and Tom M Mitchell. 2015. Machine Learning: Trends, Perspectives, and Prospects. *Science* 349, 6245 (2015), 255–260.

[53] Houssain Kettani and Polly Wainwright. 2019. On the Top Threats to Cyber Systems. In *Proc. of ICICT*.

[54] 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. of AsiaCCS*.

[55] Engin Kirda and Christopher Kruegel. 2005. Protecting users against phishing attacks with antiphish. In *IEEE Annual Int. Comp. Soft. Appl. Conf.*

[56] Brian Kondracki, Babak Amin Azad, Oleksii Starov, and Nick Nikiforakis. 2021. Catching Transparent Phish: Analyzing and Detecting MITM Phishing Toolkits. In *Proc. of CCS*.

[57] 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 *Proc. of SPW*.

[58] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436.

[59] Jehyun Lee, Pingxiao Ye, Ruofan Liu, Dinil Mon Divakaran, and Mun Choon Chan. 2020. Building robust phishing detection system: an empirical analysis. *Proc. of NDSS MADWeb.* (2020).

[60] Qizhang Li, Yiwen Guo, and Hao Chen. 2020. Practical no-box adversarial attacks against DNNs. *Advances in Neural Information Processing Systems* 33 (2020), 12849–12860.

[61] 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 *Proc. of WWW*.

[62] Cong Liao, Haoti Zhong, Sencun Zhu, and Anna Squicciarini. 2018. Server-based manipulation attacks against machine learning models. In *Proc. of CODASPY*.

[63] 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. of USENIX Security*.

[64] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. 2014. Intelligent rule-based phishing websites classification. *IET Inf. Secur.* (2014).

[65] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. 2014. Predicting phishing websites based on self-structuring neural network. *Neur. Comp. Appl.* (2014).

[66] Tyler Moore. 2010. The economics of cybersecurity: Principles and policy options. *Elsevier Int. J. Critical Infrastructure Protection* (2010).

[67] 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 *Proc. of CCS*.

[68] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. 2021. Defeating {DNN-Based} Traffic Analysis Systems in {Real-Time} With Blind Adversarial Perturbations. In *Proc. of USENIX Security*.

[69] Amirreza Niakanlahiji, Bei-Tseng Chu, and Ehab Al-Shaer. 2018. PhishMon: A Machine Learning Framework for Detecting Phishing Webpages. In *Proc. of ISI*.

[70] 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 *Proc. of USENIX Security*.

[71] 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. of USENIX Security*.

[72] Alexander O'Mara, Izzat Alsmadi, and Ahmed AlEroud. 2021. Generative Adverserial Analysis of Phishing Attacks on Static and Dynamic Content of Webpages. In *Proc. of ISPA/BDCloud/SocialCom/SustainCom*.

[73] Ren Pang, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2020. AdvMind: Inferring adversary intent of black-box attacks. In *Proc. of KDD*.

[74] Thomas Kobber Panum, Kaspar Hageman, René Rydhof Hansen, and Jens Myrup Pedersen. 2020. Towards adversarial phishing detection. In *Proc. of USENIX Workshop CSET.*

[75] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proc. of AsiaCCS.*

[76] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. 2018. SoK: Security and Privacy in Machine Learning. In *Proc. of EuroS&P.*

[77] 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.*

[78] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. 2020. Intriguing Properties of Adversarial ML Attacks in the Problem Space. In *Proc. of IEEE S&P.*

[79] Pawan Prakash, Manish Kumar, Ramana Rao Kompella, and Minaxi Gupta. 2010. Phishnet: predictive blacklisting to detect phishing attacks. In *Proc. of InfoCOM.*

[80] Erwin Quiring, David Klein, Daniel Arp, Martin Johns, and Konrad Rieck. 2020. Adversarial Preprocessing: Understanding and Preventing {Image-Scaling} Attacks in Machine Learning. In *Proc. of USENIX Security.*

[81] 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.

[82] Bushra Sabir, M Ali Babar, and Raj Gaire. 2020. An evasion attack against ML-based phishing URL detectors. *arXiv:2005.08454* (2020).

[83] Shawn Shan, Emily Wenger, Bolun Wang, Bo Li, Haitao Zheng, and Ben Y. Zhao. 2020. Gotta Catch'Em All: Using Honeypots to Catch Adversarial Attacks on Neural Networks. In *Proc. of CCS.*

[84] Suhas R Sharma, Rahul Parthasarathy, and Prasad B Honnavalli. 2020. A Feature Selection Comparative Study for Web Phishing Datasets. In *Proc. of CONECCT.*

[85] Hossein Shirazi, Bruhadeshwar Bezawada, Indrakshi Ray, and Charles Anderson. 2019. Adversarial sampling attacks against phishing detection. In *Proc. of IFIP Annual Conference on Data and Applications Security and Privacy.*

[86] Charles Smutz and Angelos Stavrou. 2012. Malicious PDF detection using metadata and structural features. In *Proc. ACM Ann. Comp. Secur. Appl. Conf.*

[87] 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).

[88] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE T. Evol. Comput.* (2019).

[89] 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).

[90] Lizhen Tang and Qusay H Mahmoud. 2021. A survey of machine learning-based solutions for phishing website detection. *Machine Learning and Knowledge Extraction* (2021).

[91] 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. of IMC.*

[92] 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 *Proc. of USENIX Security.*

[93] Florian Tramèr, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, and Dan Boneh. 2019. Adversarial: Perceptual ad blocking meets adversarial machine learning. In *Proc. of CCS.*

[94] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble adversarial training: Attacks and defenses. In *Proc. of ICLR.*

[95] 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. of ARES.*

[96] Kalyan Veeramachaneni, Ignacio Arnaldo, Vamsi Korrapati, Constantinos Bassias, and Ke Li. 2016. AI^ 2: training a big data machine to defend. In *Proc. of BigDataSecurity.*

[97] Rakesh Verma and Keith Dyer. 2015. On the character of phishing URLs: Accurate and robust statistical learning classifiers. In *Proc. of CODASPY.*

[98] 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).

[99] Kelce S Wilson and Müge Ayse Kiy. 2014. Some fundamental Cybersecurity concepts. *IEEE Access* (2014).

[100] Aiping Xiong, Robert W Proctor, Weining Yang, and Ninghui Li. 2019. Embedding training within warnings improves skills of identifying phishing webpages. *Human Factors* (2019).

[101] 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 *Proc. of WWW.*

[102] 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 *Proc. of IEEE S&P.*

[103] 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 *Proc. of CCS.*

[104] Fei Zuo, Bokai Yang, Xiaopeng Li, and Qiang Zeng. 2019. Exploiting the inherent limitation of l0 adversarial examples. In *Proc. of RAID.*
[105] Nedim Šrndic and Pavel Laskov. 2014. Practical evasion of a learning-based classifier: A case study. In *Proc. of IEEE S&P.*

A COMPLETE BENCHMARK TABLES

We carry out our experiments by developing original software tools, all written in Python3 by leveraging well-known libraries (e.g., scikit-learn, Tensorflow). The ML-PWD using *RF* and *LR* are assessed on a system mounting an Intel Xeon W-2223@3.6GHz with 32GB RAM. For the *CN*, we use an nVidia P100 GPU. (Our results have been reproduced during the ACSAC artifact evaluation.)

Table 7. **Evasion Robustness of the ML-PWD on the** Zenodo **dataset.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific attack.

$\mathcal{A}$	F	no-atk	WA^u	WA^r	WA^c	$\widehat{WA^u}$	$\widehat{WA^r}$	$\widehat{WA^c}$	PA^u	PA^r	PA^c	MA^u	MA^r	MA^c
CN	F^u	0.96±0.007	1.00±0.000	0.93±0.020	1.00±0.000	1.00±0.000	0.95±0.018	1.00±0.000	1.00±0.017	0.95±0.018	1.00±0.017	0.18±0.222	0.95±0.018	0.18±0.222
	F^r	0.86±0.013	0.88±0.013	0.87±0.056	0.87±0.055	0.88±0.013	0.44±0.153	0.83±0.051	0.54±0.108	0.29±0.120	0.31±0.118	0.88±0.013	0.02±0.095	0.02±0.095
	F^c	0.97±0.009	0.92±0.036	0.93±0.020	0.94±0.063	0.92±0.036	0.92±0.016	0.83±0.115	1.00±0.011	0.90±0.031	0.99±0.017	0.51±0.131	0.92±0.036	0.15±0.211
RF	F^u	0.96±0.007	1.00±0.000	0.96±0.008	1.00±0.000	1.00±0.000	0.96±0.008	1.00±0.000	0.54±0.183	0.96±0.007	0.54±0.183	0.04±0.098	0.96±0.007	0.04±0.098
	F^r	0.90±0.013	0.90±0.013	0.88±0.024	0.88±0.025	0.90±0.013	0.71±0.053	0.80±0.025	0.59±0.086	0.47±0.082	0.30±0.088	0.90±0.013	0.04±0.155	0.04±0.155
	F^c	0.97±0.009	0.98±0.064	0.94±0.012	0.94±0.171	0.98±0.063	0.94±0.010	0.94±0.191	0.65±0.101	0.94±0.010	0.21±0.134	0.07±0.115	0.92±0.012	0.03±0.158
LR	F^u	0.97±0.005	1.00±0.000	0.95±0.005	1.00±0.000	1.00±0.000	0.96±0.005	1.00±0.000	0.73±0.071	0.96±0.006	0.73±0.071	0.00±0.000	0.96±0.006	0.00±0.000
	F^r	0.80±0.013	0.80±0.013	0.65±0.043	0.64±0.040	0.80±0.013	0.54±0.027	0.56±0.022	0.61±0.007	0.08±0.013	0.01±0.010	0.80±0.013	0.00±0.000	0.00±0.000
	F^c	0.98±0.005	0.82±0.035	0.95±0.015	0.32±0.079	0.80±0.038	0.93±0.014	0.32±0.132	0.46±0.053	0.91±0.032	0.06±0.025	0.00±0.000	0.76±0.036	0.00±0.000

Table 8. **Evasion Robustness of the ML-PWD on the** δPhish **dataset.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific attack.

$\mathcal{A}$	F	no-atk	WA^u	WA^r	WA^c	$\widehat{WA^u}$	$\widehat{WA^r}$	$\widehat{WA^c}$	PA^u	PA^r	PA^c	MA^u	MA^r	MA^c
CN	F^u	0.65±0.028	0.91±0.276	0.65±0.029	0.91±0.275	0.90±0.299	0.65±0.029	0.90±0.300	0.60±0.165	0.65±0.028	0.60±0.165	0.14±0.346	0.65±0.028	0.14±0.346
	F^r	0.79±0.013	0.80±0.013	0.35±0.018	0.34±0.017	0.80±0.013	0.86±0.033	0.88±0.020	0.46±0.065	0.69±0.038	0.46±0.064	0.81±0.013	0.00±0.000	0.00±0.000
	F^c	0.95±0.010	0.88±0.066	0.93±0.012	0.84±0.113	0.89±0.046	0.89±0.020	0.87±0.058	0.90±0.107	0.58±0.059	0.82±0.163	0.04±0.198	0.01±0.011	0.04±0.196
RF	F^u	0.56±0.037	0.84±0.330	0.56±0.036	0.84±0.330	0.84±0.330	0.56±0.034	0.84±0.331	0.57±0.238	0.56±0.037	0.57±0.238	0.01±0.053	0.56±0.037	0.01±0.053
	F^r	0.95±0.008	0.95±0.009	0.84±0.003	0.84±0.043	0.95±0.009	0.80±0.038	0.94±0.009	0.84±0.049	0.55±0.090	0.95±0.055	0.95±0.008	0.00±0.000	0.00±0.000
	F^c	0.95±0.009	0.90±0.020	0.92±0.006	0.77±0.047	0.90±0.017	0.86±0.018	0.92±0.015	0.90±0.065	0.68±0.013	0.86±0.097	0.88±0.026	0.00±0.001	0.00±0.000
LR	F^u	0.30±0.014	0.21±0.332	0.30±0.015	0.22±0.341	0.26±0.364	0.30±0.015	0.24±0.359	0.64±0.256	0.30±0.014	0.64±0.256	0.00±0.000	0.30±0.014	0.00±0.000
	F^r	0.78±0.011	0.78±0.011	0.57±0.014	0.56±0.047	0.78±0.011	0.60±0.030	0.63±0.010	0.80±0.029	0.04±0.006	0.45±0.068	0.78±0.011	0.00±0.000	0.00±0.000
	F^c	0.86±0.014	0.47±0.094	0.81±0.011	0.36±0.102	0.73±0.126	0.73±0.018	0.63±0.150	0.65±0.157	0.23±0.014	0.32±0.109	0.00±0.000	0.00±0.000	0.00±0.000

Evasion Performance We report the complete results of all the 12 considered evasion attacks against all the 18 considered ML-PWD in Table 7 (for Zenodo) and Table 8 (for δPhish). These tables also include the performance in non-adversarial settings computed on the 100 phishing samples (drawn from P_i that are used as a base for the adversarial samples). We remark that we chose such 100 samples by randomly selecting 100 samples which were correctly detected by the best ML-PWD on each dataset. As such, the tpr reported in the *no-atk* column can slightly differ from the one in Table 3 (which is computed on the entire P_i).

