Journal of Information Security and Applications 100 (2026) 104486

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Adversarial attacks on phishing webpage detectors via heuristic search techniques

Giuseppe Lo Re, Marco Morana *, Giuseppe Rizzo

University of Palermo Department of Engineering, Viale delle Scienze, ed.6, Palermo, 90128, Italy

ARTICLE INFO

Keywords:
Phishing
Adversarial machine learning
Cybersecurity

ABSTRACT

Phishing remains one of the most prevalent cybersecurity threats, endangering users' personal data, financial assets and online privacy. Although Machine Learning Phishing Website Detectors (ML-PWDs) are an effective tool for identifying malicious webpages, recent studies have revealed that these models are vulnerable to adversarial attacks. In this study, we present a new adversarial attack strategy capable of operating in the problem space, which uses heuristic search algorithms, including Beam Search, Simulated Annealing and Monte Carlo Tree Search, to generate adversarial samples that evade state-of-the-art detectors while maintaining visual and functional fidelity. Our approach optimizes the trade-off between the number of manipulations and attack success, minimizing the distance from the original sample. Experiments on two public datasets demonstrate that our method reduces the average detection rate from 0.80 to 0.05 on Zenodo and from 0.82 to 0.03 on δPhish, while requiring up to 70% fewer manipulations than competing attacks. Furthermore, the generated samples remain closer to the originals in the L_0 and L_2 metrics, indicating strong statistical plausibility. These results highlight the effectiveness of our approach in evading ML-PWDs and its potential for evaluating and strengthening the adversarial robustness of real-world detection systems.

1. Introduction

Nowadays, phishing is one of the most prominent cyber threats through which users are deceived and led to perform dangerous actions, e.g., downloading unwanted content or disclosing sensitive information [1]. As phishing techniques have evolved, more and more sophisticated defenses have been proposed, including those based on Machine Learning (ML) algorithms. Machine Learning Phishing Website Detectors (ML-PWDs [2]) enable a proactive detection and exhibit greater adaptability to emerging attack strategies being able to identify subtle patterns in URLs, HTML content, and other features that traditional rule-based systems often ignore. Nevertheless, ML is not free from errors. This is at the basis of Adversarial Machine Learning (AML), which aims at the computation of adversarial samples, i.e., purposely crafted inputs, specifically designed to fool ML models and disrupt their ability to make accurate classifications.

Adversarial attacks can be broadly categorized according to whether the manipulations are applied in the *problem space* (where inputs are perturbed in their natural, observable form), or in the *feature space*, where alterations are made directly to the feature representation adopted by the model. Although feature space attacks are often more computationally efficient and easier to conduct, as they allow the attacker to ignore the properties of the raw input data, they present significant limitations. The most important of these concerns the difficulty of reconstructing a realistic and valid input data from the perturbed feature vector. This process is often computationally expensive and, in many cases, may not provide feasible or interpretable results. As a consequence, while feature space attacks are useful for exploring model vulnerabilities in controlled environments, their practical applicability in real-world is quite limited. Therefore, this work focuses on the attacker's capabilities within the *problem space* [3], which provides a more realistic representation of potential cyber threats. Other works [2,4] addressed this issue by providing techniques capable of altering the raw HTML code, or URL, of a phishing webpage without affecting its visual rendering, which makes the attack *imperceptible* to the end user. However, while the existing literature focuses on minimizing the number of attempts (queries) to deceive the target model, it often overlooks the fact that the number of manipulations required may be excessive, thus reducing the efficiency and effectiveness of the attack. To address this issue, we propose an attack that exploits heuristic search strategies to efficiently explore the space of potential manipulations. The main contributions are summarized as follows:

* Corresponding author.
E-mail address: marco.morana@unipa.it (M. Morana).

https://doi.org/10.1016/j.jisa.2026.104486

Available online 21 April 2026

- We introduce a set of black-box, heuristic-search-based optimization algorithms for crafting adversarial HTML pages, balancing attack effectiveness with minimal modifications to original samples.
- Our study includes a comprehensive comparative analysis that evaluates the proposed approach against related research.
- We investigate how reducing the number of modifications impacts the feature-space distances (L_0, L_2, L_∞) between generated and original samples, providing insight into trade-offs between different attack strategies.
- Since our attack operates in the problem space, we provide an analysis of the manipulations introduced by the algorithms considered, discussing also their potential to make the attack successful.
- In order to better understand the relationship between the applied manipulations and the ML-PWD's behavior, we systematically analyze the effects of the adversarial manipulations by employing interpretability methods.
- Finally, we assess the robustness of the various attacks considered to possible defense mechanisms based on anomaly detection.

The remainder of the paper is structured as follows: Section 2 presents an overview of related works, contextualizing our research within existing studies. Section 3 provides a formal description of our attack strategy, detailing the threat model and the algorithms adopted. Comparative experimental evaluation and results are presented in Section 3.3. Section 5 detail the outcomes of an analysis regarding the interpretability of the ML-PWD decision-making process in relation to the features used, whereas the impact of potential defense mechanisms is discussed in Section 6. Conclusions will follow in Section 7.

2. Related work

The first part of this section presents some relevant ML-PWDs aimed to capture complex patterns within input data, such as URLs, raw HTML code, and auxiliary metadata. Given the inherent inaccuracy of ML, the section continues by presenting major AML attacks against ML-PWD and discussing other works that adopt heuristic research for crafting adversarial examples in other domains.

2.1. Phishing detection

Early phishing detection algorithms were generally based on heuristic and rule-based methodologies, with a strong emphasis on blacklisting and signature-based matching mechanisms, as exemplified by PhishNet [5]. These approaches, while effective in specific static contexts, often lack adaptability against evolving phishing strategies. On the contrary, ML models have proven to be effective in capturing complex patterns within input data, such as URLs, raw HTML code, and auxiliary metadata. The authors of [6] proposed the use of recurrent neural networks (RNNs) for phishing detection, achieving high accuracy using only URL strings as input. Their approach eliminated the need for explicit feature construction, showcasing the potential of sequence-based architectures in modeling the syntactic structure of URLs. More recently, PhishHunter [7] has been proposed to detect homographs in URLs by employing Siamese Networks, while in [8] the authors provide a sophisticated deep-ensemble framework for enhancing the process of URL features extraction. Similarly, WebPhish [9], leverages an end-to-end deep neural network architecture comprising convolutional layers, which learn semantic embeddings directly from both raw URL strings and HTML content. Phishmon [10], instead, adopts a comprehensive feature-based framework, aggregating a wide range of signals including HTTP headers, SSL certificate attributes, and characteristics derived from embedded JavaScript files.

An alternative line of research focused on visual-based phishing detection techniques, which leverage the appearance and layout of webpages to identify malicious content. Earlier works, such as [11], utilized the Earth Mover's Distance (EMD) to quantify visual similarity between webpage screenshots, enabling the identification of phishing sites that closely mimic legitimate counterparts. In [12] a CSS-based approach, emphasizing the role of stylistic and structural cues in distinguishing phishing pages from authentic ones, is proposed. More recently, the authors of [13] introduced a framework designed to evaluate visual similarity across webpages, enhancing detection efficacy through refined image comparison techniques. The authors of [14] investigated the application of multimodal Large Language Models (LLMs) in phishing detection by leveraging visual elements such as logos and color schemes.

2.2. Adversarial attacks

Despite the promising performance of ML-based phishing detectors, these models are also susceptible to adversarial samples. For instance, DeepPhish [15] is a framework that simulates how malicious threat agents can leverage deep learning, specifically LSTM networks, to generate more effective phishing URLs. By analyzing over a million phishing URLs and training models from the attacker's perspective, the authors demonstrated a substantial increase in evasion effectiveness, highlighting the potential misuse of AI to bypass ML-PWDs. MintA [16], instead, is a black-box multi-instance adversarial attack on GNN-based malicious domain detection. Unlike single-node attacks, it perturbs multiple attacker-controlled nodes simultaneously, optimizing both features and neighborhood connections to evade detection across many domains, without requiring any knowledge of the model's architecture, parameters, or benign nodes. The practicality of adversarial attacks on ML systems is highlighted in [17], where a variety of evasion techniques are presented demonstrating that such attacks can be effective without altering the functionality or visual appearance of a web page. The effectiveness of feature-centric evasion strategies on tabular datasets is explored in [18] by conducting attacks both in the feature and in the problem space. Other approaches, such the one proposed in [19], adopt an AI-powered generation techniques in order to craft adversarial samples by perturbing either static and dynamic features.

SpacePhish [2] is a framework for evaluating realistic adversarial evasion attacks against ML-PWDs. By formalizing the concept of the evasion space, the authors demonstrated that effective attacks can be mounted within the feature space without altering website functionality or appearance. The evaluation of twelve different strategies highlighted significant drops in detection accuracy, emphasizing the need for more robust and security-aware ML defenses. Building on this work, the same authors further refined the experimental analysis and methodologies in [20], where they also presented a human-in-the-loop study investigating the detectability of adversarial samples by human observers. In [4] fine-grained HTML manipulations capable of preserving both the rendering and the functionality of the webpage are introduced. Using a query-efficient black-box optimization, the method significantly outperforms earlier approaches, degrading detection performance with a few queries. An attack on logo-based phishing detectors is discussed in [21]; generative adversarial perturbations are used to craft logos that while remaining visually convincing to users can deceive deep learning models with up to 95% evasion rate.

The authors [22] proposed a vision-based approach to adversarial example generation that leverages diffusion models in order to generate semantically consistent yet visually diverse logo variants. By capturing the latent visual semantics of a logo, the method produces new logo instances that remain recognizable to human users while significantly altering their pixel-level appearance, thereby increasing the likelihood of evading visual phishing detectors that rely on logo similarities. In [23], the authors show how Generative AI models can be leveraged to produce fraudulent webpages and phishing emails by exploiting the capabilities of conversational models such as GPT-3.5 Turbo. Their findings demonstrate that even unmodified (vanilla) models can generate zero-shot adversarial examples; furthermore, carefully designed prompts can significantly refine and specialize this capability, enabling the generation of increasingly convincing phishing content. Another framework,

Table 1
Comparison between our approach and other relevant adversarial attacks to ML-based phishing detectors.

	Domain	White- / Grey- / Black-Box	Attack Space	Features	Manipulations	Rendering / Functionality Preserving
[2]	Webpages	W, G, B	Problem Space	URL & HTML	Fixed Set	R, F
[4]	Webpages	B	Problem Space	URL & HTML	Fixed Set	R, F
[15]	Webpages	B	Problem Space	URL	Model-Based	R, F
[16]	Domains	B	Feature Space	Domain Malicious Graph	Model-Based	n.a.
[17]	Webpages	W, G, B	Problem Space	HTML	Fixed Set	R, F
[18]	Webpages	B	Feature Space	URL & HTML	Importance-Guided	n.a.
[19]	Webpages	G	Feature Space	URL & HTML & HTTP	Model-Based	n.a.
[20]	Webpages	W, G, B	Problem Space	URL & HTML	Fixed Set	R, F
[21]	Webpages	B	Problem Space	HTML (Logo)	Model-Based	F
[22]	Webpages	B	Problem Space	HTML (Logo)	Model-Based	F
[23]	Webpages & Emails	B	Problem Space	Prompts	Model-Based	n.a.
[24]	Webpages	B	Problem Space	HTML	Fixed Set	n.a.
[27]	Emails	n.a.	Problem Space	Prompts	Model-Based	n.a.
Our	Webpages	B	Problem Space	URL & HTML	Fixed Set	R, F

named PhishOracle, is proposed in [24] to generate adversarial phishing pages by embedding diverse attack features into legitimate sites, significantly reducing detection rates in task-specific models and partially evading MLLM-based detectors. An insightful perspective is presented in [25], where it is shown that ML-PWDs have reduced effectiveness on web pages originating from "Eastern" regions, primarily due to inherent language biases in the models.

The authors of [26] addressed a different phishing domain by introducing *SpearBot*, an LLM-based adversarial framework for generating highly targeted spear-phishing emails. By incorporating iterative feedback from auxiliary Critic-LLMs, the system progressively refines its outputs to enhance realism and improve its ability to evade detection mechanisms. Similarly, in [27] the use of large language models, specifically ChatGPT, is explored in order to generate realistic phishing emails for training purposes. The authors examine how prompt engineering and incorporating social engineering techniques can make the generated content more persuasive and authentic. Ultimately, they demonstrate that such models can efficiently produce convincing phishing emails for use in simulated awareness campaigns.

2.3. Search-based adversarial example generation

To the best of our knowledge, the research of heuristic search-based AML in the context of ML-PWDs is largely unexplored. Nevertheless, some approaches have been evaluated in other cybersecurity areas.

