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

Giovanni Apruzzese, Mauro Conti, Ying Yuan

Annual Computer Security Applications Conference

December 7th, 2022
Austin, TX, USA
In the adversarial ML domain, have you ever read a research paper showing an attack that has an effectiveness of 3%?
Current Landscape of Phishing

- Phishing attacks are continuously increasing
- Most detection methods still rely on *blocklists* of malicious URLs
  - These detection techniques can be *evaded easily* by “squatting” phishing websites!

Current Landscape of Phishing – Countermeasures

- Countering such simple (but effective) strategies can be done via *data-driven* methods.
Current Landscape of Phishing – Countermeasures (ML)

- Countering such simple (but effective) strategies can be done via *data-driven* methods

![Data-driven methods diagram](image)

- Such methods (obviously 😊) include (also) Machine Learning techniques:

![Machine Learning diagram](image)

- Machine Learning-based Phishing Website Detectors (ML-PWD) are very effective [1]
  - Even popular products and web-browsers (e.g., Google Chrome) use them! [2]

---

Phishing in a nutshell

- Phishing websites are taken down quickly
  - The moment they are reported in a blocklist, they become useless
- Even if a victim lands on a phishing website, the phishing attempt is not complete
  - The victim may be “hooked”, but they are not “phished” yet!

Most phishing attacks end up in failure [3]
Phishing in a nutshell (cont’d)

- Phishing websites are taken down quickly
  - The moment they are reported in a blocklist, they become useless

- Even if a victim lands on a phishing website, the phishing attempt is not complete
  - The victim may be “hooked”, but they are not “phished” yet!

Most phishing attacks end up in failure [3]

- Phishers are well aware of this fact... but they (clearly) keep doing it
  - Hence, they “have to” evade detection mechanisms

(Remember: Real attackers operate with a cost/benefit mindset [4])
Problem Statement: Adversarial Attacks against ML-PWD

- ML-PWD are good but...
- ...the detection of ML methods can be bypassed via (adversarial) evasion attacks!
- Adversarial Attacks exploit a perturbation, $\varepsilon$, that induces an ML model, $\mathcal{M}$, to misclassify a given input, $F_x$, by producing an incorrect output ($y_x^\varepsilon$ instead of $y_x$)

$$\text{find } \varepsilon \text{ s.t. } \mathcal{M}(F_x) = y_x^\varepsilon \neq y_x$$
Problem Statement: Adversarial Attacks against ML-PWD

- ML-PWD are good but...
- ...the detection of ML methods can be bypassed via (adversarial) evasion attacks!
- Adversarial Attacks exploit a perturbation, $\varepsilon$, that induces an ML model, $\mathcal{M}$, to misclassify a given input, $F_x$, by producing an incorrect output ($y_x^\varepsilon$ instead of $y_x$)
**Problem Statement: Adversarial Attacks against ML-PWD**

- ML-PWD are good but...
- ...the detection of ML methods *can* be bypassed via (adversarial) *evasion* attacks!
- Adversarial Attacks exploit a perturbation, $\varepsilon$, that induces an ML model, $\mathcal{M}$, to misclassify a given input, $F_x$, by producing an incorrect output ($y_{x}^{\varepsilon}$ instead of $y_x$)

- In the context of a ML-PWD, such *perturbation* can be introduced in three ‘spaces’:
Problem Statement: Adversarial Attacks against ML-PWD

- ML-PWD are good but...
- ...the detection of ML methods can be bypassed via (adversarial) evasion attacks!
- Adversarial Attacks exploit a perturbation, $\varepsilon$, that induces an ML model, $\mathcal{M}$, to misclassify a given input, $F_x$, by producing an incorrect output ($y_x^\varepsilon$ instead of $y_x$)

- In the context of a ML-PWD, such perturbation can be introduced in three ‘spaces’: 

![Diagram](image_url)
Problem Statement: Adversarial Attacks against ML-PWD

- ML-PWD are good but...
- ...the detection of ML methods can be bypassed via (adversarial) evasion attacks!
- Adversarial Attacks exploit a perturbation, $\epsilon$, that induces an ML model, $\mathcal{M}$, to misclassify a given input, $F_x$, by producing an incorrect output ($y_x^\epsilon$ instead of $y_x$)

- In the context of a ML-PWD, such perturbation can be introduced in three ‘spaces’:

---

Question: Which ‘space’ do you think an attacker is most likely to use?
Website-space Perturbations (WsP) in practice – original example