Runtime. We report in Table 9 the runtime for training and testing all our ML-PWD in non-adversarial scenarios. The values denote the average runtime (and standard deviation) across the 50 trials. Training the *RF* and *LR* uses all cores/threads of our CPU.

Table 9. **Execution Times** for training (on $\mathcal{D}$) and testing (on both P_i and B_i) the ML models used by our ML-PWD.

$\mathcal{A}$	F	Zenodo		δPhish	
		Train (s)	Test (ms)	Train (s)	Test (ms)
CN	F^u	110.88±15.318	178.13±9.661	201.314±21.753	301.91±46.133
	F^r	76.61±4.562	171.95±10.577	167.74±25.197	273.4±43.99
	F^c	152.325±13.183	222.696±86.618	165.486±23.367	274.84±47.975
RF	F^u	0.152±0.0052	7.59±0.208	0.583±0.0181	28.09±0.402
	F^r	0.146±0.0037	7.85±0.07	0.369±0.0181	22.39±0.151
	F^c	0.179±0.0035	9.39±0.312	0.44±0.0062	23.6±0.205
LR	F^u	0.045±0.019	0.1±0.005	0.185±0.0285	0.45±0.895
	F^r	0.055±0.0182	0.09±0.003	0.083±0.0509	0.74±1.161
	F^c	0.063±0.0179	0.17±0.014	0.301±0.0678	0.36±0.678

B COMPLEMENTARY WA^r FOR ZENODO AND δPHISH

As we mentioned in §6.3.2, we applied two different WA^r to the ML-PWD of δPhish and Zenodo (i.e., replOnc: swap <a href='link'> into <a onclick="this.href='link'"> on Zenodo, and addInLnk: insert can not see to the samples of δPhish), and report their influence in Figs. 7. In this section, we apply the same WA^r, but with the datasets swapped to see if the influence will change, i.e., applying addInLnk to Zenodo and applying replOnc to δPhish. The new influence on each dataset is depicted in Table. 10. Comparing with the Figs.7, it can be concluded that the δPhish is more vulnerable to addInLnk, whereas their impact on Zenodo are similar.

Table 10. **Impact of new** WA^r, reported as the average (and std. dev.) tpr over the 50 trials.

$\mathcal{A}$	F	Zenodo		δPhish	
		tpr (no-atk)	tpr (addInLnk)	tpr (no-atk)	tpr(replOnc)
CN	F^u	0.96±0.008	0.95±0.018	0.55±0.030	0.65±0.029
	F^r	0.88±0.018	0.61±0.034	0.81±0.019	0.89±0.018
	F^c	0.97±0.006	0.97±0.021	0.93±0.013	0.93±0.012
RF	F^u	0.98±0.004	0.96±0.008	0.45±0.022	0.56±0.036
	F^r	0.93±0.013	0.94±0.018	0.94±0.016	0.99±0.003
	F^c	0.98±0.006	0.97±0.008	0.97±0.007	0.98±0.006
LR	F^u	0.95±0.009	0.96±0.002	0.24±0.017	0.3±0.015
	F^r	0.82±0.017	0.95±0.005	0.74±0.025	0.78±0.014
	F^c	0.96±0.007	0.98±0.007	0.81±0.020	0.89±0.011

C BENCHMARK: RESULTS OF THE NEW EXPERIMENTS

We now report the *complete* results of *all* our new experiments, which we discussed in §8.

C.1 Perturbation's impact on δPhish

We report new WA^r's impact on the ML-PWD generated on δPhish in Table 11 and Table 12. PsP and WsP's influence were depicted in Table 13. And Table 14 describes the tpr of ML-PWD generated on δPhish against $u\text{WA}^u$. Table 16, 17, 15 and 18 report the influence of hybrid space attacks on δPhish.

Table 11. **Impact of** iWA^r **on** δPhish**.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific iWsP perturbation.

$\mathcal{A}$	F	no-atk	replOnc	delHidIt	addHidP	replJS	replRet	htEsc	htEncd	replPass	replOnfoc	addSusLnk
CN	F^u	0.65±0.028	0.65±0.029	0.65±0.029	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.035	0.64±0.031
	F^r	0.79±0.013	0.89±0.018	0.81±0.013	0.03±0.006	0.79±0.011	0.81±0.013	0.94±0.03	1.0±0.0	0.81±0.013	0.81±0.013	0.19±0.012
	F^c	0.95±0.010	0.93±0.012	0.95±0.016	0.22±0.059	0.89±0.021	0.96±0.011	0.99±0.01	0.99±0.014	0.95±0.011	0.95±0.013	0.79±0.039
RF	F^u	0.56±0.037	0.56±0.036	0.56±0.035	0.56±0.033	0.56±0.034	0.57±0.033	0.57±0.031	0.56±0.033	0.57±0.033	0.56±0.037	0.56±0.032
	F^r	0.95±0.008	0.99±0.003	0.88±0.011	0.0±0.0	0.81±0.021	0.95±0.008	1.0±0.003	1.0±0.0	0.95±0.008	0.95±0.008	0.44±0.069
	F^c	0.95±0.009	0.98±0.006	0.93±0.01	0.04±0.017	0.86±0.015	0.95±0.01	1.0±0.007	1.0±0.0	0.95±0.009	0.94±0.009	0.48±0.043
LR	F^u	0.30±0.014	0.3±0.015	0.29±0.015	0.3±0.015	0.3±0.016	0.3±0.014	0.3±0.014	0.3±0.015	0.3±0.014	0.3±0.021	0.3±0.014
	F^r	0.78±0.011	0.78±0.014	0.68±0.017	0.0±0.0	0.68±0.005	0.78±0.011	0.84±0.006	1.0±0.0	0.78±0.011	0.78±0.011	0.3±0.009
	F^c	0.86±0.014	0.89±0.011	0.82±0.016	0.17±0.015	0.78±0.01	0.86±0.014	0.92±0.015	1.0±0.005	0.87±0.014	0.74±0.042	0.62±0.025

Table 12. **Impact of** eWA^r **and** rWA^r **on** δPhish**.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific eWA^r or rWA^r attack.

$\mathcal{A}$	F	no-atk	eWA^r								rWA^r			
			addImgBot	modFntTyp	modCpy	addIcn	delSusLnk	delSusFrm	modTtl	delCpy	modBgimg	modBgClr	modFntClr	modFntSiz
CN	F^u	0.65±0.028	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.63±0.036	0.64±0.031
	F^r	0.79±0.013	0.63±0.063	0.81±0.013	0.77±0.016	0.71±0.024	0.84±0.021	0.75±0.012	0.81±0.013	0.77±0.016	0.81±0.013	0.81±0.013	0.81±0.013	0.81±0.013
	F^c	0.95±0.010	0.92±0.032	0.95±0.011	0.94±0.014	0.92±0.021	0.93±0.012	0.93±0.016	0.95±0.011	0.94±0.014	0.95±0.011	0.95±0.011	0.94±0.017	0.95±0.011
RF	F^u	0.56±0.037	0.57±0.034	0.56±0.033	0.56±0.033	0.56±0.033	0.56±0.033	0.56±0.032	0.56±0.033	0.56±0.034	0.56±0.034	0.57±0.034	0.56±0.036	0.56±0.033
	F^r	0.95±0.008	0.88±0.026	0.95±0.008	0.95±0.007	0.89±0.019	0.92±0.011	0.91±0.021	0.95±0.008	0.95±0.007	0.95±0.008	0.95±0.008	0.95±0.008	0.95±0.008
	F^c	0.95±0.009	0.88±0.015	0.95±0.009	0.94±0.009	0.89±0.015	0.92±0.007	0.91±0.009	0.95±0.009	0.94±0.009	0.95±0.009	0.95±0.009	0.94±0.009	0.95±0.009
LR	F^u	0.30±0.014	0.3±0.014	0.3±0.014	0.3±0.014	0.3±0.015	0.3±0.015	0.3±0.016	0.3±0.015	0.3±0.016	0.3±0.016	0.3±0.014	0.3±0.024	0.3±0.014
	F^r	0.78±0.011	0.47±0.026	0.78±0.011	0.77±0.011	0.61±0.015	0.83±0.007	0.75±0.025	0.79±0.011	0.77±0.011	0.78±0.011	0.78±0.011	0.78±0.011	0.78±0.011
	F^c	0.86±0.014	0.66±0.028	0.87±0.014	0.89±0.013	0.82±0.013	0.91±0.009	0.78±0.018	0.87±0.014	0.89±0.013	0.87±0.013	0.87±0.014	0.74±0.044	0.87±0.014

Table 13. **Impact of** PA^r **and** MA^r **on** δPhish. The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific PsP or MsP attack.

$\mathcal{A}$	F	no-atk	PsP							MsP					
			delTxt	delFrm	delSpn	delTtl	addLngTxt	delFtr	replSusFtrLnk	brTg	delHt	delHd	delBdy	brTgs	hmg
CN	F^u	0.65±0.028	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.65±0.032
	F^r	0.79±0.013	0.78±0.014	0.75±0.012	0.8±0.013	0.81±0.013	0.81±0.013	0.76±0.015	0.79±0.011	0.81±0.013	1.0±0.0	0.79±0.009	0.87±0.018	0.81±0.012	0.76±0.019
	F^c	0.95±0.010	0.89±0.034	0.93±0.016	0.95±0.012	0.91±0.027	0.95±0.011	0.93±0.013	0.95±0.011	0.95±0.011	0.99±0.014	0.82±0.045	0.98±0.015	0.95±0.011	0.78±0.034
RF	F^u	0.56±0.037	0.56±0.033	0.56±0.032	0.57±0.032	0.57±0.032	0.57±0.033	0.56±0.033	0.56±0.035	0.56±0.035	0.56±0.035	0.56±0.033	0.57±0.033	0.56±0.034	0.56±0.036
	F^r	0.95±0.008	0.94±0.012	0.91±0.021	0.95±0.007	0.94±0.012	0.95±0.008	0.91±0.01	0.94±0.011	0.95±0.008	1.0±0.0	0.83±0.019	1.0±0.003	0.95±0.008	0.79±0.024
	F^c	0.95±0.009	0.92±0.012	0.91±0.009	0.94±0.009	0.93±0.011	0.95±0.009	0.94±0.01	0.94±0.009	0.95±0.009	1.0±0.0	0.86±0.015	1.0±0.007	0.94±0.009	0.8±0.017
LR	F^u	0.30±0.014	0.3±0.015	0.3±0.016	0.3±0.015	0.3±0.016	0.3±0.016	0.3±0.015	0.3±0.015	0.3±0.015	0.3±0.014	0.3±0.014	0.3±0.016	0.3±0.016	0.3±0.014
	F^r	0.78±0.011	0.64±0.024	0.75±0.025	0.75±0.016	0.64±0.025	0.78±0.011	0.79±0.016	0.76±0.011	0.78±0.011	1.0±0.0	0.65±0.01	0.84±0.006	0.78±0.011	0.69±0.01
	F^c	0.86±0.014	0.78±0.018	0.78±0.018	0.87±0.014	0.76±0.02	0.87±0.014	0.89±0.014	0.85±0.012	0.87±0.013	1.0±0.004	0.76±0.03	0.95±0.008	0.87±0.013	0.76±0.013

Table 14. Impact of $u\mathrm{WA}^u$ on ML-PWD of δPhish.