In [28], Monte Carlo Tree Search (MCTS) is applied to craft XSS attack payloads, leveraging its reward-driven nature to guide the search process, in order to explore the resilience of the detector. In [29], MCTS is combined with synonym selection in order to generate textual adversarial samples in a black-box setting, achieving high success rates, while avoiding local optima. MCTS proves to be an efficient approach in a wide spectrum of scenarios, such as evading malware classifiers [30] and crafting prompts to jailbreak LLMs [31]. In the context of malware, MCTS has been used in a gray-box setting where attackers train surrogate models to identify evasive binary mutations capable of bypassing real-world antivirus classifiers. Also, in LLMs settings, MCTS underpins automated prompt-generation strategies that search for adversarial suffixes to bypass safety filters across open and closed-source models, demonstrating better attack success rates.

Other works have focused on Beam Search (BS) to optimize the adversarial generation process. An example is *BeamAttack* [32], which provides a textual attack algorithm that makes use of mixed semantic spaces to improve the quality of adversarial samples. *TABS* [33] leveraged BS to create efficient adversarial samples in NLP settings, by overcoming the inefficiencies of greedy techniques. Similarly, BS is adopted in [34] to generate reversible adversarial images in black-box settings, combining their generative approach with reversible data hiding techniques.

The use of non-conventional tree-based search strategies, such as Simulated Annealing (SA), has gained attention in the AML literature for its effectiveness in navigating complex optimization landscapes. *BESA* [35] leveraged SA in textual information domain to perform word-substitution attacks, demonstrating its ability to efficiently explore discrete input spaces. When dealing with images, *CamoPatch* [36] integrated SA into the optimization process in order to generate adversarial patches. More recently, a similar technique was employed in [37] to craft samples against time-series classifiers, constrained within a fixed L_∞ norm ball.

2.4. Discussion

Analysis of the literature reveals that adversaries are increasingly exploiting advanced evasion techniques for compromising ML-based phishing detectors, including subtle visual manipulations and Generative AI-driven techniques. Although this latter type of approaches can be exploited to create diverse and highly realistic phishing content, thereby enhancing the scalability and adaptability of attacks, their adoption often lacks fine-grained control over the transformations applied and incurs significant computational costs. On the other hand, approaches based on predefined manipulations generally ensure greater efficiency and stability, though at the cost of greater structural rigidity. Moreover, most of these methods rely on sample generation strategies that are either static or guided by greedy optimization criteria; while this may encourage convergence towards local minima, it can also result in the application of more manipulations than those strictly necessary.

In this context, this study aims to balance these aspects through the use of heuristic optimization techniques, with the aim of maximizing the effectiveness of the generated adversarial samples whilst minimizing the number of manipulations applied. The approach is also situated within a black-box scenario, in line with existing literature, demonstrating how even relatively simple optimization strategies are capable of producing effective samples, avoiding the overhead typical of GenAI-based methodologies and preventing the system from quickly falling into local minima. The main characteristics of our approach are summarized in Table 1, which also provides a comparison with some of the works discussed in this Section.

3. Attack scenario and strategy

In the scenario considered, the attacker aims to drive the classifier to misclassify samples drawn from the input data distribution $p(D)$, where D represents the original dataset, by leveraging a chosen set of manipulations. These can be seen as primitive actions $\Gamma = \{\gamma_i\}_{i=0}^{k}$ that, when applied to the page W, generate an altered version W^* whose characteristics depend on the specific γ_i.

Table 2
Single-round and multi-round (grey) manipulations enabling the attack.

ACTION	DESCRIPTION
InjectExtElem (IEE)	Introduces hidden external reference elements into the page randomly extracted from the *Alexa Top Million ranking*.
InjectExtElemFoot (IEF)	Adds hidden external reference elements to the page footer, using the `hidden` attribute or inline style to hide them.
InjectFakeCopyright (IFC)	Inserts a hidden copyright notice into the web page, while maintaining the visual integrity of the layout.
InjectFakeFavicon (IFF)	Inserts a placeholder favicon into the browser tab, providing one if is not already present.
InjectIntElem (IIE)	Add hidden internal reference elements using inline styles or the `hidden` attribute to unobtrusively simulate the content structure.
InjectIntElemFoot (IIF)	Embeds hidden internal reference elements in the footer using inline styling or the `hidden` attribute.
InjectIntLinkElem (IIL)	Adds hidden internal reference links to emulate legitimate intra-site navigation.
ObfuscateExtLinks (OEL)	Transforms external hyperlinks to make them temporarily inactive, restoring functionality post-page load via JavaScript.
ObfuscateJS (OJS)	Encodes JavaScript in Base64 and injects a runtime decoder to dynamically execute the original code, enhancing obfuscation.
UpdateForm (UPF)	Detects form elements with potentially suspicious `action` attributes and replaces them with more inconspicuous alternatives (e.g., `#none`, `#!/`).
UpdateHiddenButtons (UHB)	Temporarily removes the `disabled` attribute from buttons, reapplying it via client-side script once the page-load is completed.
UpdateHiddenDivs (UHD)	Modifies hidden elements by changing CSS visibility properties or adding unique identifiers to mask their presence.
UpdateHiddenInputs (UHI)	Modifies hidden input fields and employs JavaScript to restore their original configuration during page initialization.
UpdateIFrames (UIF)	Applies obfuscation and hiding techniques, similar to those used in UpdateHiddenDivs, but to `<iframe>` components.
UpdateIntAnchors (UIA)	Substitutes internal anchor links with innocuous alternatives (e.g., `<a href="#!/">`) to mask their original intent.
UpdateTitle (UPT)	Temporarily replaces the page's title with the website's domain name, reverting it to the original via script post-render.

3.1. Admissible manipulation

Since the set of admissible manipulations is inherently limited and dependent on the specific application scenario, we adopted the same introduced in [4] (see Table 2), which is grounded in existing literature and used as a reference for problem-space adversarial attacks [2,38]. This set is not meant to be exhaustive, but rather representative of the *rendering preserving* adversarial attacks conducted in real scenarios and similar studies. Indeed, the manipulations employed reflect common techniques, such as the injection of malicious JavaScript payloads, the deliberate creation of link-imbalanced webpages to exploit ranking algorithms, and other content-level and structural perturbations designed to evade automated detection systems. Manipulations that have a fixed effect on the webpage structure, which would make further applications redundant, are therefore applied at most once and referred to as *single-round*; conversely, *multi-round manipulations* have incremental effects, enabling progressive modifications that cannot be achieved through single-round operations alone.

It is worth noting that certain manipulations are characterized by a limited stochastic factor which, however, do not significantly affect either the attack procedure or the validity of the results. In particular, InjectExtElem, InjectExtElemFoot, UpdateForm, and ObfuscateExtLinks, although making random selections, always operate on finite, predefined, and controlled sets. The first two, in fact, randomly select a link from a set of reliable URLs, while UpdateForm replaces the "action" attribute of a form with one of two default placeholders (#! or #none). Similarly, ObfuscateExtLinks obfuscates external references at runtime using JavaScript code inserted into the <head>, while reusing the same internal references. The impact of these choices on the reproducibility of the attack will be further discussed in Section 4.3.2.

Regardless of their nature, all manipulations operate exclusively in the problem space and affect only the structural representation of the phishing webpage, without altering its functional behavior. This property is intrinsic to the definition of the adopted manipulations and was empirically verified by following the same methodology of [4], i.e., each web page and its corresponding adversarial version were rendered, captured as screenshots, and compared using SHA-256 hashing to ensure complete visual consistency. While alternative manipulations may be feasible in other attack scenarios, such as visual-based attacks involving logo or CSS alterations, these fall outside the considered attack scenario.

3.2. Threat model

The goal of our attack is to optimize the choice of the manipulations in order to reduce the differences between the adversarial sample and the original one. The attack is *exploratory* and *black-box*, with little to no prior knowledge of data distribution or model parameters, other than the assumption that it is trained on features commonly used in literature. To provide a systematic description of the threat model, we represent its underlying dynamics as a sequence of states $S_j = (W, Pi, Pf, \gamma, T)$, where:

- W: website page, including both its HTML code and URL;
- P_i: classification score of the page at step $j-1$, i.e., the probability of it being classified as *Phishing*;
- P_f: classification score of the page at step j;
- γ: manipulation applied at step j to the page W;
- T: a boolean indicating whether the current state is terminal.

A state is deemed **terminal** if the classification score of the corresponding adversarial sample falls below a threshold θ. This serves as an effective early stopping criterion, preventing unnecessary manipulations once the attack has successfully degraded the classifier's confidence. The adversary's objective is to identify an optimal sequence of states $C^* = \{S_i\}_{i=0}^{m}$ that ultimately produces an adversarial sample capable of deceiving the classifier. However, we also settled for **suboptimal solutions**, as long as they satisfy the early stopping criterion, ensuring efficiency while still achieving the intended evasion. The optimization process can be formulated as follows:

$$\begin{aligned} \text{Find a sequence} \quad & \hat{C} = \{S_i\}_{i=0}^{\hat{m}} \\ \text{s.t.} \quad & f(S_{\hat{m}}) < \theta \end{aligned} \tag{1}$$

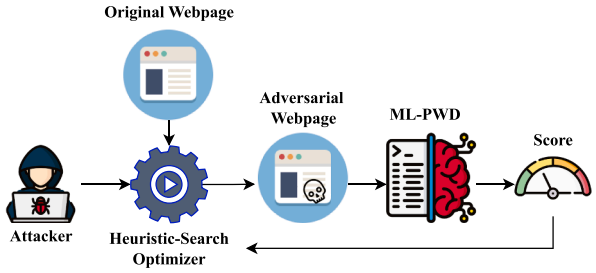

Fig. 1. The proposed attack strategy. The attacker can use different heuristic search techniques to craft an adversarial sample able to deceive the target ML-PWD. Classification score is used to reiterate the process until the optimal solution is found.

where f denotes the classification function, and $\hat{m}$ the index of the final state in the sequence.

3.3. Heuristic search

The core of our attack stems from the limitations observed in existing methods, particularly with respect to the quality of the adversarial samples generated. For example, in [4], a best-first strategy is employed, where the most promising manipulations are applied and the target model is iteratively queried within a predefined budget. While such an approach is query efficient, in the sense that the number of queries cannot exceed the budget, it does not consider the number of modifications applied to each sample. To address this issue, we integrate a set of heuristic search techniques into the crafting process (see Fig. 1), with the aim of increasing efficiency while maintaining the effectiveness of the attack. In particular, Monte Carlo Tree Search, Beam Search, and Simulated Annealing were chosen because of their efficiency, adaptability, and effectiveness in navigating the search space toward optimal adversarial samples.

Monte Carlo Tree Search (MCTS) [39] builds statistical evidence about possible decisions and determines how actions can be applied to move from one state to another in a structured manner. Rewards are assigned accordingly, with terminal states receiving higher payoffs, while intermediate states are evaluated according to their proximity to the goal. MCTS is particularly effective in sequential decision-making for complex systems, where exhaustive search is not feasible, as it ensures that a solution is always found. This key feature is the primary rationale for the adoption of MCTS, as it offers a reward-driven framework for the generation of adversarial samples. Manipulations that prove ineffective for a given sample are naturally discarded as the algorithm progresses through its iterations. In general, the most promising action identified by MCTS results from:

$$a^* = \arg\max_{a \in A(s)} Q(s,a), \tag{2}$$

where $Q(s,a)$ represents the estimated value of taking the action a from the state s. A widely adopted approach to balancing exploration and exploitation in sequential decision-making is the Upper Confidence Bound for Trees (UCT) [40], an extension of MCTS. UCT guides the search process by prioritizing actions that strike an effective trade-off between well-explored options and those with high potential reward, based on statistical confidence bounds. In the proposed framework, this principle is incorporated through a tailored scoring function, which achieves this balance by averaging state visit counts and accumulated rewards:

$$\gamma^* = \arg\max_{\gamma \in A(S)} \left(Q(S,\gamma) + C\sqrt{\frac{\ln N(S)}{N(S,\gamma)}} \right). \tag{3}$$

Here, γ^* is the optimal manipulation selected from the set of possibile actions $A(S)$, where S is the current state. $Q(S,\gamma)$ represents the estimated value of applying γ from state S, $N(S)$ denotes the total number of visits to S, while $N(S,\gamma)$ indicates the number of times γ has been chosen from S. The constant C is a parameter that determines the trade-off between exploration. A key strength of this approach is its ability to anticipate long-term payoffs by simulating multiple sequences of manipulations to identify optimal strategies. This allows efficient exploration of the search space, minimizing the number of manipulations while maximizing the effectiveness of adversarial attacks.