Figure 4: An exemplary (and true) Phishing website, whose URL is https://www.63y3hfh-fj39f30-f30if0f-f392.weebly.com/.
Website-space Perturbations (WsP) in practice – changing the URL

Website-space Perturbations (WsP) in practice – changing the HTML

```
<form enctype="multipart/form-data" action="/www.weebly.com/weebly/apps/formSubmit.php" method="POST" id="form-723155629711391878">
  <div id="form-723155629711391878-form-parent" class="wsite-form-container" style="margin-top:10px;">
    <ul class="formlist id="723155629711391878-form-list">
      <div class="wsite-form-field" style="margin:5px 0px 5px 0px;">
        <label class="wsite-form-label" for="input-227982018179653776">Email Address <span class="form-not-required">*</span></label>
        <div class="wsite-form-input-container">
          <input id="input-227982018179653776" class="wsite-form-input wsite-input wsite-input-width-370px" type="text" name="u227982018179653776" />
        </div>
      </div>
    </ul>
  </div>
</form>
```

$\mathcal{E} \text{ (WsP)}$
Website-space Perturbations (WsP) in practice – changing URL+HTML


ε (WsP)
Why do we need all of this anyway? (first reason)

“The paper focuses on test-time evasion attacks in the so-called problem space, where the challenge lies in modifying real input-space objects that correspond to an adversarial feature vector. The main challenge resides in the inverse feature-mapping problem since in many settings it is not possible to convert a feature vector into a problem-space object because the feature mapping function is neither invertible nor differentiable.”
Why do we need all of this anyway? (first reason) [cont’d]

2020 IEEE Symposium on Security and Privacy

Intriguing Properties of Adversarial ML Attacks in the Problem Space

Fabio Pierazzi*,†, Feargus Pendlebury*†‡§, Jacopo Cortellazzi†, Lorenzo Cavallaro†
† King’s College London, ‡ Royal Holloway, University of London, § The Alan Turing Institute

“This paper focuses on test-time evasion attacks in the so-called **problem space**, where the challenge lies in modifying real input-space objects that correspond to an adversarial feature vector. The main challenge resides in the **inverse feature-mapping** problem since in many settings it is not possible to convert a feature vector into a problem-space object because the feature mapping function is neither invertible nor differentiable.”

- This observation is well-founded, however...
- ...if the attacker has access to the feature space, then such “problem” does not apply.

*Perturbations* in the feature space are **not unrealistic**: they simply require the attacker to compromise the ML system.
- This is possible [5], but it has a high cost!
- All past work considering “feature space” perturbations can be made valuable by assuming that the attack has a higher cost!

Why do we need all of this anyway? (second reason)

- Most existing work in the ML-PWD domain has shortcomings, among which:
  - Some craft perturbations in the “feature” space (not impossible, but costly!)
  - Others assume strong attackers (full knowledge, or massive queries)
    - Liang et al. [57] took days!
  - No statistical validation (crucial for a fair evaluation!)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Liang [57]</td>
<td>2016</td>
<td>Problem</td>
<td>$F^c$</td>
<td>SL</td>
<td>$\times$</td>
<td>1 (X)</td>
<td>X</td>
</tr>
<tr>
<td>Corona [30]</td>
<td>2017</td>
<td>Feature</td>
<td>$F^r, F^c$</td>
<td>SL</td>
<td>$\checkmark$</td>
<td>1 (✓)</td>
<td>X</td>
</tr>
<tr>
<td>Bahnsen [20]</td>
<td>2018</td>
<td>Problem</td>
<td>$F^u$</td>
<td>DL</td>
<td>$\times$</td>
<td>1 (X)</td>
<td>X</td>
</tr>
<tr>
<td>Shirazi [79]</td>
<td>2019</td>
<td>Feature</td>
<td>$F^c$</td>
<td>SL</td>
<td>$\times$</td>
<td>4 (✓)</td>
<td>✓*</td>
</tr>
<tr>
<td>Sabir [77]</td>
<td>2020</td>
<td>Problem</td>
<td>$F^u$</td>
<td>SL, DL</td>
<td>$\checkmark$</td>
<td>1 (X)</td>
<td>X</td>
</tr>
<tr>
<td>Lee [55]</td>
<td>2020</td>
<td>Feature</td>
<td>$F^c$</td>
<td>SL</td>
<td>$\checkmark$</td>
<td>1 (✓)</td>
<td>X</td>
</tr>
<tr>
<td>Abdelnabi [8]</td>
<td>2020</td>
<td>Problem</td>
<td>$F^r$</td>
<td>DL</td>
<td>$\checkmark$</td>
<td>1 (✓)</td>
<td>X</td>
</tr>
<tr>
<td>Song [81]</td>
<td>2021</td>
<td>Problem</td>
<td>$F^c$</td>
<td>SL</td>
<td>$\checkmark$</td>
<td>1 (✓)</td>
<td>X</td>
</tr>
<tr>
<td>Bac [18]</td>
<td>2021</td>
<td>Feature</td>
<td>$F^u$</td>
<td>SL, DL</td>
<td>$\times$</td>
<td>1 (X)</td>
<td>X</td>
</tr>
<tr>
<td>Lin [59]</td>
<td>2021</td>
<td>Feature</td>
<td>$F^c$</td>
<td>DL</td>
<td>$\checkmark$</td>
<td>1 (✓)</td>
<td>X</td>
</tr>
<tr>
<td>O’Mara [67]</td>
<td>2021</td>
<td>Feature</td>
<td>$F^r$</td>
<td>SL</td>
<td>$\checkmark$</td>
<td>1 (✓)</td>
<td>X</td>
</tr>
<tr>
<td>Al-Qurashi [10]</td>
<td>2021</td>
<td>Feature</td>
<td>$F^u, F^c$</td>
<td>SL, DL</td>
<td>$\times$</td>
<td>4 (✓)</td>
<td>X</td>
</tr>
<tr>
<td>Gressel [36]</td>
<td>2021</td>
<td>Feature</td>
<td>$F^c$</td>
<td>SL, DL</td>
<td>$\checkmark$</td>
<td>1 (X)</td>
<td>X</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td></td>
<td>Both</td>
<td>$F^u, F^r, F^c$</td>
<td>DL, SL</td>
<td>$\checkmark$</td>
<td>2 (✓)</td>
<td>✓</td>
</tr>
</tbody>
</table>