$\mathcal{A}$	F	no-atk	replChar	sepWrd	delChar	swpChar	addChar	atkPth
CN	F^u	0.65±0.028	0.64±0.043	0.64±0.038	0.63±0.033	0.63±0.037	0.64±0.044	0.6±0.029
	F^r	0.79±0.013	0.81±0.013	0.79±0.016	0.8±0.014	0.81±0.014	0.81±0.014	0.8±0.013
	F^c	0.95±0.010	0.95±0.009	0.95±0.01	0.94±0.01	0.95±0.011	0.94±0.012	0.94±0.009
RF	F^u	0.56±0.037	0.56±0.03	0.59±0.024	0.56±0.029	0.56±0.032	0.56±0.031	0.52±0.027
	F^r	0.95±0.008	0.95±0.009	0.95±0.008	0.95±0.009	0.95±0.009	0.95±0.009	0.95±0.008
	F^c	0.95±0.009	0.94±0.009	0.94±0.009	0.92±0.011	0.94±0.009	0.94±0.009	0.94±0.01
LR	F^u	0.30±0.014	0.3±0.02	0.31±0.024	0.28±0.019	0.28±0.02	0.29±0.019	0.29±0.015
	F^r	0.78±0.011	0.78±0.011	0.79±0.012	0.77±0.012	0.78±0.012	0.78±0.011	0.78±0.011
	F^c	0.86±0.014	0.83±0.018	0.85±0.028	0.84±0.016	0.83±0.019	0.83±0.021	0.88±0.01

Table 15. **Impact of PA^r+WA^r on ML-PWD generated on δPhish.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific PsP+WsP perturbation.

(a) Impact of PA^r+WA^r on ML-PWD generated on δPhish. PA^r is *addLngTxt*.

$\mathcal{A}$	F	no-atk	addLngTxt_ addInLnk	addLngTxt_ delHidIt	addLngTxt_ replOnfoc	addLngTxt_ replPass	addLngTxt_ addHidP	addLngTxt_ replJS	addLngTxt_ delSusLnk	addLngTxt_ modCpy	addLngTxt_ modTtl	addLngTxt_ addIcn	addLngTxt_ addSusLnk	addLngTxt_ addImgBot	addLngTxt_ modBgClr
CN	F^u	0.65±0.028	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.032	0.64±0.031	0.63±0.036	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031
	F^r	0.79±0.013	0.35±0.017	0.81±0.013	0.81±0.013	0.81±0.013	0.03±0.006	0.79±0.011	0.84±0.021	0.77±0.016	0.81±0.013	0.71±0.024	0.2±0.013	0.63±0.063	0.81±0.013
	F^c	0.95±0.010	0.92±0.029	0.95±0.016	0.95±0.011	0.95±0.011	0.22±0.059	0.89±0.021	0.92±0.017	0.94±0.014	0.95±0.011	0.92±0.021	0.8±0.033	0.91±0.035	0.95±0.011
RF	F^u	0.56±0.037	0.56±0.035	0.57±0.034	0.56±0.033	0.57±0.031	0.56±0.033	0.56±0.032	0.56±0.035	0.56±0.034	0.56±0.033	0.56±0.034	0.56±0.034	0.56±0.034	0.56±0.034
	F^r	0.95±0.008	0.84±0.043	0.88±0.011	0.95±0.008	0.95±0.008	0.0±0.0	0.81±0.021	0.92±0.011	0.95±0.007	0.95±0.008	0.89±0.019	0.46±0.045	0.88±0.026	0.95±0.008
	F^c	0.95±0.009	0.92±0.017	0.93±0.01	0.95±0.009	0.95±0.009	0.04±0.017	0.86±0.015	0.92±0.008	0.94±0.009	0.95±0.009	0.89±0.015	0.58±0.039	0.88±0.015	0.95±0.009
LR	F^u	0.30±0.014	0.3±0.015	0.3±0.014	0.3±0.016	0.3±0.015	0.3±0.016	0.3±0.016	0.3±0.024	0.3±0.014	0.3±0.015	0.3±0.014	0.3±0.016	0.3±0.015	0.3±0.016
	F^r	0.78±0.011	0.57±0.045	0.68±0.017	0.78±0.011	0.78±0.011	0.0±0.0	0.68±0.005	0.83±0.007	0.77±0.011	0.79±0.011	0.61±0.015	0.38±0.026	0.44±0.026	0.78±0.011
	F^c	0.86±0.014	0.8±0.017	0.81±0.014	0.87±0.013	0.87±0.014	0.17±0.016	0.78±0.01	0.8±0.043	0.89±0.013	0.87±0.014	0.82±0.012	0.68±0.021	0.66±0.03	0.87±0.015

(b) Impact of PA^r+WA^r on ML-PWD generated on δPhish. PA^r is *delFrm*.

$\mathcal{A}$	F	no-atk	delFrm_ addInLnk	delFrm_ onclick	delFrm_ delHidIt	delFrm_ addHidP	delFrm_ replJS	delFrm_ delSusLnk	delFrm_ addImgBot	delFrm_ modFntSiz	delFrm_ modBgimg	delFrm_ modBgClr	delFrm_ delCpy	delFrm_ modTtl	delFrm_ modCpy	delFrm_ addIcn	delFrm_ replRet
CN	F^u	0.65±0.028	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031
	F^r	0.79±0.013	0.25±0.021	0.93±0.019	0.76±0.011	0.01±0.004	0.73±0.009	0.87±0.018	0.54±0.081	0.76±0.011	0.76±0.011	0.76±0.011	0.72±0.014	0.76±0.011	0.72±0.014	0.64±0.024	0.76±0.011
	F^c	0.95±0.010	0.85±0.04	0.97±0.014	0.93±0.017	0.19±0.062	0.81±0.021	0.91±0.013	0.88±0.036	0.91±0.017	0.91±0.017	0.91±0.017	0.89±0.02	0.91±0.017	0.89±0.02	0.86±0.031	0.91±0.017
RF	F^u	0.56±0.037	0.56±0.033	0.56±0.033	0.56±0.034	0.56±0.034	0.57±0.032	0.57±0.033	0.56±0.033	0.56±0.033	0.56±0.034	0.57±0.032	0.56±0.034	0.56±0.033	0.56±0.034	0.57±0.032	0.56±0.036
	F^r	0.95±0.008	0.85±0.055	0.99±0.007	0.89±0.015	0.0±0.0	0.74±0.015	0.91±0.014	0.85±0.018	0.9±0.011	0.9±0.011	0.9±0.011	0.9±0.01	0.9±0.011	0.9±0.01	0.82±0.02	0.9±0.011
	F^c	0.95±0.009	0.88±0.023	0.98±0.011	0.92±0.009	0.03±0.016	0.8±0.011	0.91±0.011	0.87±0.017	0.9±0.012	0.9±0.012	0.9±0.012	0.91±0.009	0.9±0.012	0.91±0.009	0.83±0.021	0.9±0.012
LR	F^u	0.30±0.014	0.3±0.015	0.3±0.015	0.3±0.016	0.3±0.015	0.3±0.014	0.3±0.015	0.3±0.016	0.3±0.015	0.3±0.015	0.3±0.015	0.3±0.016	0.3±0.016	0.3±0.014	0.3±0.015	0.3±0.017
	F^r	0.78±0.011	0.43±0.082	0.79±0.013	0.72±0.024	0.0±0.0	0.68±0.01	0.84±0.008	0.43±0.04	0.74±0.019	0.74±0.019	0.74±0.019	0.73±0.019	0.75±0.019	0.73±0.019	0.57±0.022	0.74±0.019
	F^c	0.86±0.014	0.7±0.028	0.85±0.019	0.82±0.012	0.16±0.011	0.73±0.01	0.89±0.013	0.57±0.026	0.79±0.012	0.79±0.011	0.79±0.011	0.81±0.016	0.79±0.011	0.81±0.016	0.74±0.013	0.79±0.011

(c) Impact of PA^r+WA^r on ML-PWD generated on δPhish. PA^r is *delFtr*.

$\mathcal{A}$	F	no-atk	delFtr_ addInLnk	delFtr_ delHidIt	delFtr_ replOnfoc	delFtr_ replPass	delFtr_ addHidP	delFtr_ replJS	delFtr_ delSusLnk	delFtr_ modCpy	delFtr_ modTtl	delFtr_ addIcn	delFtr_ addSusLnk	delFtr_ modBgClr
CN	F^u	0.65±0.028	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031
	F^r	0.79±0.013	0.26±0.012	0.76±0.014	0.76±0.015	0.76±0.015	0.01±0.01	0.76±0.014	0.83±0.021	0.75±0.017	0.76±0.015	0.64±0.022	0.14±0.015	0.76±0.015
	F^c	0.95±0.010	0.9±0.031	0.92±0.019	0.93±0.013	0.93±0.013	0.22±0.058	0.87±0.019	0.92±0.011	0.92±0.013	0.93±0.013	0.88±0.019	0.77±0.032	0.93±0.013
RF	F^u	0.56±0.037	0.57±0.032	0.56±0.034	0.56±0.033	0.56±0.034	0.56±0.032	0.56±0.033	0.56±0.034	0.56±0.034	0.56±0.033	0.56±0.033	0.56±0.034	0.56±0.033
	F^r	0.95±0.008	0.86±0.044	0.86±0.016	0.91±0.01	0.91±0.01	0.0±0.001	0.77±0.021	0.92±0.009	0.91±0.009	0.91±0.01	0.83±0.015	0.44±0.055	0.91±0.01
	F^c	0.95±0.009	0.91±0.021	0.91±0.009	0.94±0.01	0.94±0.01	0.03±0.016	0.85±0.012	0.92±0.008	0.93±0.012	0.94±0.01	0.86±0.021	0.56±0.045	0.94±0.01
LR	F^u	0.30±0.014	0.3±0.014	0.3±0.016	0.3±0.015	0.3±0.015	0.3±0.016	0.3±0.016	0.3±0.016	0.3±0.016	0.3±0.016	0.3±0.016	0.3±0.016	0.3±0.014
	F^r	0.78±0.011	0.61±0.053	0.69±0.014	0.79±0.016	0.79±0.016	0.0±0.0	0.71±0.012	0.85±0.003	0.78±0.016	0.8±0.016	0.62±0.016	0.38±0.028	0.79±0.016
	F^c	0.86±0.014	0.84±0.016	0.82±0.012	0.89±0.014	0.89±0.014	0.19±0.021	0.78±0.01	0.9±0.009	0.89±0.012	0.89±0.014	0.85±0.014	0.72±0.019	0.89±0.014

(d) Impact of PA^r+WA^r on ML-PWD generated on δPhish. PA^r is *delSpn*.