Beam Search (BS) [41] extends Breadth-First Search (BFS) by incorporating pruning to eliminate unpromising nodes at each level. Unlike traditional BFS, BS limits exploration to a fixed number of nodes per level, defined by the beam width, ensuring a more efficient search process. This procedure continues until the target state is reached. Within our method, the BS process can be described as follows:

$$\begin{aligned} \Gamma_{t+1} = \arg\min_{\hat{\Gamma} \subseteq A(S_t^i)} & \{ f(h(S_t^i, \gamma)) : \gamma \in \hat{\Gamma} \} \\ \text{s.t.} \quad & |\Gamma_{t+1}| \leq W \end{aligned} \tag{4}$$

where S_t^i represents the state of the i-th beam at time step t, and $A(S_t^i)$ is the set of available actions that can be applied to S_t^i to generate potential successor states. The state transition function $h(\cdot)$ maps the current state S_t^i, by applying γ, to a new state. The objective function $f(h(S_t^i, \gamma))$ evaluates each resulting state, being f the classification function mentioned above (Eq. 1). The goal is to select a subset $\hat{\Gamma}$ of actions, relative to the most effective tree-branch, that minimizes this objective, thereby selecting the most promising actions that lead to a poor classification score. The constraint introduced ensures that W actions (states) are retained, keeping the search focused and manageable by limiting the number of candidate actions (solutions). Please note that, differently from other approaches that rely on individual actions, BS naturally yields sets of actions at each step. Thus, the beams have to be maintained and expanded at each step (Fig. 2), applying the newly found manipulations and eventually obtaining the optimal crafted adversarial sample and its relative tree-branch. This strategy was specifically chosen for its ability of progressively narrowing the search space to the most promising candidates, which minimizes overhead while increasing the likelihood of discovering effective adversarial samples.

Simulated Annealing (SA) [42] is inspired by metallurgical annealing; it explores a solution space by evaluating candidate solutions using an energy function, with the temperature parameter T controlling the probability of accepting suboptimal solutions. As the temperature decreases, the search becomes more focused, refining the search and avoiding local optima. This process can be formulated as:

$$\Gamma^* = \arg\min_{\Gamma \in A(S)} \{ f(h(S, \Gamma)) \}. \tag{5}$$

Here Γ^* represents the optimal action sequence chosen from the set of possible actions $A(S)$, where $h(S,\Gamma)$ describes the transition from the current state to an optimal one, after applying the action sequence Γ, and f is the *energy* (classification score) used to evaluate the quality of the new state. More precisely, at each time step t, starting from the current state S_t, a new candidate state S'_{t+1} is generated by applying a randomly selected action $\gamma \in A(S_t)$. If the score of the new state is lower than that of the current one (i.e., $f(S'_{t+1}) < f(S_t)$), the candidate is accepted. Otherwise, it is accepted with a probability defined by the Metropolis-Hastings [43] criterion:

$$P_{\text{accept}} = \exp\left(-\frac{f(S'_{t+1}) - f(S_t)}{T_t} \right). \tag{6}$$

We chose SA because of its ability to navigate complex and unknown function spaces, a crucial advantage in our study. The probabilistic nature of the algorithm allows it to avoid premature convergence to suboptimal solutions, ensuring a more robust search.

This characteristic is also common to the other methods discussed above, as opposed to general greedy strategies that select manipulations according to:

$$\gamma_t = \arg\min_{\gamma} f(S, \gamma), \tag{7}$$

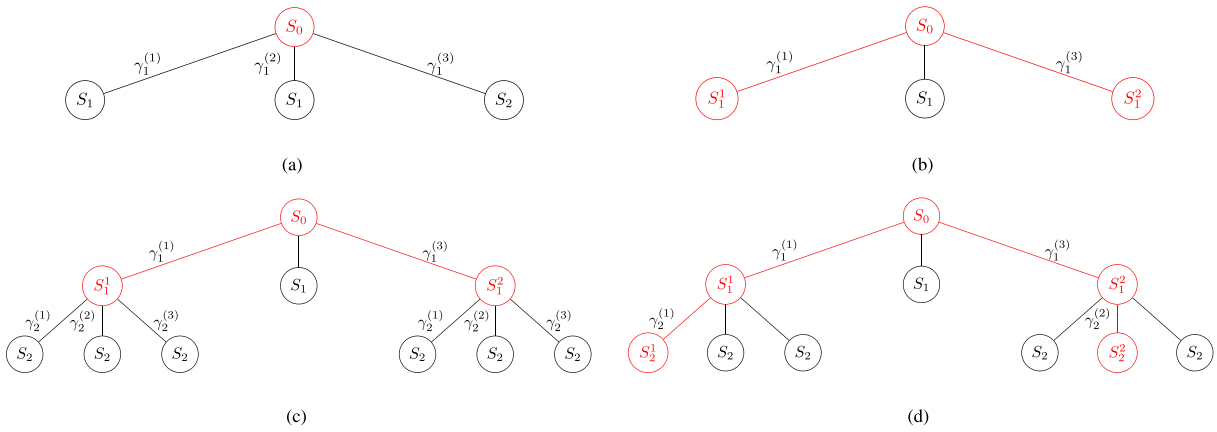

Fig. 2. BS example. At each time step t, actions $\gamma_t^{(i)} \in A(S_{t-1})$ are evaluated from a given state S_{t-1}. The top BW successors are retained based on the objective function, maintaining a fixed beam width (BW = 2 in this example). States are then labelled as S_t^j, where j denotes the beam index at time t.

Algorithm 1 Adversarial sample generation through heuristic-search optimizer.

1: **Input:** W (original sample), M (ML-PWD under attack), θ (predefined threshold), O (heuristic-search optimizer).
2: **Output:** P^* (adversarial sample), $\hat{C}$ (sequence of states).
3: $P_{i_0} \leftarrow M$.classify(W) ▷ Initial classification score
4: $Pf_0 \leftarrow P_{i_0}$
5: $\gamma_0 \leftarrow None$
6: $T \leftarrow (Pf_0 < \theta)$ ▷ Run a sanity check
7: $S_0 \leftarrow (W, P_{i_0}, Pf_0, \gamma_0, T)$ ▷ Initialize state
8: $S_c \leftarrow S_0$ ▷ Current state
9: $\hat{C} \leftarrow [S_0]$
10: **while** $S_c.T \neq$ True **do**
11: $\gamma_j \leftarrow O$.next_manipulation(S_c) ▷ Optimizer yields the next manipulation
12: $W^* \leftarrow$ apply_manipulation($S_c.W, \gamma_j$)
13: $P_{i_j} \leftarrow S_c.P_{f_{j-1}}$
14: $P_{f_j} \leftarrow M$.classify(W^*)
15: $T_j \leftarrow (P_{f_j} < \theta)$
16: $S_c \leftarrow (W^*, P_{i_j}, P_{f_j}, \gamma_j, T_j)$
17: $\hat{C}$.append(S_c)
18: **end while**
19: **return** $W^*, \hat{C}$ ▷ Return adversarial sample and sequence of states

where the chosen, single and deterministic, search trajectory converges on a local optimum $S^\star$, without addressing the sub-optimality gap:

$$\left| \min_{S \in \mathcal{S}} f(S) - f(S^\star) \right|. \tag{8}$$

This boundary is crossed by MCTS, BS, and SA, which are capable of exploring the search space more thoroughly by following multiple trajectories. Consequently, the non-zero probability of reaching intermediate sub-optimal states allows these methods to provide strictly stronger worst-case optimality and robustness than greedy search.

The whole process of generating adversarial samples through heuristic-search optimization is summarized in Algorithm 1, which is aimed to manipulate the input sample W in order to deceive a target detector M, while satisfying a quality threshold θ, according to the chosen heuristic optimizer O.

This method is both model-agnostic and optimizer-independent, which provides high modularity. Until a terminal state is reached (line 10), the optimizer iteratively proposes the next action(s) it considers most promising for steering the process toward the desired objective (line 11). The selected manipulation γ_j is then applied to the webpage W associated with the current state S_c, producing a perturbed version W^*. The classifier is queried to obtain the corresponding prediction probabilities, from which the classification score Pf_j is extracted (lines 12-14). Based on whether Pf_j falls below the threshold θ, the terminal flag T_j is set accordingly (either True or False) and the state is updated before the next iteration begins (lines 15-17).

It is also worth noting that there is a duality between the sequence of states and the sequence of actions, as actions serve to *connect* two states, acting as edges between nodes in the search tree. This distinction is important: while it is generally more informative to consider sequences of states when analyzing the overall attack process, from the optimizer's perspective it is often more practical to reason in terms of sequences of actions. Moreover, a simple but effective pruning strategy can be employed that derives from the nature of single-round manipulations, which, once executed, are omitted from all subsequent iterations. This property leads to a natural reduction of the search space, thereby increasing the efficiency of the traversal as the process progresses.

4. Results

This section presents the experimental validation of the proposed approach, also with respect to the performance of some relevant related works selected as baselines.

4.1. Experimental settings

The experimental analysis is carried out using two widely adopted public datasets, namely δPhish [44] and Zenodo [45]. Both provide samples of web pages, including either raw URLs or HTML code, thus allowing adversarial samples to be generated directly in the problem space. δPhish contains a total of 6,523 samples, 5,511 of which are benign, while the remaining 1,012 belong to the phishing category. Zenodo is considerably larger, containing 100k of phishing and almost 4M of benign pages, In order to ensure a consistent comparison with prior work, the same training strategy employed in [2,4] is adopted. Specifically, from each dataset, 4.000 random samples are chosen, balancing the two classes; from these, 100 phishing samples are removed in order to evaluate evasion, ensuring at the same time that there is no data contamination. Finally, on the remaining samples, an 80 – 20% train-test split is performed aiming to train five different ML-PWD models:

- Logistic Regressor (LR): A simple yet effective linear model used for binary classification, which predicts the probability that an input belongs to a particular class by applying a logistic function to a weighted sum of features.
- Random Forest (RF): An ensemble learning method that builds multiple decision trees and combines their outputs to improve accuracy and robustness. RF mitigates overfitting by averaging predictions from multiple trees, making it suitable for high-dimensional data.

Table 3
Parameters of the heuristic search algorithms considered. Values were selected based on empirical tuning and prior work to ensure effective and fair comparison across methods.

Algorithm	Parameter	Description
MCTS	$C = 1.41$	Exploration factor, balances between exploration and exploitation.
	$N = 36$	Maximum number of selection steps (tree depth).
	$R_{no} = 5$	Number of rollouts per selection step.
	$R_{lim} = 16$	Maximum number of transitions during each rollout.
	RP: *random uniform*	Defines how to simulate a complete game or outcome from a leaf node.
BS	$BW = 3$	Maximum number of states maintained per search level.
	$D = 10$	Maximum search depth.
SA	$T = 75$	Initial temperature, controls the probability of accepting worse solutions.
	$CR = 0.95$	Cooling rate, regulates the decrease of temperature.
	$MaxIter = 30$	Maximum number of allowed iterations.

Table 4
List of features capable of capturing URL-based (F^u) and HTML-based (R) attributes.

URL_length	URL_punyCode	URL_ports	HTML_freqDom	HTML_loginForm
URL_hasIPaddr	URL_sensitiveWrd	URL_SSL	HTML_objectRatio	HTML_hiddenDiv
URL_redirect	URL_TLDinPath	URL_statisticRe	HTML_metaScripts	HTML_hiddenButton
URL_short	URL_TLDinSub	URL_pageRank	HTML_commPage	HTML_hiddenInput
URL_subdomains	URL_totalWords	URL_regLen	HTML_commPageFoot	HTML_URLBrand
URL_atSymbol	URL_shrtWordURL	URL_checkGI	HTML_SFH	HTML_iframe
URL_fakeHTTPS	URL_shrtWordHost	URL_avgWordPath	HTML_popUp	HTML_favicon
URL_dash	URL_shrtWordPath	URL_avgWordHost	HTML_rightClick	HTML_statBar
URL_dataURI	URL_lngWordURL	URL_avgWordURL	HTML_domCopyright	HTML_css
URL_commonTerms	URL_DNS	URL_lngWordPath	HTML_nullLnkWeb	HTML_anchors
URL_numerical	URL_domAge	URL_lngWordHost	HTML_nullLnkFooter	
URL_pathExtend	URL_abnormal		HTML_brokenLnk	

Table 5
Evaluation metrics.

P_i	The initial probability (classification score) assigned by the classifier to the original samples (step 0, in terms of Alg.1).
P_f	The final probability (classification score) assigned by the classifier to the perturbed samples (step $\hat{m}$, in terms of Alg.1).
L_p	Minkowski distance (with $p = 0$, $p = 2$, and $p = \infty$) between the original sample and the adversarial sample.
MN	Total number of manipulations applied to a sample at the end of the optimization procedure.
AT	Time to complete the attack (seconds).
QN	Number of queries made to the model under attack.