What is the true impact of realistic adversarial attacks against ML-PWD?
Evaluation – Workflow

- Are “cheap” perturbations (i.e., blind WsP) effective? Let’s assess their impact!
- First, we develop proficient ML-PWD (high $tpr$, low $fpr$)
Evaluation – Baseline

- Are “cheap” perturbations (i.e., blind WsP) effective? Let’s assess their impact!
- First, we develop proficient ML-PWD (high \( tpr \), low \( fpr \))

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

<table>
<thead>
<tr>
<th>( \mathcal{A} )</th>
<th>( F )</th>
<th>Zenodo</th>
<th>( \delta \text{phish} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( tpr )</td>
<td>( fpr )</td>
<td>( tpr )</td>
</tr>
<tr>
<td>\text{CN}</td>
<td>( F^u )</td>
<td>0.96±0.008</td>
<td>0.021±0.0077</td>
</tr>
<tr>
<td></td>
<td>( F^r )</td>
<td>0.88±0.018</td>
<td>0.155±0.0165</td>
</tr>
<tr>
<td></td>
<td>( F^c )</td>
<td>0.97±0.006</td>
<td>0.018±0.0088</td>
</tr>
<tr>
<td>\text{RF}</td>
<td>( F^u )</td>
<td>0.98±0.004</td>
<td>0.007±0.0055</td>
</tr>
<tr>
<td></td>
<td>( F^r )</td>
<td>0.93±0.013</td>
<td>0.025±0.0118</td>
</tr>
<tr>
<td></td>
<td>( F^c )</td>
<td>0.98±0.006</td>
<td>0.007±0.0046</td>
</tr>
<tr>
<td>\text{LR}</td>
<td>( F^u )</td>
<td>0.95±0.009</td>
<td>0.037±0.0100</td>
</tr>
<tr>
<td></td>
<td>( F^r )</td>
<td>0.82±0.017</td>
<td>0.144±0.0171</td>
</tr>
<tr>
<td></td>
<td>( F^c )</td>
<td>0.96±0.007</td>
<td>0.025±0.0077</td>
</tr>
</tbody>
</table>

- Results comparable to the state-of-the-art 😊
- Let’s attack such ML-PWD
  - The \( tpr \) will decrease!
Results – Are WsP effective?

- In some cases, NO
  - This is significant because most past studies show ML-PWD being bypassed “regularly”!
- In some cases, VERY LITTLE
  - This is also significant, because even a 3% decrease in detection rate can be problematic when dealing with thousands of samples!
- In other cases (not shown here), YES
  - This is very significant, because WsP are cheap and are likely to be exploited by attackers
Results – What about attacks in the other spaces?

In general, attacks in the other spaces (via PsP and MsP) are more disruptive...