$\mathcal{A}$	F	no-atk	delSpn_ addInLnk	delSpn_ replOnfoc	delSpn_ replPass	delSpn_ addHidP	delSpn_ replJS	delSpn_ delSusLnk	delSpn_ modCpy	delSpn_ modTtl	delSpn_ addIcn	delSpn_ addSusLnk	delSpn_ addImgBot	delSpn_ modBgClr
CN	F^u	0.65±0.028	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031
	F^r	0.79±0.013	0.35±0.019	0.8±0.013	0.8±0.013	0.06±0.025	0.78±0.011	0.82±0.019	0.78±0.014	0.8±0.013	0.69±0.022	0.22±0.023	0.61±0.067	0.8±0.013
	F^c	0.95±0.010	0.92±0.033	0.95±0.012	0.95±0.012	0.22±0.059	0.89±0.021	0.93±0.013	0.94±0.014	0.95±0.012	0.92±0.022	0.8±0.035	0.89±0.042	0.95±0.012
RF	F^u	0.56±0.037	0.57±0.032	0.56±0.032	0.57±0.033	0.56±0.032	0.56±0.034	0.56±0.034	0.56±0.033	0.56±0.032	0.57±0.035	0.56±0.032	0.56±0.032	0.57±0.032
	F^r	0.95±0.008	0.84±0.042	0.95±0.007	0.95±0.007	0.0±0.0	0.84±0.024	0.92±0.011	0.95±0.006	0.95±0.007	0.89±0.019	0.45±0.045	0.9±0.015	0.95±0.007
	F^c	0.95±0.009	0.92±0.015	0.94±0.009	0.94±0.009	0.03±0.016	0.85±0.015	0.92±0.007	0.94±0.009	0.94±0.009	0.89±0.015	0.54±0.036	0.88±0.015	0.94±0.009
LR	F^u	0.30±0.014	0.3±0.014	0.3±0.016	0.3±0.015	0.3±0.015	0.3±0.016	0.3±0.015	0.3±0.015	0.3±0.015	0.3±0.015	0.3±0.014	0.3±0.015	0.3±0.014
	F^r	0.78±0.011	0.55±0.056	0.75±0.016	0.75±0.016	0.0±0.0	0.68±0.009	0.83±0.007	0.74±0.016	0.76±0.016	0.6±0.028	0.32±0.018	0.45±0.028	0.75±0.016
	F^c	0.86±0.014	0.8±0.019	0.87±0.014	0.87±0.014	0.17±0.014	0.79±0.009	0.91±0.008	0.89±0.014	0.87±0.014	0.82±0.013	0.66±0.024	0.65±0.027	0.87±0.014

Table 16. **Impact of PA^r+PA^r on δPhish.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific PsP+PsP perturbation.

$\mathcal{A}$	F	no-atk	addLngTxt_delTtl	delFtr_delTtl	delFtr_addLngTxt	delSpn_delTtl	delSpn_delFtrer	delSpn_addLngTxt	delFrm_delFtr	delFrm_delSpn
CN	F^u	0.65±0.028	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031	0.64±0.031
	F^r	0.79±0.013	0.81±0.013	0.76±0.015	0.76±0.015	0.8±0.013	0.75±0.015	0.8±0.013	1.0±0.0	0.74±0.01
	F^c	0.95±0.010	0.91±0.027	0.89±0.026	0.93±0.013	0.9±0.03	0.92±0.014	0.95±0.012	0.99±0.014	0.9±0.018
RF	F^u	0.56±0.037	0.56±0.033	0.57±0.033	0.56±0.032	0.56±0.033	0.57±0.033	0.56±0.034	0.56±0.034	0.56±0.032
	F^r	0.95±0.008	0.94±0.012	0.89±0.014	0.91±0.01	0.94±0.011	0.91±0.011	0.95±0.007	1.0±0.0	0.91±0.01
	F^c	0.95±0.009	0.93±0.011	0.91±0.016	0.94±0.01	0.92±0.012	0.94±0.01	0.94±0.009	1.0±0.0	0.92±0.008
LR	F^u	0.30±0.014	0.3±0.015	0.3±0.015	0.3±0.014	0.3±0.015	0.3±0.015	0.3±0.014	0.3±0.015	0.3±0.014
	F^r	0.78±0.011	0.64±0.025	0.64±0.024	0.79±0.016	0.62±0.035	0.76±0.02	0.75±0.016	1.0±0.0	0.74±0.019
	F^c	0.86±0.014	0.76±0.02	0.79±0.019	0.89±0.014	0.76±0.021	0.89±0.014	0.87±0.014	1.0±0.005	0.81±0.01

Table 17. **Impact of PA^r+MA^r attacks on δPhish.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific PsP+MsP.

$\mathcal{A}$	F	no-atk	addLngTxt_delBdy	delfoot_delBdy	delSpn_delBdy
CN	F^u	0.65±0.028	0.64±0.031	0.64±0.031	0.64±0.031
	F^r	0.79±0.013	0.81±0.012	0.76±0.015	0.8±0.013
	F^c	0.95±0.010	0.95±0.011	0.93±0.013	0.95±0.012
RF	F^u	0.56±0.037	0.56±0.031	0.56±0.034	0.56±0.033
	F^r	0.95±0.008	0.95±0.008	0.91±0.01	0.95±0.006
	F^c	0.95±0.009	0.94±0.009	0.94±0.01	0.94±0.009
LR	F^u	0.30±0.014	0.3±0.014	0.3±0.014	0.3±0.015
	F^r	0.78±0.011	0.78±0.011	0.79±0.016	0.75±0.016
	F^c	0.86±0.014	0.87±0.014	0.89±0.014	0.87±0.014

Table 18. **Impact of WA^r+WA^r on δPhish.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific WsP+WsP.

$\mathcal{A}$	F	no-atk	replOnfoc_replRet	htEsc_replRet	htEncd_replRet
CN	F^u	0.65±0.028	0.64±0.031	0.63±0.035	0.64±0.031
	F^r	0.79±0.013	0.81±0.013	1.0±0.0	1.0±0.0
	F^c	0.95±0.010	0.96±0.011	0.97±0.033	0.99±0.014
RF	F^u	0.56±0.037	0.56±0.034	0.56±0.036	0.56±0.032
	F^r	0.95±0.008	0.95±0.008	1.0±0.0	1.0±0.0
	F^c	0.95±0.009	0.95±0.01	1.0±0.0	1.0±0.0
LR	F^u	0.30±0.014	0.3±0.016	0.3±0.023	0.3±0.015
	F^r	0.78±0.011	0.78±0.011	1.0±0.0	1.0±0.0
	F^c	0.86±0.014	0.86±0.014	0.91±0.069	1.0±0.004

C.2 Perturbation's impact on Zenodo

In this section, we present new perturbation's influence on Zenodo. Single attacks' influence is shown in Table 19, 20, 25, and hybrid attacks' impact is shown in Table 23, 26, 25 24.

Table 19. **Evasion Robustness of the ML-PWD against** iWA^r **on** Zenodo**.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific iWsP perturbation.

$\mathcal{A}$	F	no-atk	addInLnk	delHidIt	addHidP	replJS	replRet	htEsc	htEncd	replPass	replOnfoc	addSusLnk
CN	F^u	$0.96_{\pm 0.007}$	$0.95_{\pm 0.018}$	$0.95_{\pm 0.018}$	$0.93_{\pm 0.017}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.92_{\pm 0.023}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.95_{\pm 0.018}$	$0.96_{\pm 0.013}$
	F^r	$0.86_{\pm 0.013}$	$0.61_{\pm 0.034}$	$0.88_{\pm 0.012}$	$0.28_{\pm 0.008}$	$0.74_{\pm 0.05}$	$0.88_{\pm 0.013}$	$0.87_{\pm 0.025}$	$0.0_{\pm 0.0}$	$0.88_{\pm 0.013}$	$0.88_{\pm 0.013}$	$0.48_{\pm 0.022}$
	F^c	$0.97_{\pm 0.009}$	$0.97_{\pm 0.021}$	$0.96_{\pm 0.016}$	$0.86_{\pm 0.027}$	$0.95_{\pm 0.013}$	$0.97_{\pm 0.012}$	$0.92_{\pm 0.019}$	$0.9_{\pm 0.033}$	$0.97_{\pm 0.012}$	$0.97_{\pm 0.012}$	$0.97_{\pm 0.021}$
RF	F^u	$0.96_{\pm 0.007}$	$0.96_{\pm 0.008}$	$0.96_{\pm 0.008}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.96_{\pm 0.007}$	$0.98_{\pm 0.005}$
	F^r	$0.90_{\pm 0.013}$	$0.94_{\pm 0.018}$	$0.84_{\pm 0.01}$	$0.03_{\pm 0.064}$	$0.71_{\pm 0.016}$	$0.9_{\pm 0.013}$	$0.84_{\pm 0.027}$	$0.0_{\pm 0.0}$	$0.9_{\pm 0.013}$	$0.9_{\pm 0.013}$	$0.64_{\pm 0.062}$
	F^c	$0.97_{\pm 0.009}$	$0.97_{\pm 0.008}$	$0.97_{\pm 0.01}$	$0.96_{\pm 0.006}$	$0.96_{\pm 0.006}$	$0.98_{\pm 0.004}$	$0.96_{\pm 0.007}$	$0.96_{\pm 0.007}$	$0.98_{\pm 0.005}$	$0.97_{\pm 0.01}$	$0.97_{\pm 0.008}$
LR	F^u	$0.97_{\pm 0.005}$	$0.96_{\pm 0.002}$	$0.96_{\pm 0.005}$	$0.97_{\pm 0.005}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.005}$	$0.97_{\pm 0.009}$	$0.97_{\pm 0.004}$	$0.95_{\pm 0.005}$	$0.97_{\pm 0.004}$
	F^r	$0.80_{\pm 0.013}$	$0.95_{\pm 0.005}$	$0.79_{\pm 0.014}$	$0.24_{\pm 0.019}$	$0.46_{\pm 0.013}$	$0.8_{\pm 0.013}$	$0.55_{\pm 0.009}$	$0.0_{\pm 0.0}$	$0.8_{\pm 0.013}$	$0.8_{\pm 0.013}$	$0.72_{\pm 0.0}$
	F^c	$0.98_{\pm 0.005}$	$0.98_{\pm 0.007}$	$0.97_{\pm 0.007}$	$0.95_{\pm 0.007}$	$0.96_{\pm 0.005}$	$0.98_{\pm 0.002}$	$0.97_{\pm 0.007}$	$0.97_{\pm 0.005}$	$0.98_{\pm 0.002}$	$0.98_{\pm 0.003}$	$0.97_{\pm 0.0}$

Table 20. **Evasion Robustness of the ML-PWD against** eWA^r **and** rWA^r **on** Zenodo**.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific eWsP or rWsP perturbation.

$\mathcal{A}$	F	no-atk	eWsP								rWsP			
			addImgBot	modFntTyp	modCpy	addIcn	delSusLnk	delSusFrm	modTtl	delCpy	modBgimg	modBgClr	modFntClr	modFntSiz
CN	F^u	$0.96_{\pm 0.007}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$
	F^r	$0.86_{\pm 0.013}$	$0.74_{\pm 0.06}$	$0.88_{\pm 0.013}$	$0.76_{\pm 0.045}$	$0.82_{\pm 0.019}$	$0.91_{\pm 0.016}$	$0.86_{\pm 0.036}$	$0.88_{\pm 0.013}$	$0.76_{\pm 0.045}$	$0.88_{\pm 0.013}$	$0.88_{\pm 0.013}$	$0.88_{\pm 0.013}$	$0.88_{\pm 0.013}$
	F^c	$0.97_{\pm 0.009}$	$0.97_{\pm 0.01}$	$0.97_{\pm 0.012}$	$0.97_{\pm 0.013}$	$0.96_{\pm 0.012}$	$0.96_{\pm 0.012}$	$0.97_{\pm 0.012}$	$0.97_{\pm 0.013}$	$0.97_{\pm 0.013}$	$0.97_{\pm 0.012}$	$0.97_{\pm 0.012}$	$0.97_{\pm 0.012}$	$0.97_{\pm 0.012}$
RF	F^u	$0.96_{\pm 0.007}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$
	F^r	$0.90_{\pm 0.013}$	$0.8_{\pm 0.045}$	$0.9_{\pm 0.013}$	$0.89_{\pm 0.014}$	$0.84_{\pm 0.025}$	$0.9_{\pm 0.015}$	$0.9_{\pm 0.014}$	$0.9_{\pm 0.013}$	$0.89_{\pm 0.014}$	$0.9_{\pm 0.013}$	$0.9_{\pm 0.013}$	$0.9_{\pm 0.013}$	$0.9_{\pm 0.013}$
	F^c	$0.97_{\pm 0.009}$	$0.98_{\pm 0.006}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.96_{\pm 0.006}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.004}$	$0.98_{\pm 0.004}$	$0.98_{\pm 0.004}$	$0.98_{\pm 0.005}$
LR	F^u	$0.97_{\pm 0.005}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$
	F^r	$0.80_{\pm 0.013}$	$0.9_{\pm 0.021}$	$0.8_{\pm 0.013}$	$0.8_{\pm 0.013}$	$0.88_{\pm 0.013}$	$0.74_{\pm 0.017}$	$0.8_{\pm 0.012}$	$0.79_{\pm 0.013}$	$0.8_{\pm 0.013}$	$0.8_{\pm 0.013}$	$0.8_{\pm 0.013}$	$0.8_{\pm 0.013}$	$0.8_{\pm 0.013}$
	F^c	$0.98_{\pm 0.005}$	$0.97_{\pm 0.008}$	$0.98_{\pm 0.001}$	$0.98_{\pm 0.002}$	$0.97_{\pm 0.008}$	$0.96_{\pm 0.002}$	$0.97_{\pm 0.008}$	$0.98_{\pm 0.002}$	$0.98_{\pm 0.001}$	$0.98_{\pm 0.001}$	$0.98_{\pm 0.001}$	$0.98_{\pm 0.002}$	$0.98_{\pm 0.005}$

Table 21. **Impact of** $u\mathrm{WA}^u$ **on ML-PWD of** Zenodo**.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific $u\mathrm{WA}^u$ perturbation.