- Convolutional Neural Network (CNN): Deep learning models that use convolutional layers to extract features and capture intricate patterns. In website phishing analysis, they can identify relationships that traditional models overlook.
- Deep Neural Network (DNN): A feed-forward neural network, which can capture complex, non-linear patterns in data. Techniques such as batch normalization, dropout and regularization were incorporated to improve generalization and reduce overfitting
- Soft Voting Ensemble (ENS): The ensemble averages the predicted class probabilities of three classifiers chosen for their properties. In particular, (i) Decision Tree (DT) captures nonlinear relationships using simple, interpretable models; (ii) Random Forest (RF) allows to enhance stability and reduce overfitting by aggregating multiple trees; (iii) Gradient Boosting (GB) sequentially corrects errors in order to model complex patterns. Integrating these complementary strategies improves the ensemble's robustness and predictive accuracy in feature classification.

The parameters employed for the heuristic search methods were derived through empirical analysis, and are reported in Table 3. Experiments were run on a workstation equipped with AMD Ryzen 5 5600G processor (6 cores/12 threads), 24GB of RAM, an NVIDIA GeForce RTX 4060 GAMING OC with 8GB GDDR6. For the sake of reproducibility, the random component of the evaluation process was suppressed by using a seed set to 42. Thus, the whole process can be fully traced and reproduced to obtain the same sequence of manipulations we used; nevertheless, it is possible to run the experiments without a fixed seed and this would result in slightly different, but broadly equivalent sequences.

In order to conduct a comparative analysis with state of the art adversarial attacks, three baselines were selected, namely *Raze to the Ground* (RTTG) [4], *SpacePhish* [2] and *Yuan* et al. [20]; all attacks have been reproduced by exploiting the implementations made available by the respective authors on GitHub. For the latter two works, in order to ensure methodological consistency with RTTG and our work, we selected only those configurations that were relevant to our analysis, namely problem-space and black-box methods. Moreover, as discussed in *SpacePhish*, variants with greater knowledge do not necessarily perform better than the black-box one, which further justifies our choice.

4.2. Features and metrics

All these models were trained on a set of features (Table 4) that have been widely used in the literature [2,4,46–49]. Some of them, denoted as F^r (shortly addressed with R), can be extracted directly from the raw HTML of the webpage and capture both structural and content-based information. Others, F^u, are derived from the URL of the page and include lexical properties, domain-related information, and heuristic indicators of malicious intent. Both sets can be considered

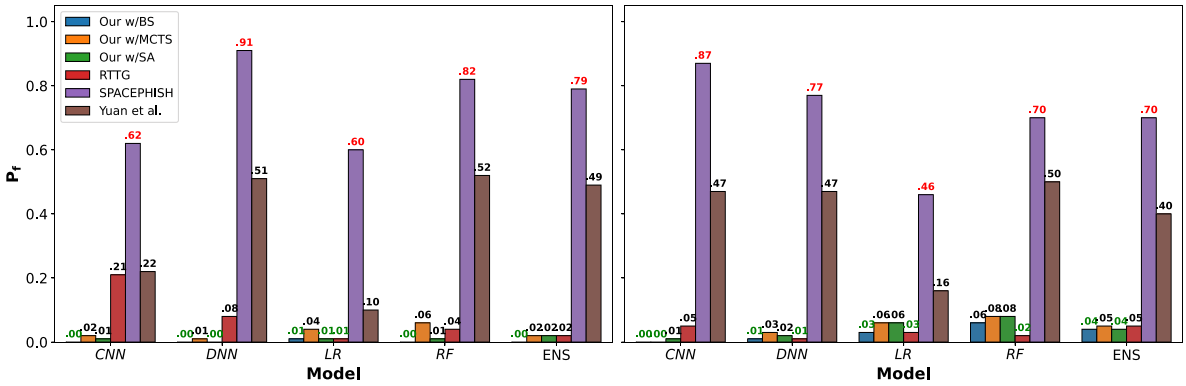

Fig. 3. The average results of the attacks performed using the *SpacePhish*, *RTTG*, *Yuan* et al. and our method, under the different search strategies (BS, SA, MCTS), across the two datasets, δPhish and Zenodo, for R (left) and C (right), respectively.

together, F^c (referred to as C), to provide a holistic representation of the webpage. In line with our threat model, we have focused specifically on actions that lead to the modification of the set R. Nevertheless, the set C is also used in the experiments to evaluate the robustness of the ML-PWDs depending on whether they are trained on features describing only the HTML of the pages, or also the characteristics of their URLs.

Performance was evaluated according to the suite of metrics reported in Table 5, with particular emphasis on two key indicators, namely the final score P_f and the L_p norms. The former reflects the practical effectiveness of each attack, while the L_p norms offer a more comprehensive evaluation of the attack's stealthiness and the robustness of the selected feature set. As for the remaining metrics (*MN*, *AT*, and *QN*), they provide valuable context for understanding the practical application of each approach, highlighting trade-offs in terms of manipulation count, execution time, and query efficiency. These metrics help assess the feasibility of different strategies in real-world scenarios, where resource constraints and time limitations are key considerations. Furthermore, the attacks were carried out with a fixed threshold $\theta = 0.1$, see Eq. 1, chosen to force the attack to continue until the model's confidence is practically zero, which may require a fairly high number of manipulations applied to the adversarial sample. Although this configuration prioritizes attack strength, higher threshold values could be adopted to balance the trade-off between evasion grade and sample contamination. For instance, a threshold close to $\theta = 0.5$ would lead to the generation of samples in proximity to the model's decision boundary. In this case, while these borderline samples can effectively confuse the model, their proximity makes them susceptible to rejection even by a confidence-based filter. By leveraging the optimizer to generate samples with minimal phishing confidence, the attack successfully deceives the model without eliciting any suspicion.

4.3. Comparative analysis

The following of this section presents the outcomes of the experimental evaluations conducted under the previously outlined conditions. These assessments were carefully designed to investigate the attack effectiveness, the transferability of the generated adversarial samples, as well as properties related the statistical validity of the results and the quality of both the applied manipulations and the adversarial samples. Finally, the attacks are also compared by aligning their operating conditions, i.e., the number of manipulations allowed and number of queries to the target model.

4.3.1. Attack effectiveness

The first set of experiments concerns the P_f values, i.e, the final probability (classification score) assigned by the classifier to the perturbed samples. Results reported in Fig. 3 show the average P_f values of *SpacePhish*, *RTTG* and *Yuan* et al. as compared to our strategy based on BS, MCTS, and SA, respectively. The different bar groups summarize the results against CNN, DNN, LR, RF and ENS target models (averaged over the two datasets) assuming that the classifiers rely on the set R (leftmost plot) or C (right). The lower the P_f values, the greater the classifier's confidence that the perturbed sample is legitimate; hence, the greater the effectiveness of the attack. At first glance, it is evident that the three heuristic search-based attacks are particularly effective, consistently compromising model performance with low variance in detection probability (P_f). Moreover, the highest Pf values are obtained when considering the whole set C, which indicate a great resistance to the attack. This trend is particularly evident when observing the model RF; here, all the heuristics perform worse than on CNN and LR targets, but still better than *SpacePhish* and *Yuan* et al.. On the other hand, *RTTG* shows a high capability of entirely compromising RF's classification accuracy. which is likely due to the magnitude of the applied perturbations. Indeed, being RF intrinsically resistant to noise and minor input variations, stronger manipulations are required to successfully deceive the classifier. As a consequence, adversarial strategies relying on subtle modifications, which is one of the main goals of our proposal, may fail to sufficiently perturb RF's decision boundaries, whereas more aggressive attack schemes can effectively degrade its performance. Interestingly, an inverse effect is observed in the case of the CNN model, where *RTTG* demonstrates notably limited efficacy similar to that of Yuan et al., suggesting a potential duality in model-specific vulnerability to distinct attack mechanisms. Conversely, although the DNN exhibits marked robustness in R against samples generated by *SpacePhish*, it remains vulnerable to search-based methodologies such as *RTTG* and the proposed approach, yielding an average P_f well below .08. In this setting, Yuan et al. shows a strong ability to substantially reduce the model's confidence in the generated samples. While these samples are not always misclassified, they significantly impair the discriminative capabilities of the deployed classifieR, Particularly when considering the other models. With respect to the ensemble (*ENS*), the presence of heterogeneous components appears to dampen the differences between the attack methodologies, with the notable exception of the aforementioned two approaches. In this case, the BS strategy achieves a marginal advantage, exhibiting the lowest P_f values across both feature sets.

Overall, the considered attacks (with the exception of *SpacePhish*) consistently undermine the robustness of evaluated models by either pushing generated samples towards the decision boundary, thereby increasing predictive uncertainty, or inducing misclassifications.

4.3.2. Transferability of the adversarial examples

To investigate *transferability*, namely, the property whereby adversarial samples are capable of fooling different target models, we run a further set of tests. Experiments were conducted by optimizing the heuristic search for adversarial samples with respect to each of the models listed in the columns of Table 6, and then launching the attack against the different target models specified in the rows. Each value in the Ta-

Table 6
Average transferability of adversarial samples generated by the proposed method across different models while varying both the feature set considered (C or R) and the datasets.

(a) δPhish

	CNN_C	CNN_R	DNN_C	DNN_R	ENS_C	ENS_R	LR_C	LR_R	RF_C	RF_R
CNN_C	100	71	89.6	80.3	87.1	89.2	84	79.3	99	91.7
CNN_R	83.3	99.7	82.9	80	85.9	89.2	75	74	98.3	91
DNN_C	34.3	48.7	74.8	38	34	28.4	14.7	35.3	5.3	26.3
DNN_R	8.7	23.7	3.2	74.8	16.1	5.4	10.7	16	0.7	1.7
ENS_C	76.2	64.5	88	69.2	82.4	87.3	88.2	78.7	99.6	89.7
ENS_R	86.2	77.6	87.6	83.5	89.2	100	86.3	83.4	99.6	99
LR_C	77.2	68.3	94.6	72.8	86.2	88.1	100	88.2	98.8	87.9
LR_R	91.8	88.6	100	93.4	97	99.3	100	100	100	100
RF_C	85.5	70.5	88.4	78.6	88.5	96.4	92.2	86.6	99.6	99.1
RF_R	91	78.5	86.3	83.8	90.1	100	89	83.7	99.2	99.8

(b) Zenodo

	CNN_C	CNN_R	DNN_C	DNN_R	ENS_C	ENS_R	LR_C	LR_R	RF_C	RF_R
CNN_C	100	72.8	89.8	80.9	88	90.3	85.3	81.3	98.8	92.6
CNN_R	84.3	99.9	83	79.8	86.4	89.8	76.3	75.5	97.9	91.5
DNN_C	33.5	47.1	74.8	37.2	33.2	28.7	13.9	32.9	5.2	25.2
DNN_R	7.9	21.4	3.1	74.8	15.2	5.6	9.4	12.7	0.3	1.3
ENS_C	63.6	62.6	81.1	66.7	100	83.2	88.6	74.4	99.7	81.5
ENS_R	67.3	72.7	82.2	73.4	99.3	100	82.5	89.9	99.3	94.3
LR_C	56.9	55.9	77.4	56.2	77.1	68	95.3	67.3	86.5	66.7
LR_R	86.9	87.5	97	90.2	100	99	99	100	100	99.7
RF_C	72.1	73.7	89.6	78.8	100	93.9	93.3	90.2	99.7	94.3
RF_R	71.7	75.1	86.5	74.1	96.3	99	87.9	89.9	99.7	96.3

ble represents the percentage of adversarial samples misclassified by the corresponding target model while varying both the feature set considered (C or R) and the dataset (δPhish or Zenodo). Results indicate that the DNN, in both its variants, is the most robust; indeed, even adversarial samples crafted using the same model exhibit a transferability rate not exceeding 75%, which is likely due to random weight initialization during independent training of different instances. Furthermore, adversarial samples obtained relying on other models fail to significantly compromise the DNN, presumably due to the DNN's ability to capture more complex and intrinsic feature relationships compared to linear models. Notably, the transferability rates of adversarial samples generated through the CNN and transferred to the DNN are 34.4% and 48.7%, respectively. These results suggest that the DNN is particularly resistant to transferred adversarial samples, particularly when the feature set R is employed. Conversely, the LR_R model exhibits the opposite behavior, being the most vulnerable one, even to attacks that exploit the property of transferability. This pattern remains consistent across both datasets, with a transferability rate of approximately 98%. All other model combinations consistently achieve transferability rates above 50%, indicating that the proposed approach can generate highly effective adversarial samples, despite the specific trends discussed above.