However, such attacks also have a higher cost! Will real attackers truly use them just to evade a ML-PWD?
Demonstration: competition-grade ML-PWD

- [https://spacephish.github.io](https://spacephish.github.io) ([https://tinyurl.com/spacephish-demo](https://tinyurl.com/spacephish-demo))
Demonstration: competition-grade ML-PWD

- [https://spacephish.github.io](https://spacephish.github.io)
- [https://nbviewer.org/github/hihey54/acsac22_spacephish/blob/main/mlsec_folder/mlsec_artifact-manipulate.ipynb](https://nbviewer.org/github/hihey54/acsac22_spacephish/blob/main/mlsec_folder/mlsec_artifact-manipulate.ipynb)

```python
def websiteAttacks_html(in_html,string,num):
    ind=in_html.find('</body>')
    content=""
    for i in range(0, num):
        content=content+string
    out_html=in_html[:ind]+content+in_html[ind:]
    return out_html
```

```python
In [6]:  # TEST ORIGINAL
with open(original_file)
    original_data = f.
    original_response = re
    print(original_response)

    {
    "n_models": 8,
    "p_mod_00": 0.891,
    "p_mod_01": 0.811,
    "p_mod_02": 0.891,
    "p_mod_03": 0.811,
    "p_mod_04": 0.806,
    "p_mod_05": 0.741,
    "p_mod_06": 0.806,
    "p_mod_07": 0.741
    }

In [8]:  # TEST ADVERSARIAL
with open(output_file)
    adversarial_data = f.
    adversarial_response = re
    print(adversarial_response)

    {
    "n_models": 8,
    "p_mod_00": 0.426,
    "p_mod_01": 0.794,
    "p_mod_02": 0.426,
    "p_mod_03": 0.794,
    "p_mod_04": 0.864,
    "p_mod_05": 0.774,
    "p_mod_06": 0.794,
    "p_mod_07": 0.741
    }
```
Demonstration: competition-grade ML-PWD

- [https://spacephish.github.io](https://tinyurl.com/spacephish-demo)
- [https://nbviewer.org/github/hihey54/acsac22_spacephish/blob/main/mlsec_folder/mlsec_artifact-manipulate.ipynb](https://nbviewer.org/github/hihey54/acsac22_spacephish/blob/main/mlsec_folder/mlsec_artifact-manipulate.ipynb)

```python
def websiteAttacks_html(in_html,string,num):
    ind=in_html.find('</body>')
    content=""
    for i in range(0, num):
        content=content+string
    out_html=in_html[:ind]+content+in_html[ind:]
    return out_html
```

```
In [6]: # TEST ORIGINAL
    
    with open(original_file):
        original_data = f.read()
        original_response = response
        print(original_response)

    {
        "n_models": 8,
        "p_model_00": 0.891,
        "p_model_01": 0.811,
        "p_model_02": 0.891,
        "p_model_03": 0.811,
        "p_model_04": 0.891,
        "p_model_05": 0.741,
        "p_model_06": 0.806,
        "p_model_07": 0.741
    }

In [8]: # TEST ADVERSARIAL
    
    with open(output_file):
        adversarial_data = adversarial
        adversarial_response = response
        print(adversarial_response)

    {
        "n_models": 8,
        "p_model_00": 0.426,
        "p_model_01": 0.891,
        "p_model_02": 0.426,
        "p_model_03": 0.794,
        "p_model_04": 0.864,
        "p_model_05": 0.774,
        "p_model_06": 0.794,
        "p_model_07": 0.741
    }
```
Demonstration: competition-grade ML-PWD

- [https://spacephish.github.io](https://spacephish.github.io) ([https://tinyurl.com/spacephish-demo](https://tinyurl.com/spacephish-demo))
- [https://nbviewer.org/github/hihey54/acsac22_spacephish/blob/main/mlsec_folder/mlsec_artifact-manipulate.ipynb](https://nbviewer.org/github/hihey54/acsac22_spacephish/blob/main/mlsec_folder/mlsec_artifact-manipulate.ipynb)

```python
def websiteAttacks_html(in_html, string, num):
    ind = in_html.find('</body>')</body>''
```

**Review #6C**  
2 Oct 2022

**Overall merit**

3. Reusable

**Comments**

The code and dataset are well documented in the repo. The scripts and dataset are easily reused. All the questions are included in the repo. The results are consistent with the paper, the supplementary file, and the repo's result files.
SpacePhish: The Evasion-space of Adversarial Attacks against Phishing Website Detectors using Machine Learning

Giovanni Apruzzese, Mauro Conti, Ying Yuan