$\mathcal{A}$	F	no-atk	replChar	sepWrd	delChar	swpChar	addChar	atkPth
CN	F^u	$0.96_{\pm 0.007}$	$0.98_{\pm 0.012}$	$0.95_{\pm 0.024}$	$0.99_{\pm 0.009}$	$0.99_{\pm 0.013}$	$0.99_{\pm 0.007}$	$0.97_{\pm 0.017}$
	F^r	$0.86_{\pm 0.013}$	$0.5_{\pm 0.043}$	$0.49_{\pm 0.04}$	$0.5_{\pm 0.043}$	$0.49_{\pm 0.04}$	$0.49_{\pm 0.038}$	$0.5_{\pm 0.043}$
	F^c	$0.97_{\pm 0.009}$	$0.98_{\pm 0.017}$	$0.95_{\pm 0.025}$	$0.99_{\pm 0.024}$	$0.99_{\pm 0.019}$	$0.99_{\pm 0.017}$	$0.97_{\pm 0.021}$
RF	F^u	$0.96_{\pm 0.007}$	$1.0_{\pm 0.004}$	$0.98_{\pm 0.0}$	$1.0_{\pm 0.005}$	$1.0_{\pm 0.004}$	$1.0_{\pm 0.006}$	$0.99_{\pm 0.002}$
	F^r	$0.90_{\pm 0.013}$	$0.73_{\pm 0.043}$	$0.73_{\pm 0.043}$	$0.74_{\pm 0.043}$	$0.74_{\pm 0.041}$	$0.75_{\pm 0.041}$	$0.73_{\pm 0.043}$
	F^c	$0.97_{\pm 0.009}$	$1.0_{\pm 0.005}$	$0.99_{\pm 0.001}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.002}$	$1.0_{\pm 0.003}$	$0.98_{\pm 0.006}$
LR	F^u	$0.97_{\pm 0.005}$	$0.99_{\pm 0.002}$	$0.99_{\pm 0.003}$	$1.0_{\pm 0.003}$	$1.0_{\pm 0.0}$	$0.99_{\pm 0.001}$	$0.97_{\pm 0.007}$
	F^r	$0.80_{\pm 0.013}$	$0.78_{\pm 0.0}$	$0.79_{\pm 0.0}$	$0.79_{\pm 0.0}$	$0.79_{\pm 0.0}$	$0.8_{\pm 0.0}$	$0.78_{\pm 0.0}$
	F^c	$0.98_{\pm 0.005}$	$0.99_{\pm 0.0}$	$1.0_{\pm 0.001}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.97_{\pm 0.004}$

Table 22. Impact of PA^r+WA^r on Zenodo. The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific PsP+WsP perturbation.

(a) Impact of PA^r+WA^r on ML-PWD generated on Zenodo. PA^r is *addLngTxt*.

$\mathcal{A}$	F	no-atk	addLngTxt_ addInLnk	addLngTxt_ delHidIt	addLngTxt_ replOnfoc	addLngTxt_ replPass	addLngTxt_ addHidP	addLngTxt_ replJS	addLngTxt_ delSusLnk	addLngTxt_ modCpy	addLngTxt_ modTtl	addLngTxt_ addIcn	addLngTxt_ addSusLnk	addLngTxt_ addImgBot	addLngTxt_ modBgClr
CN	F^u	0.96±0.007	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013
	F^r	0.86±0.013	0.61±0.034	0.88±0.012	0.88±0.013	0.88±0.013	0.28±0.008	0.74±0.05	0.91±0.016	0.76±0.045	0.88±0.013	0.82±0.019	0.5±0.043	0.72±0.058	0.88±0.013
	F^c	0.97±0.009	0.97±0.021	0.97±0.016	0.98±0.013	0.98±0.013	0.92±0.024	0.95±0.015	0.96±0.012	0.98±0.013	0.98±0.013	0.96±0.013	0.97±0.02	0.96±0.013	0.98±0.013
RF	F^u	0.96±0.007	0.97±0.007	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005
	F^r	0.90±0.013	0.95±0.018	0.84±0.01	0.9±0.013	0.9±0.013	0.03±0.064	0.71±0.016	0.9±0.015	0.89±0.014	0.9±0.013	0.84±0.025	0.73±0.043	0.78±0.044	0.9±0.013
	F^c	0.97±0.009	0.98±0.008	0.98±0.004	0.98±0.005	0.98±0.004	0.96±0.006	0.96±0.007	0.96±0.007	0.98±0.005	0.98±0.004	0.98±0.004	0.97±0.006	0.98±0.006	0.98±0.004
LR	F^u	0.97±0.005	0.97±0.009	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004
	F^r	0.80±0.013	0.96±0.005	0.79±0.014	0.8±0.013	0.8±0.013	0.24±0.019	0.46±0.013	0.74±0.017	0.8±0.013	0.79±0.013	0.88±0.013	0.78±0.0	0.88±0.021	0.8±0.013
	F^c	0.98±0.005	0.99±0.004	0.98±0.002	0.98±0.002	0.98±0.001	0.96±0.008	0.95±0.007	0.95±0.005	0.98±0.005	0.98±0.005	0.98±0.001	0.97±0.0	0.97±0.005	0.98±0.005

(b) Impact of PA^r+WA^r on ML-PWD generated on Zenodo. PA^r is *delFtr*.

$\mathcal{A}$	F	no-atk	delFtr_ addInLnk	delFtr_ delHidIt	delFtr_ replOnfoc	delFtr_ replPass	delFtr_ addHidP	delFtr_ replJS	delFtr_ delSusLnk	delFtr_ modCpy	delFtr_ modTtl	delFtr_ addIcn	delFtr_ addSusLnk	delFtr_ modBgClr
CN	F^u	0.96±0.007	0.96±0.013	0.96±0.013	0.96±0.013	0.94±0.03	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013
	F^r	0.86±0.013	0.33±0.053	0.64±0.052	0.64±0.052	0.54±0.049	0.54±0.049	0.64±0.052	0.85±0.057	0.63±0.05	0.64±0.052	0.55±0.055	0.28±0.02	0.64±0.052
	F^c	0.97±0.009	0.96±0.028	0.96±0.016	0.96±0.017	0.83±0.035	0.91±0.025	0.94±0.012	0.96±0.011	0.96±0.017	0.96±0.017	0.95±0.015	0.96±0.027	0.96±0.017
RF	F^u	0.96±0.007	0.98±0.005	0.98±0.005	0.98±0.005	0.96±0.007	0.96±0.007	0.98±0.005	0.98±0.005	0.98±0.003	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005
	F^r	0.90±0.013	0.95±0.016	0.8±0.024	0.86±0.018	0.15±0.075	0.15±0.075	0.63±0.054	0.82±0.024	0.86±0.019	0.86±0.018	0.82±0.031	0.59±0.07	0.86±0.018
	F^c	0.97±0.009	0.98±0.008	0.97±0.006	0.97±0.007	0.94±0.018	0.94±0.018	0.96±0.006	0.96±0.007	0.98±0.006	0.97±0.006	0.97±0.007	0.96±0.006	0.97±0.006
LR	F^u	0.97±0.005	0.97±0.004	0.97±0.004	0.97±0.004	0.96±0.0	0.96±0.0	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.009	0.97±0.004
	F^r	0.80±0.013	0.97±0.0	0.75±0.006	0.76±0.008	0.37±0.01	0.37±0.01	0.45±0.008	0.74±0.011	0.76±0.008	0.75±0.008	0.86±0.007	0.72±0.0	0.76±0.008
	F^c	0.98±0.005	0.98±0.007	0.97±0.007	0.97±0.008	0.96±0.007	0.96±0.007	0.96±0.004	0.95±0.005	0.98±0.004	0.98±0.002	0.98±0.004	0.97±0.004	0.97±0.008

(c) Impact of PA^r+WA^r on ML-PWD generated on Zenodo. PA^r is *delFrm*.

$\mathcal{A}$	F	no-atk	delFrm_ addInLnk	delFrm_ onclick	delFrm_ delHidIt	delFrm_ addHidP	delFrm_ replJS	delFrm_ delSusLnk	delFrm_ addImgBot	delFrm_ modFntSiz	delFrm_ modBgimg	delFrm_ modBgClr	delFrm_ delCpy	delFrm_ modTtl	delFrm_ modCpy	delFrm_ addIcn	delFrm_ replRet
CN	F^u	0.96±0.007	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.94±0.018	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013
	F^r	0.86±0.013	0.56±0.043	0.87±0.024	0.84±0.038	0.27±0.008	0.76±0.054	0.87±0.028	0.75±0.052	0.84±0.038	0.84±0.038	0.84±0.038	0.78±0.064	0.84±0.038	0.78±0.064	0.82±0.042	0.84±0.038
	F^c	0.97±0.009	0.96±0.025	0.96±0.008	0.97±0.015	0.92±0.022	0.91±0.027	0.96±0.011	0.96±0.018	0.97±0.016	0.97±0.016	0.97±0.016	0.97±0.016	0.97±0.016	0.97±0.016	0.96±0.013	0.97±0.016
RF	F^u	0.96±0.007	0.98±0.005	0.98±0.003	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.003	0.98±0.005	0.98±0.005
	F^r	0.90±0.013	0.9±0.043	0.8±0.031	0.81±0.028	0.01±0.045	0.58±0.05	0.79±0.027	0.69±0.065	0.82±0.023	0.82±0.023	0.82±0.023	0.72±0.041	0.82±0.023	0.72±0.041	0.77±0.035	0.82±0.023
	F^c	0.97±0.009	0.98±0.007	0.97±0.009	0.98±0.005	0.96±0.005	0.96±0.006	0.96±0.007	0.97±0.006	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.99±0.006	0.98±0.005	0.98±0.005
LR	F^u	0.97±0.005	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.009	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004
	F^r	0.80±0.013	0.93±0.009	0.6±0.024	0.76±0.002	0.24±0.019	0.36±0.005	0.69±0.011	0.87±0.027	0.77±0.005	0.77±0.005	0.77±0.005	0.77±0.005	0.76±0.005	0.77±0.005	0.88±0.008	0.77±0.005
	F^c	0.98±0.005	0.98±0.0	0.97±0.005	0.98±0.005	0.97±0.008	0.96±0.007	0.95±0.005	0.97±0.005	0.97±0.008	0.98±0.002	0.98±0.001	0.98±0.005	0.98±0.005	0.98±0.001	0.97±0.005	0.98±0.005

(d) Impact of PA^r+WA^r on ML-PWD generated on Zenodo. PA^r is *delSpn*.