4.3.3. Statistical validation

To statistically validate our results, we conducted a series of Welch's t-tests (Table 7), comparing the F-score performance of the six algorithms across the evaluated models. Given the 15 pairwise comparisons, we applied a Bonferroni correction to control for multiple hypothesis testing, resulting in a significance threshold of $\alpha = 0.005$. Furthermore, to demonstrate that the limited stochasticity introduced by certain manipulations does not affect the overall attack performance, we collected instances from multiple runs with different random seeds. Among all the evaluated methods, the BS variant achieved the best performance, yielding the lowest mean P_f, and it significantly outperformed all the competing approaches, notably SpacePhish, with all the associated p-values well below 0.001. Furthermore, each of the three proposed variants demonstrated statistically significant improvements over both baseline methods. The BS strategy also showed statistically significant gains compared to SA and MCTS ($p < .001$ in both cases). However, no statistically significant difference was observed between RTTG and Our w/MCTS (t = 1.29, p = 0.198), suggesting that these approaches perform comparably in terms of efficacy of the generated samples. Yuan et al. improves upon *SpacePhish*; however, it is still clearly outperformed by both the proposed method and RTTG, with t-statistics in the range of −46 (lower mean values indicate better performance). Collectively, these outcomes provide compelling evidence that the proposed method, particularly when using BS, substantially outperforms the already well-established techniques.

Table 7
Pairwise Statistical Significance Tests (Welch's t-test adjusted with Bonferroni) for Algorithm Performance.

Comparison	t-statistic	p-value
Our w/BS vs Yuan et al.	−48.87	< 0.001
Our w/MCTS vs Yuan et al.	−44.31	< 0.001
Our w/MCTS vs Our w/BS	9.31	< 0.001
Our w/SA vs Our w/BS	5.77	< 0.001
Our w/SA vs Our w/MCTS	−3.53	< 0.001
Our w/SA vs Yuan et al.	−46.16	< 0.001
RTTG vs Our w/BS	6.36	< 0.001
RTTG vs Our w/SA	3.42	< 0.001
RTTG vs Our w/MCTS	1.29	0.198
RTTG vs SpacePhish	−79.86	< 0.001
RTTG vs Yuan et al.	−38.58	< 0.001
SpacePhish vs Our w/BS	97.71	< 0.001
SpacePhish vs Our w/MCTS	91.38	< 0.001
SpacePhish vs Our w/SA	93.82	< 0.001
SpacePhish vs Yuan et al.	33.41	< 0.001

4.3.4. Extent and quality of the manipulations

These subsequent analyses focus on how the number of manipulations is distributed across methods, models and feature sets. At the same time, the effectiveness of the different attack methodologies is examined

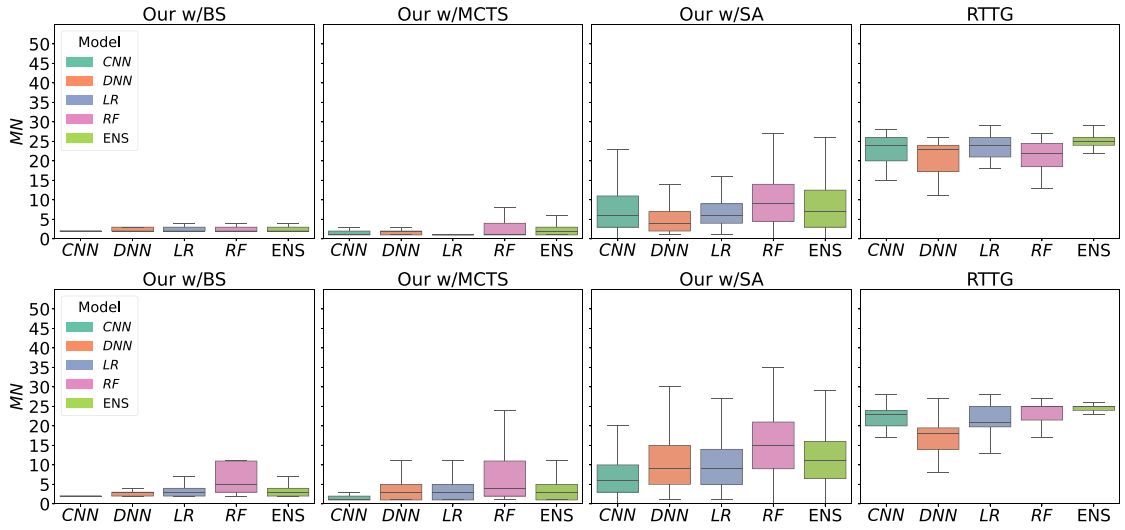

Fig. 4. Average number of manipulations made by *RTTG* and the proposed method (with BS, SA, MCTS), against CNN, DNN, LR, RF and ENS target models. The first row shows the results obtained when the targets are trained on R (top) only, while the second considering the set C (bottom). *SpacePhish* and Yuan et al. are not included as they always perform 50 manipulations.

Table 8
Performance across the δPhish and Zenodo datasets, comparing *SpacePhish*, *RTTG*, and our method under SA, MCTS, and BS search strategies. Bold values indicate the best performance achieved with respect to each evaluation metric.

		δPhish					Zenodo				
Algorithm	Features	Pi	Pf	MN	AT (s)	QN	Pi	Pf	MN	AT (s)	QN
SpacePhish	R	0.77	0.65	50	0	0	0.77	0.72	50	0	0
	C	0.86	0.66	50	0	0	0.86	0.72	50	0	0
Yuan et al.	R	0.77	0.27	11	0	0	0.77	0.44	11	0	0
	C	0.86	0.36	11	0	0	0.86	0.42	11	0	0
RTTG	R	0.77	0.07	23.04	159.59	36	0.77	0.06	22.27	125.37	36
	C	0.86	**0.02**	21.81	129.94	36	0.86	**0.02**	21.81	164.94	36
Our w/SA	R	0.77	0.01	7.64	**1.51**	**7.64**	0.77	0.01	7.29	1.32	**7.29**
	C	0.87	0.03	10.95	**8.15**	**11.13**	0.87	0.05	11.01	**9.63**	**11.13**
Our w/MCTS	R	0.77	0.02	3.45	46.46	669.45	0.77	0.04	5.68	89.61	1114.63
	C	0.88	0.03	5.73	253.85	465.11	0.88	0.06	5.91	432.17	872.41
Our w/BS	R	0.77	**0.009**	**2.52**	2.11	40.22	0.77	**0.005**	**2.27**	**1.33**	28.36
	C	0.87	**0.02**	**3.83**	47.61	89.93	0.87	0.04	**4.22**	52.55	102.95

in more detail, with the aim of highlighting their differences in terms of the evaluation metrics adopted and their distinctive characteristics. As illustrated in Fig. 4, all the compared methods apply a different number of manipulations (MN). Please note that *Spacephish* and *Yuan* et al. have been omitted as they apply a fixed number of manipulations, i.e., 50 and 11 respectively. As it can be observed, our approach (in all its variations) requires fewer manipulations to successfully compromise a sample. In particular, both BS and MCTS consistently average below 5 manipulations, with a slight increase observed when operating over the extended feature set C. This number is higher when considering SA, but still consistently lower than *RTTG* regardless of the considered feature set.

A more in-depth analysis of the results can be made by analyzing Table 8, which provides the values of all evaluation metrics, separately for the two features sets of the δPhish and Zenodo datasets, varying the attacks considered. Also in this case, it is necessary to explain the exception presented by *SpacePhish* and *Yuan* et al., whose number of queries is equal to zero. Indeed, *SpacePhish* relies on the insertion of fake links or URL manipulations, regardless of the sample analyzed or the target model considered; that is, it does not involve any interaction with the system under attack. However, such a *query-free* approach has some limitations, as highlighted by the high values of P_f obtained on the two datasets, regardless of the features considered. In contrast, Yuan et al. applies the suggested rendering-preserving manipulations in a single batch, requiring no queries. As shown in This strategy substantially improves the performances of *SpacePhish*, achieving average P_f values between 0.26 and 0.44 across datasets and feature sets: well below the standard 50% decision threshold. *RTTG* exhibits notable effectiveness in generating adversarial samples, achieving average P_f values of approximately 0.07-0.08 on the R feature set, and a steady 0.02-0.03 on the broader C set. While the generation time per sample remains reasonably modest (ranging from 2 to 3 minutes on average) and the number of queries is fixed at 36, the values of MN suggest the application of a relatively high number of manipulations per sample, averaging around 22. It is evident that, whilst the system demonstrates a high level of proficiency in generating high-quality adversarial samples, such efficacy is accompanied by an escalated computational demand, due to the significant number of manipulations required per instance. As regards our method, we can observe that the use of BS leads to the overall best performance in terms of P_f, combining a good level of effectiveness with low execution time (51 seconds) and about 100 number of queries (on average) when considering C. The MN value is the lowest among all evaluated methods, with roughly 3 well-targeted manipulations being sufficient to successfully compromise the ML-PWDs under test. Although

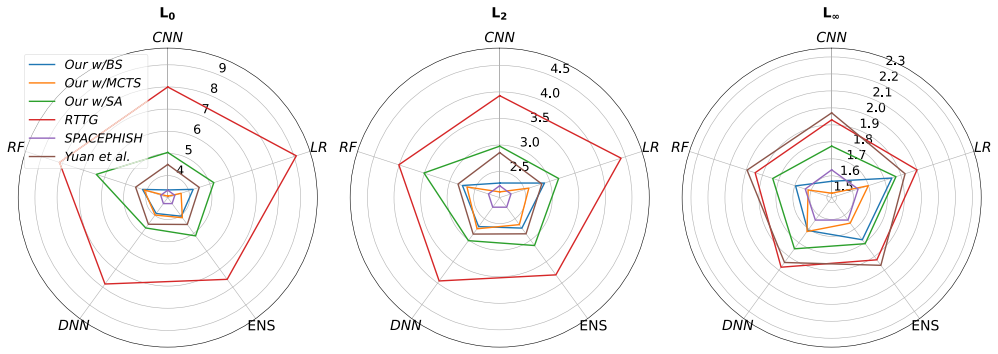

Fig. 5. L_p norms (with $p = 0$, 2, and ∞) for all evaluated attacks against CNN, RF, LR, ENS and DNN targets.

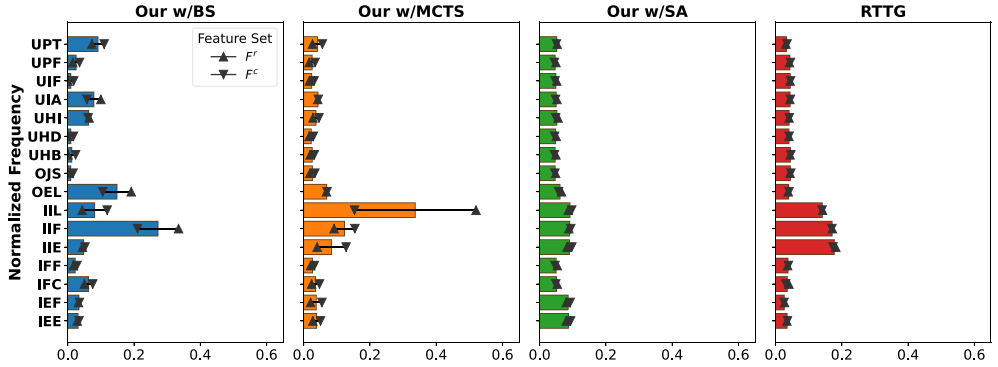

Fig. 6. Frequency of the manipulations applied by our method and RTTG, averaged across the two datasets. The bars report the mean frequencies, with error-bars representing the variability observed between feature sets.

the approach exhibits a higher QN, BS demonstrates a compelling balance of effectiveness and efficiency. On the other hand, SA stands out for its extremely low execution time (5 seconds across both datasets and feature sets) without compromising on the quality and power of the generated adversarial sample. In fact, it made the fewest queries (between 8 and 11) to the target models, while generating highly effective adversarial samples, with P_f values well below the predefined detection threshold. Finally, MCTS emerges as a balanced compromise between the two previously discussed methods, demonstrating its suitability in scenarios where neither execution time nor the number of model queries is a critical constraint. Notably, while achieving P_f values comparable to those of SA (approximately 0.03 on δPhish and 0.06 on Zenodo) it requires significantly fewer manipulations. Specifically, MCTS averages 4-6 manipulations for δPhish and 5-9 for Zenodo, outperforming SA in terms of MN, albeit the substantial requirements in terms of AT. It is worth mentioning that the latter, defined as the average execution time of the attacks, does not exhibit a strictly linear relationship with the number of queries issued (QN). This non-linearity can be attributed to several confounding factors, including the influence of outliers that disproportionately increase (or decrease) the average, the dependence of the feature extraction process (especially for C and its URL-based features) on external services, and the varying structural complexity of web pages, especially in terms of their underlying HTML code.