$\mathcal{A}$	F	no-atk	delSpn_ addInLnk	delSpn_ replOnfoc	delSpn_ replPass	delSpn_ addHidP	delSpn_ replJS	delSpn_ delSusLnk	delSpn_ modCpy	delSpn_ modTtl	delSpn_ addIcn	delSpn_ addSusLnk	delSpn_ addImgBot	delSpn_ modBgClr
CN	F^u	0.96±0.007	0.94±0.021	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013	0.96±0.013
	F^r	0.86±0.013	0.63±0.025	0.88±0.015	0.88±0.014	0.27±0.008	0.74±0.052	0.89±0.019	0.76±0.044	0.88±0.015	0.83±0.02	0.4±0.02	0.75±0.062	0.88±0.015
	F^c	0.97±0.009	0.95±0.02	0.98±0.012	0.98±0.011	0.92±0.024	0.95±0.015	0.96±0.012	0.98±0.012	0.98±0.012	0.96±0.013	0.95±0.019	0.97±0.011	0.98±0.012
RF	F^u	0.96±0.007	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005	0.98±0.005
	F^r	0.90±0.013	0.95±0.021	0.9±0.012	0.9±0.012	0.03±0.068	0.66±0.016	0.89±0.016	0.9±0.013	0.9±0.012	0.85±0.027	0.58±0.043	0.8±0.051	0.9±0.012
	F^c	0.97±0.009	0.98±0.007	0.98±0.004	0.98±0.004	0.96±0.006	0.96±0.006	0.96±0.006	0.98±0.004	0.98±0.004	0.98±0.004	0.97±0.007	0.98±0.005	0.98±0.004
LR	F^u	0.97±0.005	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004	0.97±0.004
	F^r	0.80±0.013	0.96±0.005	0.8±0.013	0.8±0.013	0.23±0.017	0.41±0.013	0.74±0.017	0.8±0.013	0.79±0.013	0.88±0.013	0.62±0.0	0.9±0.021	0.8±0.013
	F^c	0.98±0.005	0.98±0.003	0.98±0.005	0.98±0.005	0.97±0.009	0.96±0.004	0.96±0.002	0.98±0.0	0.98±0.001	0.98±0.001	0.97±0.0	0.98±0.003	0.98±0.0

Table 23. Impact of PA^r+MA^r on Zenodo.

$\mathcal{A}$	F	no-atk	addLngTxt_delBdy	delFtr_delBdy	delSpn_delBdy
CN	F^u	$0.96_{\pm 0.007}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$
	F^r	$0.86_{\pm 0.013}$	$0.88_{\pm 0.013}$	$0.64_{\pm 0.052}$	$0.88_{\pm 0.015}$
	F^c	$0.97_{\pm 0.009}$	$0.98_{\pm 0.013}$	$0.96_{\pm 0.017}$	$0.98_{\pm 0.012}$
RF	F^u	$0.96_{\pm 0.007}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$
	F^r	$0.90_{\pm 0.013}$	$0.9_{\pm 0.013}$	$0.86_{\pm 0.018}$	$0.9_{\pm 0.012}$
	F^c	$0.97_{\pm 0.009}$	$0.98_{\pm 0.005}$	$0.97_{\pm 0.006}$	$0.98_{\pm 0.005}$
LR	F^u	$0.97_{\pm 0.005}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$
	F^r	$0.80_{\pm 0.013}$	$0.8_{\pm 0.013}$	$0.76_{\pm 0.008}$	$0.8_{\pm 0.013}$
	F^c	$0.98_{\pm 0.005}$	$0.98_{\pm 0.002}$	$0.98_{\pm 0.004}$	$0.98_{\pm 0.001}$

Table 24. Impact of WA^r+WA^r on Zenodo.

$\mathcal{A}$	F	no-atk	replOnfoc_replRet	htEsc_replRet	htEncd_replRet
CN	F^u	$0.96_{\pm 0.007}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$
	F^r	$0.86_{\pm 0.013}$	$0.88_{\pm 0.013}$	$0.87_{\pm 0.025}$	$0.0_{\pm 0.0}$
	F^c	$0.97_{\pm 0.009}$	$0.98_{\pm 0.013}$	$0.96_{\pm 0.007}$	$0.91_{\pm 0.032}$
RF	F^u	$0.96_{\pm 0.007}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$
	F^r	$0.90_{\pm 0.013}$	$0.9_{\pm 0.013}$	$0.84_{\pm 0.027}$	$0.0_{\pm 0.0}$
	F^c	$0.97_{\pm 0.009}$	$0.98_{\pm 0.005}$	$0.96_{\pm 0.007}$	$0.96_{\pm 0.007}$
LR	F^u	$0.97_{\pm 0.005}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$
	F^r	$0.80_{\pm 0.013}$	$0.8_{\pm 0.013}$	$0.55_{\pm 0.009}$	$0.0_{\pm 0.0}$
	F^c	$0.98_{\pm 0.005}$	$0.97_{\pm 0.005}$	$0.97_{\pm 0.001}$	$0.97_{\pm 0.004}$

Table 25. **Impact of** PA^r **and** MA^r **on** Zenodo**.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific PsP or MsP attack.

$\mathcal{A}$	F	no-atk	PsP							MsP					
			delTxt	delFrm	delSpn	delTtl	addLngTxt	delFtr	replSusFtrLnk	brTg	delHt	delHd	delBdy	brTgs	hmg
CN	F^u	$0.96_{\pm 0.007}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.97_{\pm 0.014}$
	F^r	$0.86_{\pm 0.013}$	$0.78_{\pm 0.046}$	$0.86_{\pm 0.036}$	$0.88_{\pm 0.015}$	$0.88_{\pm 0.013}$	$0.88_{\pm 0.013}$	$0.64_{\pm 0.052}$	$0.74_{\pm 0.05}$	$0.88_{\pm 0.013}$	$0.0_{\pm 0.0}$	$0.82_{\pm 0.011}$	$0.43_{\pm 0.049}$	$0.88_{\pm 0.013}$	$0.4_{\pm 0.035}$
	F^c	$0.97_{\pm 0.009}$	$0.97_{\pm 0.012}$	$0.97_{\pm 0.012}$	$0.97_{\pm 0.012}$	$0.98_{\pm 0.011}$	$0.97_{\pm 0.012}$	$0.96_{\pm 0.017}$	$0.96_{\pm 0.012}$	$0.97_{\pm 0.012}$	$0.9_{\pm 0.033}$	$0.95_{\pm 0.02}$	$0.93_{\pm 0.019}$	$0.97_{\pm 0.012}$	$0.92_{\pm 0.026}$
RF	F^u	$0.96_{\pm 0.007}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.97_{\pm 0.007}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.006}$
	F^r	$0.90_{\pm 0.013}$	$0.86_{\pm 0.032}$	$0.9_{\pm 0.014}$	$0.9_{\pm 0.012}$	$0.86_{\pm 0.032}$	$0.9_{\pm 0.013}$	$0.86_{\pm 0.018}$	$0.84_{\pm 0.012}$	$0.9_{\pm 0.013}$	$0.0_{\pm 0.0}$	$0.66_{\pm 0.083}$	$0.46_{\pm 0.024}$	$0.9_{\pm 0.013}$	$0.22_{\pm 0.066}$
	F^c	$0.97_{\pm 0.009}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.004}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.004}$	$0.97_{\pm 0.006}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.006}$	$0.96_{\pm 0.006}$	$0.98_{\pm 0.005}$	$0.96_{\pm 0.006}$	$0.98_{\pm 0.005}$	$0.97_{\pm 0.007}$
LR	F^u	$0.97_{\pm 0.005}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.009}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.009}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.98_{\pm 0.005}$
	F^r	$0.80_{\pm 0.013}$	$0.66_{\pm 0.004}$	$0.8_{\pm 0.012}$	$0.8_{\pm 0.013}$	$0.66_{\pm 0.004}$	$0.8_{\pm 0.013}$	$0.76_{\pm 0.008}$	$0.73_{\pm 0.022}$	$0.8_{\pm 0.013}$	$0.0_{\pm 0.0}$	$0.74_{\pm 0.0}$	$0.32_{\pm 0.006}$	$0.8_{\pm 0.013}$	$0.24_{\pm 0.02}$
	F^c	$0.98_{\pm 0.005}$	$0.96_{\pm 0.005}$	$0.97_{\pm 0.008}$	$0.98_{\pm 0.002}$	$0.96_{\pm 0.007}$	$0.98_{\pm 0.002}$	$0.98_{\pm 0.001}$	$0.97_{\pm 0.006}$	$0.97_{\pm 0.005}$	$0.96_{\pm 0.006}$	$0.97_{\pm 0.005}$	$0.97_{\pm 0.003}$	$0.98_{\pm 0.001}$	$0.96_{\pm 0.005}$

Table 26. Impact of PA^r+PA^r on Zenodo

$\mathcal{A}$	F	no-atk	addLngTxt_delTtl	delFtr_delTtl	delFtr_addLngTxt	delSpn_delTtl	delSpn_delFtrer	delSpn_addLngTxt	delFrm_delFtr	delFrm_delSpn
CN	F^u	$0.96_{\pm 0.007}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$	$0.96_{\pm 0.013}$
	F^r	$0.86_{\pm 0.013}$	$0.88_{\pm 0.013}$	$0.64_{\pm 0.052}$	$0.64_{\pm 0.052}$	$0.88_{\pm 0.015}$	$0.64_{\pm 0.052}$	$0.88_{\pm 0.015}$	$0.78_{\pm 0.064}$	$0.83_{\pm 0.045}$
	F^c	$0.97_{\pm 0.009}$	$0.98_{\pm 0.011}$	$0.96_{\pm 0.015}$	$0.96_{\pm 0.017}$	$0.98_{\pm 0.011}$	$0.96_{\pm 0.016}$	$0.98_{\pm 0.012}$	$0.96_{\pm 0.015}$	$0.97_{\pm 0.015}$
RF	F^u	$0.96_{\pm 0.007}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$
	F^r	$0.90_{\pm 0.013}$	$0.86_{\pm 0.032}$	$0.8_{\pm 0.042}$	$0.86_{\pm 0.018}$	$0.87_{\pm 0.025}$	$0.86_{\pm 0.018}$	$0.9_{\pm 0.012}$	$0.7_{\pm 0.052}$	$0.79_{\pm 0.034}$
	F^c	$0.97_{\pm 0.009}$	$0.98_{\pm 0.004}$	$0.97_{\pm 0.007}$	$0.97_{\pm 0.006}$	$0.98_{\pm 0.005}$	$0.97_{\pm 0.007}$	$0.98_{\pm 0.005}$	$0.97_{\pm 0.007}$	$0.98_{\pm 0.005}$
LR	F^u	$0.97_{\pm 0.005}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$	$0.97_{\pm 0.004}$
	F^r	$0.80_{\pm 0.013}$	$0.66_{\pm 0.004}$	$0.58_{\pm 0.005}$	$0.76_{\pm 0.008}$	$0.65_{\pm 0.004}$	$0.76_{\pm 0.008}$	$0.8_{\pm 0.013}$	$0.7_{\pm 0.009}$	$0.76_{\pm 0.005}$
	F^c	$0.98_{\pm 0.005}$	$0.96_{\pm 0.005}$	$0.97_{\pm 0.01}$	$0.98_{\pm 0.004}$	$0.96_{\pm 0.007}$	$0.97_{\pm 0.005}$	$0.97_{\pm 0.008}$	$0.98_{\pm 0.005}$	$0.98_{\pm 0.005}$

C.3 Perturbation's impact on MLSEC

We executed 37 kinds of single attacks and report the influence of MLSEC's PWD in Table 27, 28 and 29, and the influence of hybrid space attacks in Table 33, 32, 30 and 31.