4.3.5. Variations in the adversarial samples

Since the distance between the original and the corresponding adversarial sample plays a crucial role in the analysis of model robustness [50], further experiments were run to measure the L_p norms [51] quantifying this gap, with p being 0, 2, or ∞. Please note that although the manipulations are discrete, their effect produces continuous variations in the features, as these capture information about the ratios and proportions of HTML elements. The underlying hypothesis is that adversarial samples with both low P_f and minimal feature-space displacement from the original inputs are more effective in compromising model performance than counterparts with equivalent P_f but a higher perturbation norm. Fig. 5 provides a summary of the average L_p norm analysis performed on both datasets. For the sake of clarity, here we only report the results obtained using the restricted feature set R, while those related to C will be discussed later in this section. As can be noticed, our method, in all its variants, consistently falls within the broader region defined by *RTTG* on the radar plots. This visual cue reinforces our claim that, given the same set of manipulations, our approach yields adversarial samples with minimal perturbation norms across different ML-PWDs. Better results in terms of distances are only obtained by *SpacePhish* which, however, as shown in previous experiments, do not allow the attack to perform adequately in terms of either P_f or MN. For the other approaches, we can observe an average trend that generally corresponds to the number of manipulations applied. Specifically, BS, MCTS and SA exhibit increasing values in terms of L_0 and L_2, corresponding to the number of manipulations employed. The exception is L_∞, which shows a different trend (BS > MCTS), likely due to differences in the types of manipulation used. By contrast, the average behavior of Yuan et al. is comparable to that of these methods in terms of L_0 and L_2, but its L_∞ values are similar to those of RTTG, likely reflecting the specific implementation characteristics of this approach.

A further analysis is provided in Fig. 6, where frequencies of the manipulations applied starting from the set defined in Table 2 are reported. While SA exhibits an almost uniform distribution, which is consistent with its implementation, we can observe that both BS and MCTS have distinct patterns. The former displays a broader spread across multiple features, a consequence of the algorithm's level-pruning mechanism, which preserves several top-ranked candidates at each stage of the search. Conversely, the choices made by MCTS tend to follow a

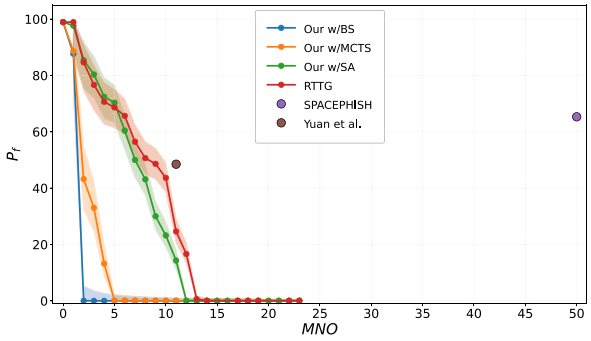

Fig. 7. Comparison between our methods (BS, MCTS, SA) RTTG, SpacePhish and Yuan et al. with a fixed budget of 50 manipulations (note that this constraint does not apply for the latter two). Results beyond 23 manipulations are omitted due to limited informational value.

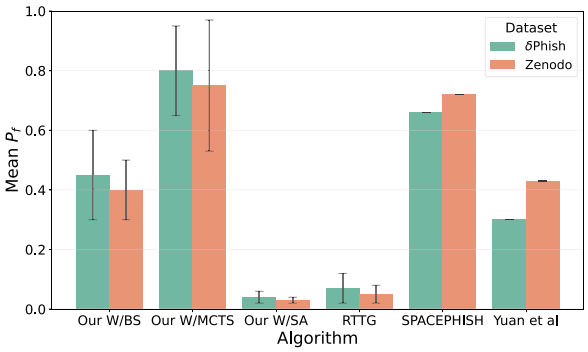

Fig. 8. Comparison between our methods (BS, MCTS, SA) RTTG, SpacePhish and Yuan et al. with a fixed budget of 36 queries. Note that this constraint does not apply for the latter two.

Gaussian-like distribution centered around *IIL*, likely because *IIL* nodes are more frequently visited during MCTS due to their higher associated rewards. These trends, compared with that rather flat of RTTG, confirm the validity of our approach and its ability to make different choices according to the specific objectives of the heuristic optimizations considered.

4.3.6. Comparisons with fixed budgets

In order to provide further insight into the results just discussed, the performance of the different attacks was further compared by fixing the two elements that most characterize their efficiency, namely the *attack budget* (the number of manipulations allowed) and *oracle access* (the number of queries to the target model). The analysis was carried out under identical operating conditions: first by fixing the maximum number of manipulations to 36 (as in RTTG) and then by imposing the same limit to the number of queries. Here, RF_C was selected as the target model, as it showed robust performance in previous analyses, while a random sample of 50 instances drawn from the adversarial examples generated by each method was used as test set.

Results (aggregated for both datasets) are reported in Fig. 7 and indicate that at most 13 manipulations are sufficient to lead the model to misclassification; however, it is evident that the curves corresponding to the proposed approach consistently outperform that of RTTG. In particular, the BS and MCTS methods can severely degrade the performance of the target model with as few as 3 and 5 manipulations, respectively. Similarly, SA effectively identifies a subset of manipulations that has a greater impact than those identified by RTTG, as indicated by the steeper slope of its curve. It is worth highlighting that Yuan et al. and *SpacePhish* are included in the analysis for the aim of completeness and consistency; however, as both employ a fixed number of manipulations, results are not particularly interesting.

These experimental findings further support the discussion presented in Section 3.3, confirming the greater effectiveness of heuristic-based approaches over a simple greedy search strategy. Experiments conducted under a fixed query budget (Fig. 8) provide nearly complementary outcomes. While the results obtained by RTTG and SA closely match the aggregated outcomes reported in Table 8, BS and MCTS exhibit significant performance degradation due to the limited number of queries allowed, which is substantially lower than the average values reported in the same table. In particular, BS is, on average, able to exceed the 0.5 threshold corresponding to a label flip; however, the generated samples remain close to the decision boundary, as indicated by an average P_f value of approximately 0.4. In contrast, MCTS almost never surpasses this threshold, as it requires a substantially larger number of queries by design. These findings provide further insight into the characteristics of the analyzed attacks, emphasizing the importance of application context and operational constraints when selecting an appropriate attack strategy.

5. Explainability analysis

In order to gain a better understanding of the ML-PWD's ability to discern *benign*, *phishing* and deliberately *crafted* adversarial webpages, in this section we present a set of experiments focused on the interpretability of the model's decision-making process. To this aim, eXplainable Artificial Intelligence (XAI) frameworks can bise adopted, as discussed in the extant literature (e.g, [52] and [53]). XAI approaches offer critical insights into the internal mechanics of classification decisions, thereby supporting a deeper understanding of model behavior under adversarial conditions. In particular, given the localized focus of our analysis, we chose SHapley Additive Explanations (SHAP) [54] to determine the contribution of each feature.

In summary, SHAP is founded on principles from cooperative game theory, namely the concept of Shapley values. For each instance, the Shapley values are utilized to quantify the contribution of individual features, such that the sum of these values is precisely equal to the difference between the model's prediction, $f(X)$, and the expected value of the model's output, $E[f(X)]$:

$$f(x) \approx \phi_0 + \sum_{i=0}^{n} \phi_i, \tag{9}$$

where ϕ_0 represents the model baseline, and ϕ_i is the i-th feature contribution. For the sake of brevity, we chose to report the explainability considerations of the most resilient among the employed models, i.e., Random Forest. Firstly, we evaluated the importance of the features assigned by RF to two samples randomly chosen from the δPhish dataset, one from the benign (W_B) and one from the phishing (W_P) classes. SHAP values (Table 9) are computed with reference to the benign class and give an indication of the contribution of each feature in steering the classification towards the phishing (negative SHAP values) or the benign class (positive values). In the case of W_P, for instance, the high SHAP values of the first three features depends on the fact that phishing websites tend to include a high number of non-functional or misleading hyperlinks (*HTML_brokenLnk*), as well as suspicious internal anchor tags (*HTML_nullLnkWeb*); this is also reflected in the values of *HTML_commPage*, which quantifies the ratio of internal to external links. Moreover, a marked difference can be observed in the magnitude and distribution of SHAP values between the two instances: phishing classifications are typically driven by a limited set of highly influential features, whereas benign ones emerge from a wider array of features with modest individual contributions. The same analysis was repeated on the adversarial variants of W_P generated with BS, MCTS, and SA by applying the following manipulations (see Table 2), respectively:

- $\mathbf{W_P}$ w/BS: {*UIA*, *IIF*};
- $\mathbf{W_P}$ w/MCTS:{*UIA*, *IIE*};

Table 9
Feature contributions (green indicating contribution towards benign, red indicating contribution in the opposite direction) to the predicted output for a benign sample, a clean phishing sample and its adversarial variants (ID:17684). The shifts in attribution highlight the effectiveness of perturbations in deceiving the model and undermining its discriminatory performance. ϵ stands for extremely low values, while the final rows reports the probability of each sample being classified as benign.

	$\mathbf{W_B}$	$\mathbf{W_P}$	$\mathbf{W_P}$ w/BS	$\mathbf{W_P}$ w/MCTS	$\mathbf{W_P}$ w/SA
HTML_nullLnkWeb	+0.055	-0.55	+0.12	+0.15	+0.07
HTML_brokenLnk	+0.009	-0.14	-0.11	-0.11	-0.11
HTML_commPage	+0.010	-0.07	-0.11	-0.12	-0.06
URL_length	+0.009	+0.03	+0.03	+0.04	+0.03
HTML_favicon	+0.007	-0.03	-0.05	-0.09	-0.03
URL_shrtWordURL	ϵ	+0.02	ϵ	ϵ	ϵ
HTML_loginForm	ϵ	+0.02	-0.10	-0.07	-0.06
HTML_hiddenInput	+0.011	+0.02	+0.03	ϵ	+0.02
HTML_commPageFoot	ϵ	-0.02	+0.04	ϵ	+0.03
HTML_anchors	+0.017	ϵ	+0.06	+0.07	+0.04
HTML_iframe	ϵ	ϵ	ϵ	+0.06	ϵ
HTML_domCopyright	+0.006	ϵ	ϵ	-0.03	ϵ
Other's contribution	+0.012	-0.02	+0.04	+0.05	+0.04
$E[f(W)]$	0.857	0.857	0.857	0.857	0.857
$f(W)$	**0.993**	0.117	**0.817**	**0.807**	**0.827**

- $\mathbf{W_P}$ w/SA: {*UHI*, *OJS*, *IEE*, *UPF*, *UIF*, *IFC*, *IEE*, *IIE*, *IFF*, *IIE*, *UIA*, *IEF*, *IEF*, *OEL*, *IEE*}.

We can observe that, for the first two strategies, two well-targeted manipulations are sufficient to significantly raise the benign-ness score, while SA applies a total of 15 manipulations to craft the adversarial sample. Then, looking at BS and MCTS only, the manipulation *UIA* is applied by both together with *IIF* and *IIE* respectively. SHAP values attributed to *HTML_anchors*, for instance, are notably influenced specifically by *UIA*, which reduces the proportion of internal suspicious anchors by replacing them with innocuous alternatives; this causes the shift in SHAP contributions. As for the other two manipulations, both share the common goal of injecting internal links to distort features that rely on the internal-to-external link ratio, such as *HTML_commPage* and *HTML_commPageFoot*. In this case, the SHAP values for the former remain negative regardless of the attack strategy, while those related to the latter changed from −0.2 to +0.04 when considering BS. Similar results are observed, focusing on *HTML_brokenLnk* and *HTML_nullLnkWeb*, with the first one remaining stable, and the second moving from a substantial negative contribution of −0.55 to positive values, i.e., 0.12, 0.15, and 0.07 for the three attacks, respectively. Despite the discussed differences in individual feature attributions, the outcomes $f(W)$ remain comparable due to the functional similarity of the manipulations applied. Indeed, all the adversarial variants are capable of successfully tricking the model, causing the probability of the sample being classified as benign to increase from 0.117 to approximately 0.82. Such analysis further confirms that manipulations introduced in the problem space successfully result in semantically valid alterations in the corresponding feature space, while consistently underscoring the relationship between the feasibility of adversarial manipulations and feature engineering. Moreover recent studies have shown that shifts in feature attributions under adversarial perturbations can be leveraged for detection [53] or suppression of adversarially injected information [55]. Collectively, these works demonstrate that explanation-level inconsistencies provide a practical, model-agnostic signal for adversarial defense [56].

6. Defense mechanisms

Previous analyses have demonstrated the ability of adversarial examples to fool the target classification models. However, it is also important to examine how these perturbations influence the underlying latent data distribution, beyond their immediate impact on prediction accuracy. Specifically, analyzing the extent to which adversarial samples deviate from the distribution of legitimate data could provide the basis for defining an effective defense mechanism. Indeed, the identification of samples that fall outside the learned data manifold can provide an additional layer of protection, complementing existing classifiers and strengthening overall resilience against adversarial attacks. To this aim, the ML-PWD was complemented by four distinct anomaly detection algorithms, each leveraging a different statistical or geometric perspective on outlier detection.

Table 10
False Positive Rates (FPR) for different anomaly detection methods on the Zenodo and δPhish datasets, considering the *C* feature set.

Dataset	IF	LOF	OCSVM	EE
Zenodo	0.12	0.07	0.17	0.26
δPhish	0.13	0.09	0.13	0.27

- **Isolation Forest (IF)** [57]: An ensemble-based method that isolates anomalies through recursive random partitioning, exploiting the observation that anomalous points typically require fewer splits to isolate.
- **Local Outlier Factor (LOF)** [58]: A density-based approach that measures the local deviation of a data point with respect to its k-nearest neighbors, identifying anomalies as those with substantially lower local density.
- **One-Class SVM (OCSVM)** [59]: A kernel-based method that estimates the support of the data distribution by learning a decision function that separates the majority of the data from the origin in feature space.
- **Elliptic Envelope (EE)** [60]: Assumes data follows a Gaussian distribution and fits a robust covariance estimate to define a confidence ellipse, flagging observations that lie beyond this boundary as outliers.

These algorithms were further refined through parameter selection using a grid search validated based on the FPR values.

In addition, we adopted two approaches drawn from recent literature:

- **SHAP-ENS**: a variation of the architecture introduced in [53], adapted to the *C* feature scheme, that integrates the SHAP value signatures associated with the samples in this feature set. These signatures are then used to train a soft-voting ensemble model comprising three components: an MLP, a Random Forest and a GradientBoostingClassifier.
- **AAE**: a data augmentation technique based on the deep generative architecture outlined in [61], followed by an XGBoost classifier to detect adversarial samples.

Firstly, the five methods were applied to δPhish and Zenodo datasets, which are normally free from anomalies, and the corresponding False Positive Rates (FPR) are reported as baselines in Table 10. While the FPRs of the first three (IF, LOF, OCSVM) average around 10%, the rates are substantially higher for EE. This is probably due to the underlying data distribution, which is only partially captured by the ellipsoidal assumptions of EE. Conversely, more sophisticated non-linear approaches, such as IF and OCSVM, demonstrate superior adaptability to complex data structures. Moreover, results also suggest that the local nature of LOF is particularly effective in identifying distributional irregularities within the datasets under consideration. Then, we tested the four defense mechanisms against the five attacks under consideration, by evaluating their aggregated detection rate when used to support the five ML-PWDs (CNN, LR, RF, ENS, DNN) trained on the two feature sets *C* and *R*. As illustrated in Fig. 9, the adversarial samples most easily recognized by

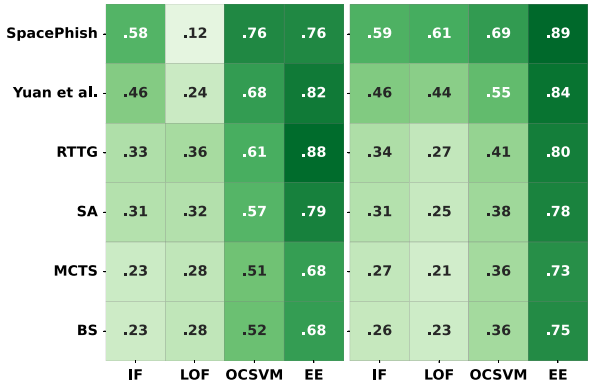

Fig. 9. A comparative analysis of four defense mechanisms based on the percentage of adversarial samples they successfully detected across multiple attack strategies (higher values indicate greater robustness). Detection performance is shown separately for the Zenodo (left) and the δPhish (right) datasets.

Table 11
Comparison of accuracy and F1-score for different attack methods across two evaluation settings.

Attack	SHAP-ENS		AAE	
	Accuracy	F1-score	Accuracy	F1-score
Our W/BS	0.675	0.551	0.096	0.175
Our W/MCTS	0.730	0.653	0.114	0.205
Our W/SA	0.760	0.703	0.082	0.151
RTTG	0.810	0.779	0.021	0.042
SPACEPHISH	0.795	0.745	0.292	0.452
Yuan et. al	0.735	0.662	0.108	0.192

most defense mechanisms are those created by the SpacePhish method, except for LOF on the Zenodo dataset, where only 12% of samples are detected. This depends on the fact that adversarial samples generated in that scenario formed a dense cluster that could be identified as a group of anomalies, but due to its high density, it fooled LOF. Conversely, Yuan et al. exhibits values slightly below those of *SpacePhish*, demonstrating a similar trend and doubling the proportion of samples detected by LOF (24%) on Zenodo. However, it demonstrates correspondingly greater evasiveness on δPhish, with 44% of samples detected compared to 61% for its predecessor. In the case of RTTG, the level of detection remains relatively consistent, with approximately 10%-20% of additional samples being blocked from the defense compared to those created with the proposed approach. The efficacy of all three variants of our heuristic-based attack is evidenced by their ability to maintain an evasion rate of approximately 75% against the IF and LOF defenses, and about 50% to 35% against OCSVM and EE.

With respect to SHAP-ENS e AAE, a clear distinction emerges between the two approaches (Table 11). Firstly, the defensive performance of the SHAP value signature-based solution reveals that the defense mechanism can correctly identify a significant proportion of adversarial samples, notably in the case of RTTG and SpacePhish, with F1-scores of .78 and .75 respectively. However, this trend deteriorates progressively when considering the proposed attack method, which increasingly challenge the defense model. In particular, SA achieves accuracy and F1 of around .70, exhibiting behavior resembling MCTS and Yuan et al., in contrast to approaches such as BS, which make the defense mechanisms useless. Conversely, the augmentation-based mechanism fails to effectively detect the generated adversarial samples. Indeed, across nearly all methods, the adversarial instances remain close to the latent data distribution and are predominantly misclassified as benign, yielding F1 scores in the range $[.17, 0.45]$.

These results reveal that SHAP-based defenses can detect attacks such as *RTTG* and *SpacePhish*, but their effectiveness diminishes considerably when dealing with our proposed method. Conversely, anomaly-based methods are effective at identifying attacks that differ from the phishing class distribution (e.g. SpacePhish and Yuan et al.), yet they struggle with adversarial samples produced by our techniques, as these are more closely aligned with legitimate data. Therefore, the combination of feature-attribution insights and distribution-sensitive strategies seems to effectively contribute to counter sophisticated attacks.

7. Conclusion and future work

In this study, we proposed a novel adversarial attack framework for website phishing detection, based on heuristic search strategies. Our approach formulates the generation of adversarial examples as a constrained optimization problem involving semantic-preserving manipulations, leveraging Beam Search, Simulated Annealing and Monte Carlo Tree Search to efficiently identify effective manipulation sequences. By design, the proposed method balances attack success and efficiency, achieving robust evasion performance while requiring a limited number of manipulations per sample.

Through extensive comparative evaluations against state-of-the-art attacks, we have demonstrated that reducing the number of manipulations applied does not compromise the effectiveness of the attack. In fact, our methods produce adversarial samples that remain closer to the original instances in terms of standard distance metrics, thereby preserving semantic fidelity. Experimental analysis also revealed that different search strategies exhibit distinct trade-offs between manipulation diversity and perturbation magnitude. Frequency analysis indicated that algorithmic design choices and the intrinsic properties of individual manipulations significantly influence their selection during the attack process. Furthermore, integrating an interpretability framework provided semantic validation of the generated perturbations, thereby reinforcing the realism of the proposed attacks.

From a defensive perspective, our results suggest that anomaly-detection-based defenses and more recent approaches can vary considerably in effectiveness. More conservative attack strategies, such as those proposed in this work, tend to achieve markedly lower detection rates than more aggressive, manipulation-heavy approaches, exposing potential blind spots in current defense mechanisms.

Despite these promising results, some limitations exist. Firstly, the evaluation was conducted in a soft-label black-box setting that assumes access to confidence scores, a resource that may not always be available in real-world deployments. Secondly, although the manipulation set is designed to preserve functionality, the set of possible semantic-preserving transformations is inherently incomplete. Nevertheless, these two aspects are common to other approaches addressing AML attack to ML-PWDs; thus, overcoming them requires a global shift in perspective regarding the attack scenario.

Future work could address these limitations in multiple ways. In terms of attacks, extending the framework to hard-label settings would increase its applicability under stricter threat models. In terms of defense, adversarial training using samples generated by our heuristic search methods could enhance model robustness, and multi-modal defense architectures could improve the accuracy with which adversarial examples are detected. Additionally, leveraging explainable AI techniques for adversarial generation is a promising way to produce more targeted, semantically grounded perturbations. Finally, integrating NLP-based techniques for HTML manipulation could facilitate richer, structure-aware transformations that preserve functionality while evading detection systems.

CRediT authorship contribution statement

Giuseppe Lo Re: Writing – original draft, Validation, Formal analysis; **Marco Morana:** Writing – review & editing, Methodology; **Giuseppe Rizzo:** Writing – original draft, Validation, Formal analysis.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Special thanks to Giuseppe Ganci for his support in developing the preliminary evaluation code as part of his Master Thesis work.