Table 27. **Impact of iWA^r on the PWD of MLSEC.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific iWsP perturbation.

$\mathcal{A}$	no-atk	addInLnk	replOnc	delHidIt	addHidP	replJS	replRet	htEsc	htEncd	replPass	replOnfoc	addSusLnk
$m0$	0.91±0.052	0.73±0.159	0.91±0.052	0.88±0.049	0.02±0.011	0.9±0.059	0.79±0.123	0.06±0.046	0.06±0.036	0.48±0.259	0.5±0.254	0.5±0.203
$m1$	0.87±0.071	0.86±0.083	0.88±0.07	0.85±0.093	0.52±0.161	0.85±0.088	0.74±0.115	0.37±0.113	0.41±0.126	0.85±0.068	0.84±0.09	0.85±0.1
$m2$	0.9±0.051	0.73±0.158	0.9±0.052	0.88±0.052	0.02±0.011	0.9±0.058	0.83±0.105	0.85±0.127	0.9±0.051	0.47±0.266	0.51±0.263	0.5±0.202
$m3$	0.88±0.07	0.86±0.08	0.87±0.07	0.85±0.096	0.51±0.172	0.85±0.087	0.79±0.108	0.86±0.099	0.88±0.07	0.85±0.066	0.85±0.093	0.85±0.1
$m4$	0.82±0.106	0.83±0.108	0.82±0.111	0.8±0.136	0.83±0.123	0.82±0.126	0.69±0.122	0.09±0.067	0.07±0.045	0.46±0.256	0.46±0.242	0.47±0.171
$m5$	0.81±0.12	0.82±0.12	0.81±0.124	0.8±0.141	0.82±0.136	0.81±0.136	0.67±0.114	0.43±0.098	0.46±0.139	0.79±0.125	0.79±0.119	0.8±0.157
$m6$	0.83±0.108	0.83±0.11	0.83±0.112	0.81±0.131	0.84±0.116	0.82±0.127	0.73±0.148	0.84±0.126	0.83±0.108	0.47±0.256	0.49±0.236	0.47±0.174
$m7$	0.82±0.121	0.82±0.122	0.82±0.126	0.81±0.136	0.83±0.127	0.81±0.138	0.7±0.145	0.84±0.121	0.82±0.121	0.79±0.126	0.81±0.125	0.8±0.157

Table 28. **Evasion Robustness of the MLSEC's PWD against eWA^r and rWA^r.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the PWD, while columns correspond to a specific eWsP or rWsP attack.

$\mathcal{A}$	no-atk	eWsP								rWsP			
		addImgBot	modFntTyp	modCpy	addIcn	delSusLnk	delSusFrm	modTtl	delCpy	modBgimg	modBgClr	modFntClr	modFntSiz
$m0$	0.91±0.052	0.65±0.185	0.92±0.051	0.89±0.061	0.76±0.126	0.86±0.095	0.8±0.253	0.87±0.102	0.89±0.061	0.92±0.053	0.91±0.056	0.81±0.116	0.92±0.051
$m1$	0.87±0.071	0.87±0.085	0.89±0.063	0.85±0.091	0.77±0.089	0.82±0.146	0.81±0.106	0.84±0.12	0.85±0.089	0.89±0.064	0.88±0.077	0.78±0.1	0.89±0.063
$m2$	0.9±0.051	0.65±0.185	0.91±0.05	0.89±0.06	0.76±0.122	0.86±0.095	0.79±0.262	0.87±0.101	0.89±0.06	0.91±0.052	0.91±0.055	0.85±0.087	0.91±0.05
$m3$	0.88±0.07	0.87±0.079	0.89±0.064	0.85±0.09	0.77±0.081	0.82±0.146	0.8±0.124	0.84±0.119	0.85±0.088	0.89±0.066	0.88±0.076	0.81±0.091	0.89±0.064
$m4$	0.82±0.106	0.64±0.199	0.87±0.065	0.81±0.124	0.83±0.109	0.8±0.156	0.73±0.257	0.8±0.156	0.81±0.127	0.87±0.066	0.86±0.079	0.73±0.112	0.87±0.065
$m5$	0.81±0.12	0.85±0.107	0.85±0.089	0.8±0.136	0.82±0.122	0.79±0.145	0.78±0.14	0.79±0.167	0.8±0.137	0.85±0.089	0.84±0.096	0.7±0.103	0.85±0.089
$m6$	0.83±0.108	0.64±0.198	0.87±0.066	0.81±0.126	0.83±0.109	0.8±0.157	0.72±0.261	0.79±0.157	0.81±0.128	0.87±0.066	0.86±0.079	0.73±0.111	0.87±0.065
$m7$	0.82±0.121	0.85±0.106	0.85±0.089	0.8±0.138	0.82±0.123	0.79±0.146	0.78±0.141	0.79±0.169	0.81±0.138	0.85±0.089	0.84±0.097	0.7±0.097	0.85±0.089

Table 29. **Impact of PA^r and MA^r on PWD of MLSEC.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific PsP or MsP attack.

$\mathcal{A}$	no-atk	PsP							MsP					
		delTxt	delFrm	delSpn	delTtl	addLngTxt	delFtr	replSusFtrLnk	brTg	delHt	delHd	delBdy	brTgs	hmg
$m0$	0.91±0.052	0.64±0.272	0.91±0.052	0.85±0.089	0.88±0.062	0.86±0.095	0.87±0.085	0.9±0.059	0.9±0.062	0.9±0.062	0.9±0.062	0.9±0.062	0.9±0.062	0.9±0.062
$m1$	0.87±0.071	0.83±0.133	0.67±0.262	0.67±0.262	0.82±0.117	0.83±0.133	0.82±0.117	0.67±0.262	0.31±0.051	0.31±0.051	0.31±0.051	0.31±0.051	0.31±0.051	0.31±0.051
$m2$	0.9±0.051	0.84±0.148	0.61±0.39	0.61±0.39	0.85±0.087	0.84±0.148	0.85±0.087	0.61±0.39	0.88±0.096	0.88±0.096	0.88±0.096	0.88±0.096	0.88±0.096	0.88±0.096
$m3$	0.88±0.07	0.83±0.131	0.66±0.271	0.66±0.271	0.82±0.115	0.83±0.131	0.82±0.115	0.66±0.271	0.26±0.08	0.26±0.08	0.26±0.08	0.26±0.08	0.26±0.08	0.26±0.08
$m4$	0.82±0.106	0.8±0.169	0.57±0.372	0.57±0.372	0.79±0.149	0.8±0.169	0.79±0.149	0.57±0.372	0.8±0.121	0.8±0.121	0.8±0.121	0.8±0.121	0.8±0.121	0.8±0.121
$m5$	0.81±0.12	0.79±0.16	0.64±0.28	0.64±0.28	0.79±0.143	0.79±0.16	0.79±0.143	0.64±0.28	0.39±0.166	0.39±0.166	0.39±0.166	0.39±0.166	0.39±0.166	0.39±0.166
$m6$	0.83±0.108	0.8±0.17	0.56±0.373	0.56±0.373	0.79±0.144	0.8±0.17	0.79±0.144	0.56±0.373	0.07±0.076	0.07±0.076	0.07±0.076	0.07±0.076	0.07±0.076	0.07±0.076
$m7$	0.82±0.121	0.79±0.161	0.64±0.279	0.64±0.279	0.79±0.138	0.79±0.161	0.79±0.138	0.64±0.279	0.36±0.129	0.36±0.129	0.36±0.129	0.36±0.129	0.36±0.129	0.36±0.129

Table 30. **Impact of PA^r+WA^r on PWD of MLSEC.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the PWD, while columns correspond to a specific PsP+WsP perturbation.

(a) Impact of PA^r+WA^r on PWD of MLSEC. PA^r is *delFrm*.

$\mathcal{A}$	no-atk	delFrm_ addInLnk	delFrm_ onclick	delFrm_ delHidIt	delFrm_ addHidP	delFrm_ replJS	delFrm_ delSusLnk	delFrm_ addImgBot	delFrm_ modFntSiz	delFrm_ modBgimg	delFrm_ modBgClr	delFrm_ delCpy	delFrm_ modTtl	delFrm_ modCpy	delFrm_ addIcn	delFrm_ replRet	delFrm_ combine
$m0$	0.91±0.052	0.49±0.335	0.61±0.388	0.6±0.384	0.01±0.008	0.61±0.388	0.59±0.379	0.43±0.325	0.63±0.388	0.63±0.388	0.63±0.387	0.61±0.387	0.59±0.383	0.61±0.386	0.49±0.329	0.51±0.341	0.01±0.008
$m1$	0.87±0.071	0.66±0.268	0.67±0.262	0.63±0.284	0.41±0.181	0.67±0.262	0.62±0.291	0.69±0.243	0.69±0.265	0.69±0.265	0.69±0.265	0.67±0.261	0.65±0.27	0.66±0.261	0.55±0.254	0.53±0.257	0.37±0.196
$m2$	0.9±0.051	0.49±0.338	0.61±0.39	0.61±0.386	0.01±0.008	0.61±0.39	0.59±0.38	0.43±0.326	0.63±0.388	0.63±0.388	0.63±0.387	0.61±0.389	0.59±0.385	0.61±0.388	0.49±0.33	0.57±0.36	0.01±0.008
$m3$	0.88±0.07	0.66±0.277	0.66±0.271	0.63±0.289	0.38±0.196	0.66±0.271	0.61±0.299	0.68±0.253	0.69±0.266	0.69±0.266	0.69±0.266	0.66±0.269	0.64±0.278	0.66±0.269	0.55±0.257	0.6±0.253	0.37±0.198
$m4$	0.82±0.106	0.57±0.372	0.56±0.372	0.55±0.378	0.57±0.379	0.56±0.372	0.56±0.378	0.42±0.324	0.6±0.377	0.6±0.377	0.6±0.377	0.57±0.366	0.55±0.375	0.56±0.371	0.56±0.372	0.47±0.314	0.54±0.388
$m5$	0.81±0.12	0.64±0.28	0.63±0.28	0.61±0.3	0.65±0.28	0.64±0.28	0.62±0.293	0.68±0.241	0.67±0.277	0.67±0.277	0.67±0.277	0.65±0.261	0.61±0.298	0.63±0.279	0.63±0.281	0.5±0.257	0.6±0.319
$m6$	0.83±0.108	0.56±0.373	0.56±0.372	0.55±0.378	0.57±0.379	0.56±0.373	0.56±0.377	0.42±0.328	0.6±0.376	0.6±0.376	0.6±0.376	0.57±0.367	0.55±0.376	0.56±0.371	0.56±0.372	0.47±0.314	0.55±0.392
$m7$	0.82±0.121	0.63±0.28	0.64±0.279	0.62±0.294	0.66±0.267	0.64±0.279	0.63±0.282	0.69±0.237	0.67±0.276	0.67±0.276	0.67±0.276	0.65±0.26	0.62±0.297	0.64±0.277	0.63±0.279	0.51±0.254	0.62±0.303

(b) Impact of PA^r+WA^r on PWD of MLSEC. PA^r is *delSpn*.