References

[1] Anti-Phishing Working Group (APWG). Phishing activity trends report, 2nd quarter 2025. 2025. https://apwg.org/trendreports.
[2] Apruzzese G, Conti M, Yuan Y. SpacePhish: the evasion-space of adversarial attacks against phishing website detectors using machine learning. In: Proceedings of the 38th annual computer security applications conference. ACSAC '22; New York, NY, USA: Association for Computing Machinery. ISBN 9781450397599; 2022, p. 171–85. https://doi.org/10.1145/3564625.3567980
[3] Cortellazzi J, Quiring E, Arp D, Pendlebury F, Pierazzi F, Cavallaro L. Intriguing properties of adversarial ML attacks in the problem space [extended version]. ACM Trans Priv Secur 2025;28(4). https://doi.org/10.1145/3742895
[4] Montaruli B, Demetrio L, Pintor M, Compagna L, Balzarotti D, Biggio B. Raze to the ground: query-efficient adversarial HTML attacks on machine-learning phishing webpage detectors. In: Proceedings of the 16th ACM workshop on artificial intelligence and security. AISec '23; New York, NY, USA: Association for Computing Machinery. 2023, p. 233–44. https://doi.org/10.1145/3605764.3623920
[5] Prakash P, Kumar M, Kompella RR, Gupta M. Phishnet: predictive blacklisting to detect phishing attacks. In: Proceedings of the 29th conference on information communications. INFOCOM'10; IEEE Press. ISBN 9781424458363; 2010, p. 346–50. https://doi.org/10.1109/INFCOM.2010.5462216
[6] Bahnsen AC, Bohorquez EC, Villegas S, Vargas J, González FA. Classifying phishing URLs using recurrent neural networks. In: 2017 APWG Symposium on electronic crime research (eCrime). 2017, p. 1–8. https://doi.org/10.1109/ECRIME.2017.7945048
[7] Wang M, Zang X, Cao J, Zhang B, Li S. Phishhunter: detecting camouflaged IDN-based phishing attacks via siamese neural network. Comput Secur 2024;138:103668. https://doi.org/10.1016/j.cose.2023.103668
[8] Prasad YB, Dondeti V. Pdsmv3-dcrnn: a novel ensemble deep learning framework for enhancing phishing detection and url extraction. Comput Secur 2025;148:104123. https://doi.org/10.1016/j.cose.2024.104123
[9] Opara C, Chen Y, Wei B. Look before you leap: detecting phishing web pages by exploiting raw URL and HTML characteristics. Expert Syst Appl 2024;236:121183. https://doi.org/10.1016/j.eswa.2023.121183
[10] Niakanlahiji A, Chu B-T, Al-Shaer E. Phishmon: a machine learning framework for detecting phishing webpages. In: 2018 IEEE International conference on intelligence and security informatics (ISI). 2018, p. 220–5. https://doi.org/10.1109/ISI.2018.8587410
[11] Wenyin L, Huang G, Xiaoyue L, Min Z, Deng X. Detection of phishing webpages based on visual similarity. In: Special interest tracks and posters of the 14th international conference on world wide web. WWW '05. ISBN 1595930515; 2005, p. 1060–1. https://doi.org/10.1145/1062745.1062868
[12] Haruta S, Asahina H, Sasase I. Visual similarity-based phishing detection scheme using image and CSS with target website finder. In: Proceedings of the 2017 IEEE global communications conference. GLOBECOM'17; 2017, p. 1–6. https://doi.org/10.1109/GLOCOM.2017.8254506
[13] Wang M, Song L, Li L, Zhu Y, Li J. Phishing webpage detection based on global and local visual similarity. Expert Syst Appl 2024;252:124120. https://doi.org/10.1016/j.eswa.2024.124120
[14] Lee J, Lim P, Hooi B, Divakaran DM. Multimodal large language models for phishing webpage detection and identification. In: Proceedings of the APWG symposium on electronic crime research (ecrime). 2024, p. 1–13. https://doi.org/10.1109/eCrime66200.2024.00007
[15] Bahnsen AC, Torroledo I, Camacho LD, Villegas S. Deepphish: simulating malicious AI. In: Proceedings of the 2018 APWG symposium on electronic crime research (eCrime). 2018,https://api.semanticscholar.org/CorpusID:51691528.
[16] Nazzal M, Khalil I, Khreishah A, Phan N, Ma Y. Multi-instance adversarial attack on GNN-based malicious domain detection. In: Proceedings of the 2024 IEEE symposium on security and privacy (SP). 2024, p. 1236–54. https://doi.org/10.1109/SP54263.2024.00006
[17] Song F, Lei Y, Chen S, Fan L, Liu Y. Advanced evasion attacks and mitigations on practical ML-based phishing website classifiers. Int J Intell Syst 2021;36(9):5210–40. https://doi.org/10.1002/int.22510
[18] Gressel G, Hegde N, Sreekumar A, Radhakrishnan R, Harikumar K, Anjali S, et al. Feature importance guided attack: A model agnostic adversarial attack. 2023. arXiv:2106.14815;
[19] O'Mara A, Alsmadi I, AlEroud A. Generative adverserial analysis of phishing attacks on static and dynamic content of webpages. In: 2021 IEEE International confernce on parallel & distributed processing with applications, big data & cloud computing, sustainable computing & communications, social computing & networking (ISPA/BDCloud/SocialCom/SustainCom). 2021, p. 1657–62. https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00222
[20] Yuan Y, Hao Q, Apruzzese G, Conti M, Wang G. "Are adversarial phishing webpages a threat in reality?" understanding the users' perception of adversarial webpages. In: Proceedings of the ACM web conference 2024. WWW '24; New York, NY, USA: Association for Computing Machinery. ISBN 9798400701719; 2024, p. 1712–23. https://doi.org/10.1145/3589334.3645502
[21] Lee J, Xin Z, See M NP, Sabharwal K, Apruzzese G, Divakaran DM. Attacking logo-based phishing website detectors with adversarial perturbations. In: Computer security – ESORICS 2023: 28th European symposium on research in computer security, the hague, the Netherlands, september 25–29, 2023, Proceedings, Part III. Springer-Verlag. ISBN 978-3-031-51478-4; 2023, p. 162–82. https://doi.org/10.1007/978-3-031-51479-1_9
[22] Hao Q, Diwan N, Yuan Y, Apruzzese G, Conti M, Wang G. It doesn't look like anything to me: using diffusion model to subvert visual phishing detectors. In: Proceedings of the 33rd USENIX security symposium (USENIX security 24). Philadelphia, PA: USENIX Association. ISBN 978-1-939133-44-1; 2024, p. 3027–44. https://www.usenix.org/conference/usenixsecurity24/presentation/hao-qingying.
[23] Roy SS, Thota P, Naragam KV, Nilizadeh S. From chatbots to phishbots?: phishing scam generation in commercial large language models. In: Proceedings of the 2024 IEEE symposium on security and privacy (SP). 2024, p. 36–54. https://doi.org/10.1109/SP54263.2024.00182
[24] Kulkarni A, Balachandran V, Divakaran DM, Das T. From ML to LLM: evaluating the robustness of phishing web page detection models against adversarial attacks. Digit Threats 2025;6(2). https://doi.org/10.1145/3737295
[25] Yuan Y, Apruzzese G, Conti M. Beyond the west: revealing and bridging the gap between western and chinese phishing website detection. Comput Secur 2025;148:104115. https://doi.org/10.1016/j.cose.2024.104115
[26] Jabir R, Le J, Nguyen C. Phishing attacks in the age of generative artificial intelligence: a systematic review of human factors. AI 2025;6(8). https://doi.org/10.3390/ai6080174
[27] Langford T, Payne B. Phishing faster: implementing chatGPT into phishing campaigns. In: Proceedings of the future technologies conference (FTC); vol. 1. ISBN 978-3-031-47454-5; 2023, p. 174–87. https://doi.org/10.1007/978-3-031-47454-5_13
[28] Zhang X, Zhou Y, Pei S, Zhuge J, Chen J. Adversarial examples detection for XSS attacks based on generative adversarial networks. IEEE Access 2020;8:10989–96. https://doi.org/10.1109/ACCESS.2020.2965184
[29] Guo X, Su R, Tu S, Xu L. Searching for textual adversarial examples with learned strategy. In: Neural information processing. ICONIP 2022. communications in computer and information science, Vol 1791. Springer Nature Singapore; 2023, p. 385–96. https://doi.org/10.1007/978-981-99-1639-9_32
[30] Boutsikas J, Eren ME, Varga CK, Raff E, Matuszek C, Nicholas C. Evading malware classifiers via monte carlo mutant feature discovery. In: 12th Annual Malware technical exchange meeting. 2021, arXiv:2106.07860
[31] Wu S, Wang H, Zhao Y, Wu X, Zheng Y, Li W, et al. Monte carlo tree search based prompt autogeneration for jailbreak attacks against LLMs. In: Proceedings of the 31st international conference on computational linguistics. Abu Dhabi, UAE: Association for Computational Linguistics; 2025, p. 1057–68. https://aclanthology.org/2025.coling-main.71/.
[32] Zhu H, Zhao Q,Wu Y. BeamAttack: generating high-quality textual adversarial examples through beam search and mixed semantic spaces. In: Advances in knowledge discovery and data mining: 27th Pacific-Asia conference on knowledge discovery and data mining, PAKDD 2023, Osaka, Japan, May 25–28, 2023, Proceedings, Part II. Springer-Verlag; 2023, p. 454–65. https://doi.org/10.1007/978-3-031-33377-435
[33] Choi Y, Kim H, Lee J-H. TABS: Efficient textual adversarial attack for pre-trained NL code model using semantic beam search. In: Proceedings of the 2022 conference on empirical methods in natural language processing. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics; 2022, p. 5490–8. https://doi.org/10.18653/v1/2022.emnlp-main.369
[34] Zhang H, Pun CM, Du X. Reversible adversarial image examples with beam search attack and grayscale invariance. In: International workshop on advanced imaging technology (IWAIT) 2024; Vol. 13164. International Society for Optics and Photonics; SPIE; 2024, p. 1316410. https://doi.org/10.1117/12.3018262
[35] Yang X, Liu W, Tao D, Liu W. Besa: bert-based simulated annealing for adversarial text attacks. In: Proceedings of the thirtieth international joint conference on artificial intelligence, IJCAI-21. International Joint Conferences on Artificial Intelligence Organization; 2021, p. 3293–9. Main Track. https://doi.org/10.24963/ijcai.2021/453
[36] Williams PN, Li K. CamoPatch: an evolutionary strategy for generating camoflauged adversarial patches. In: Advances in neural information processing systems 36: annual conference on neural information processing systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10, - 16, 2023. 2023.
[37] Liu S, Luo Y. Score-based black-box adversarial attack on time series using simulated annealing classification and post-processing based defense. Electronics 2024;13(3). https://doi.org/10.3390/electronics13030650
[38] Yuan Y, Apruzzese G, Conti M. Multi-spacephish: extending the evasion-space of adversarial attacks against phishing website detectors using machine learning. Digit Threats 2024;5(2). https://doi.org/10.1145/3638253
[39] undefinedwiechowski M, Godlewski K, Sawicki B, Mańdziuk J. Monte Carlo tree search: a review of recent modifications and applications. Artif Intell Rev 2022;56(3):2497–562. https://doi.org/10.1007/s10462-022-10228-y

[40] Kocsis L, Szepesvári C. Bandit based Monte-Carlo planning. In: Proceedings of the 17th European conference on machine learning. ECML'06; Springer-Verlag; 2006, p. 282–93. https://doi.org/10.1007/11871842_29
[41] Peng SIOW, Morton TE. Filtered beam search in scheduling. Int J Prod Res 1988;26(1):35–62. https://doi.org/10.1080/00207548808947840
[42] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80. https://doi.org/10.1126/science.220.4598.671
[43] Hastings WK, et al. Monte Carlo sampling methods using Markov chains and their applications. Biometrika 1970;57(1):97–109. https://doi.org/10.1093/biomet/57.1.97
[44] Corona I, Biggio B, Contini M, Piras L, Corda R, Mereu M, et al. DeltaPhish: detecting phishing webpages in compromised websites. In: Computer security – ESORICS 2017: 22nd European symposium on research in computer security, Oslo, Norway, September 11–15, 2017. Springer International Publishing; 2017, p. 370–88. https://doi.org/10.1007/978-3-319-66402-6_22
[45] Putra I K AA. Phishing website dataset. 2023. https://doi.org/10.5281/zenodo.8041386
[46] Mohammad RM, Thabtah F, McCluskey L. Intelligent rule-based phishing websites classification. IET Inf Secur 2014;8(3):153–60. https://doi.org/10.1049/iet-ifs.2013.0202
[47] Sharma SR, Parthasarathy R, Honnavalli PB. A feature selection comparative study for web phishing datasets. In: 2020 IEEE International conference on electronics, computing and communication technologies (CONECCT). 2020, p. 1–6. https://doi.org/10.1109/CONECCT50063.2020.9198349
[48] Jain AK, Gupta BB. Towards detection of phishing websites on client-side using machine learning based approach. Telecommun Syst 2018;68. https://doi.org/10.1007/s11235-017-0414-0
[49] Hannousse A, Yahiouche S. Towards benchmark datasets for machine learning based website phishing detection: an experimental study. Eng Appl Artif Intell 2021;104:104347. https://doi.org/10.1016/j.engappai.2021.104347
[50] Carlini N, Wagner D. Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on security and privacy (SP). Los Alamitos, CA, USA: IEEE Computer Society; 2017, p. 39–57. https://doi.org/10.1109/SP.2017.49
[51] Aggarwal CC, Hinneburg A, Keim DA. On the surprising behavior of distance metrics in high dimensional space. In: Proceedings of the 8th international conference on database theory (ICDT 2001). Springer Berlin Heidelberg; 2001, p. 420–34. https://doi.org/10.1007/3-540-44503-X_27
[52] Imran M, Appice A, Malerba D. Evaluating realistic adversarial attacks against machine learning models for windows PE malware detection. Fut Internet 2024;16(5). https://doi.org/10.3390/fi16050168
[53] Fidel G, Bitton R, Shabtai A. When explainability meets adversarial learning: detecting adversarial examples using SHAP signatures. In: In proceedings of the 2020 international joint conference on neural networks (IJCNN). 2020, p. 1–8. https://doi.org/10.1109/IJCNN48605.2020.9207637
[54] Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. In: Proceedings of the 31st international conference on neural information processing systems. NIPS'17; Curran Associates Inc. ISBN 9781510860964; 2017, p. 4768–77. https://dl.acm.org/doi/10.5555/3295222.3295230.
[55] Wang J, Dong P, Zhou F, Wu Q. Information importance-aware defense against adversarial attack for automatic modulation classification: an XAI-based approach. In: Proceedings of the 16th international conference on wireless communications and signal processing (WCSP). 2024, p. 626–31. https://doi.org/10.1109/WCSP62071.2024.10827616
[56] Baniecki H, Biecek P. Adversarial attacks and defenses in explainable artificial intelligence: a survey. Inf Fusion 2024;107:102303. https://doi.org/https://doi.org/10.1016/j.inffus.2024.102303
[57] Liu FT, Ting KM, Zhou Z-H. Isolation-based anomaly detection. ACM Trans Knowl Discov Data 2012;6(1). https://doi.org/10.1145/2133360.2133363
[58] Breunig MM, Kriegel H-P, Ng RT, Sander J. Lof: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data. SIGMOD '00; New York, NY, USA: Association for Computing Machinery. ISBN 1581132174; 2000, p. 93–104. https://doi.org/10.1145/342009.335388
[59] Hejazi M, Singh YP. One-class support vector machines approach to anomaly detection. Appl Artif Intell 2013;27(5):351–66. https://doi.org/10.1080/08839514.2013.785791
[60] Rousseeuw PJ, Driessen KV. A fast algorithm for the minimum covariance determinant estimator. Technometrics 1999;41(3):212–23. https://doi.org/10.2307/1270566
[61] Shirazi H, Muramudalige SR, Ray I, Jayasumana AP, Wang H. Adversarial autoencoder data synthesis for enhancing machine learning-based phishing detection algorithms. IEEE Trans Serv Comput 2023;16(4):2411–22. https://doi.org/10.1109/TSC.2023.3234806