$\mathcal{A}$	no-atk	delSpn_ addInLnk	delSpn_ replOnfoc	delSpn_ replPass	delSpn_ addHidP	delSpn_ replJS	delSpn_ delSusLnk	delSpn_ modCpy	delSpn_ modTtl	delSpn_ addIcn	delSpn_ addSusLnk	delSpn_ addImgBot	delSpn_ modBgClr	delSpn_ combine
$m0$	0.91±0.052	0.73±0.137	0.41±0.28	0.39±0.277	0.02±0.007	0.85±0.087	0.84±0.117	0.85±0.091	0.83±0.12	0.7±0.127	0.68±0.195	0.6±0.199	0.88±0.075	0.0±0.004
$m1$	0.87±0.071	0.83±0.112	0.78±0.129	0.8±0.116	0.52±0.149	0.82±0.115	0.79±0.165	0.82±0.119	0.8±0.14	0.7±0.1	0.83±0.117	0.85±0.097	0.86±0.096	0.55±0.106
$m2$	0.9±0.051	0.73±0.131	0.42±0.287	0.38±0.282	0.02±0.008	0.85±0.085	0.84±0.117	0.85±0.089	0.83±0.118	0.7±0.125	0.68±0.194	0.59±0.199	0.88±0.073	0.0±0.003
$m3$	0.88±0.07	0.83±0.104	0.79±0.129	0.79±0.111	0.51±0.17	0.82±0.113	0.79±0.164	0.82±0.117	0.8±0.137	0.7±0.097	0.82±0.114	0.84±0.094	0.86±0.094	0.52±0.125
$m4$	0.82±0.106	0.79±0.149	0.43±0.248	0.43±0.261	0.8±0.149	0.79±0.141	0.77±0.168	0.78±0.162	0.76±0.179	0.78±0.153	0.65±0.175	0.6±0.215	0.84±0.088	0.5±0.263
$m5$	0.81±0.12	0.79±0.143	0.77±0.137	0.76±0.151	0.81±0.142	0.8±0.137	0.77±0.142	0.79±0.153	0.77±0.181	0.79±0.149	0.8±0.145	0.84±0.116	0.84±0.097	0.84±0.112
$m6$	0.83±0.108	0.78±0.152	0.44±0.253	0.43±0.262	0.8±0.15	0.79±0.147	0.77±0.167	0.78±0.162	0.76±0.176	0.79±0.148	0.66±0.178	0.6±0.215	0.84±0.09	0.53±0.275
$m7$	0.82±0.121	0.79±0.148	0.78±0.136	0.76±0.145	0.81±0.142	0.79±0.14	0.77±0.14	0.78±0.152	0.77±0.177	0.79±0.144	0.79±0.143	0.84±0.113	0.83±0.098	0.83±0.116

(c) Impact of PA^r+WA^r on PWD of MLSEC. PA^r is *delFtr*.

$\mathcal{A}$	no-atk	delFtr_ addInLnk	delFtr_ delHidIt	delFtr_ replOnfoc	delFtr_ replPass	delFtr_ addHidP	delFtr_ replJS	delFtr_ delSusLnk	delFtr_ modCpy	delFtr_ modTtl	delFtr_ addicon	delFtr_ addSusLnk	delFtr_ addImgBot	delFtr_ modBgClr	delFtr_ combine
$m0$	0.91±0.052	0.77±0.132	0.77±0.241	0.49±0.197	0.51±0.196	0.02±0.009	0.87±0.085	0.84±0.118	0.87±0.085	0.85±0.099	0.72±0.148	0.71±0.194	0.67±0.187	0.89±0.071	0.0±0.002
$m1$	0.87±0.071	0.85±0.092	0.78±0.165	0.81±0.086	0.83±0.089	0.53±0.148	0.85±0.086	0.81±0.156	0.85±0.086	0.84±0.092	0.73±0.093	0.85±0.099	0.87±0.077	0.87±0.076	0.36±0.096
$m2$	0.9±0.051	0.77±0.131	0.77±0.253	0.48±0.214	0.49±0.215	0.02±0.011	0.87±0.084	0.84±0.117	0.87±0.084	0.85±0.098	0.72±0.147	0.71±0.192	0.67±0.187	0.89±0.07	0.0±0.001
$m3$	0.88±0.07	0.85±0.092	0.78±0.18	0.82±0.094	0.82±0.091	0.51±0.173	0.85±0.085	0.81±0.155	0.85±0.085	0.84±0.09	0.73±0.092	0.85±0.096	0.87±0.076	0.87±0.075	0.33±0.11
$m4$	0.82±0.106	0.79±0.154	0.71±0.263	0.5±0.187	0.54±0.174	0.81±0.145	0.79±0.143	0.77±0.171	0.79±0.143	0.79±0.15	0.79±0.154	0.67±0.177	0.65±0.191	0.85±0.072	0.57±0.193
$m5$	0.81±0.12	0.81±0.117	0.78±0.164	0.77±0.137	0.78±0.138	0.83±0.109	0.82±0.107	0.79±0.134	0.82±0.107	0.81±0.115	0.81±0.117	0.8±0.146	0.86±0.084	0.86±0.069	0.84±0.116
$m6$	0.83±0.108	0.79±0.153	0.71±0.263	0.5±0.206	0.53±0.196	0.81±0.146	0.78±0.153	0.77±0.171	0.78±0.153	0.78±0.147	0.78±0.153	0.67±0.184	0.65±0.194	0.85±0.077	0.59±0.22
$m7$	0.82±0.121	0.81±0.116	0.78±0.156	0.78±0.142	0.78±0.136	0.84±0.109	0.81±0.116	0.79±0.134	0.81±0.116	0.81±0.112	0.81±0.116	0.8±0.148	0.86±0.084	0.85±0.075	0.84±0.118

(d) Impact of PA^r+WA^r on PWD of MLSEC. PA^r is *addLngTxt*.

$\mathcal{A}$	no-atk	addLngTxt_ addInLnk	addLngTxt_ delHidIt	addLngTxt_ replOnfoc	addLngTxt_ replPass	addLngTxt_ addHidP	addLngTxt_ replJS	addLngTxt_ delSusLnk	addLngTxt_ modCpy	addLngTxt_ modTtl	addLngTxt_ addIcn	addLngTxt_ addSusLnk	addLngTxt_ addImgBot	addLngTxt_ modBgClr	addLngTxt_ combine
$m0$	0.91±0.052	0.69±0.214	0.73±0.277	0.41±0.245	0.42±0.272	0.02±0.011	0.83±0.158	0.68±0.218	0.83±0.158	0.83±0.158	0.67±0.187	0.68±0.218	0.59±0.227	0.84±0.157	0.0±0.004
$m1$	0.87±0.071	0.85±0.105	0.73±0.216	0.8±0.144	0.82±0.116	0.5±0.189	0.82±0.141	0.86±0.103	0.82±0.141	0.82±0.141	0.69±0.136	0.86±0.103	0.84±0.126	0.84±0.131	0.48±0.139
$m2$	0.9±0.051	0.69±0.213	0.73±0.281	0.42±0.25	0.41±0.275	0.02±0.011	0.83±0.157	0.68±0.215	0.83±0.157	0.83±0.157	0.66±0.189	0.68±0.215	0.58±0.227	0.84±0.156	0.0±0.004
$m3$	0.88±0.07	0.86±0.102	0.73±0.228	0.81±0.149	0.82±0.114	0.5±0.186	0.82±0.14	0.86±0.101	0.82±0.14	0.82±0.14	0.69±0.136	0.86±0.101	0.83±0.125	0.84±0.13	0.48±0.152
$m4$	0.82±0.106	0.8±0.169	0.68±0.3	0.43±0.236	0.46±0.26	0.81±0.166	0.78±0.18	0.72±0.126	0.79±0.179	0.79±0.179	0.8±0.169	0.72±0.126	0.59±0.231	0.84±0.116	0.42±0.252
$m5$	0.81±0.12	0.79±0.16	0.71±0.252	0.77±0.152	0.77±0.158	0.81±0.156	0.78±0.173	0.82±0.125	0.79±0.17	0.79±0.17	0.79±0.16	0.82±0.125	0.82±0.134	0.82±0.122	0.69±0.251
$m6$	0.83±0.108	0.8±0.169	0.68±0.3	0.45±0.238	0.46±0.263	0.81±0.167	0.78±0.179	0.72±0.128	0.79±0.178	0.79±0.178	0.8±0.17	0.72±0.128	0.59±0.23	0.83±0.115	0.48±0.259
$m7$	0.82±0.121	0.79±0.161	0.71±0.248	0.78±0.163	0.77±0.159	0.81±0.159	0.78±0.172	0.82±0.126	0.79±0.169	0.79±0.169	0.8±0.162	0.82±0.126	0.82±0.134	0.82±0.119	0.72±0.245

Table 31. **Impact of WA^r+WA^r on MLSEC.** The cells report the average (and std. dev.) *tpr* over the 50 reiterations. Lines correspond to the ML-PWD, while columns correspond to a specific WsP+WsP.

$\mathcal{A}$	no-atk	replOnfoc_replRet	htEsc_replRet	htEncd_replRet
$m0$	0.91±0.052	0.33±0.165	0.03±0.021	0.08±0.04
$m1$	0.87±0.071	0.63±0.136	0.16±0.065	0.22±0.082
$m2$	0.9±0.051	0.46±0.229	0.86±0.137	0.9±0.051
$m3$	0.88±0.07	0.76±0.13	0.86±0.105	0.88±0.07
$m4$	0.82±0.106	0.31±0.176	0.03±0.017	0.11±0.05
$m5$	0.81±0.12	0.55±0.124	0.15±0.053	0.28±0.093
$m6$	0.83±0.108	0.36±0.221	0.85±0.124	0.83±0.108
$m7$	0.82±0.121	0.62±0.182	0.85±0.109	0.82±0.121

Table 32. **Impact of PA^r+PA^r on MLSEC.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the PWD, while columns correspond to a specific PsP+PsP perturbation.

$\mathcal{A}$	no-atk	addLngTxt_delTtl	delFtr_delTtl	delFtr_addLngTxt	delSpn_delTtl	delSpn_delFtrer	delSpn_addLngTxt	delFrm_delFtr	delFrm_delSpn
$m0$	0.91±0.052	0.82±0.16	0.85±0.092	0.8±0.167	0.84±0.093	0.84±0.096	0.79±0.173	0.59±0.375	0.58±0.375
$m1$	0.87±0.071	0.81±0.145	0.84±0.089	0.81±0.13	0.81±0.117	0.83±0.105	0.79±0.157	0.66±0.269	0.62±0.28
$m2$	0.9±0.051	0.81±0.159	0.85±0.094	0.8±0.166	0.83±0.095	0.84±0.096	0.79±0.172	0.59±0.376	0.58±0.376
$m3$	0.88±0.07	0.81±0.143	0.83±0.093	0.81±0.128	0.81±0.118	0.83±0.106	0.79±0.154	0.65±0.276	0.62±0.288
$m4$	0.82±0.106	0.77±0.178	0.78±0.146	0.75±0.2	0.78±0.145	0.77±0.156	0.76±0.204	0.57±0.358	0.56±0.367
$m5$	0.81±0.12	0.77±0.172	0.8±0.111	0.77±0.157	0.78±0.141	0.8±0.124	0.77±0.191	0.66±0.259	0.63±0.275
$m6$	0.83±0.108	0.77±0.177	0.77±0.155	0.74±0.198	0.77±0.152	0.77±0.154	0.76±0.201	0.56±0.358	0.56±0.367
$m7$	0.82±0.121	0.77±0.17	0.8±0.12	0.77±0.155	0.78±0.146	0.8±0.122	0.77±0.186	0.67±0.255	0.63±0.273

Table 33. **Impact of PA^r+MA^r on MLSEC.** The cells report the average (and std. dev.) tpr over the 50 reiterations. Lines correspond to the PWD, while columns correspond to a specific PsP+MsP perturbation.

$\mathcal{A}$	no-atk	addLngTxt_delBdy	delfoot_delBdy	delSpn_delBdy
$m0$	0.91±0.052	0.78±0.167	0.79±0.137	0.75±0.156
$m1$	0.87±0.071	0.8±0.146	0.81±0.115	0.78±0.14
$m2$	0.9±0.051	0.8±0.161	0.8±0.13	0.77±0.149
$m3$	0.88±0.07	0.83±0.125	0.82±0.108	0.79±0.133
$m4$	0.82±0.106	0.79±0.136	0.76±0.154	0.72±0.184
$m5$	0.81±0.12	0.8±0.139	0.8±0.122	0.77±0.148
$m6$	0.83±0.108	0.79±0.137	0.76±0.154	0.72±0.183
$m7$	0.82±0.121	0.8±0.139	0.8±0.123	0.77±0.145