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

Giovanni Apruzzese, Mauro Conti, Ying Yuan
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!

Image source: [https://www.tessian.com/blog/phishing-statistics-2020/](https://www.tessian.com/blog/phishing-statistics-2020/)
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

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

- 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]

---

[1]: Tian, Ke, et al. "Needle in a haystack: Tracking down elite phishing domains in the wild." Internet Measurement Conference 2018.
[2]: El Kouari, Oumaima, Hafssa Benaboud, and Saïda Lazaar. "Using machine learning to deal with Phishing and Spam Detection: An overview." International Conference on Networking, Information Systems & Security. 2020.
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])

[3] Adam Oest, et al “Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale.” In Proc. USENIX Secur. Symp. (2020)
[4] Kelce S Wilson and Müge Ayse Kiy. 2014. Some fundamental Cybersecurity concepts. IEEE Access (2014).
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 $y_x$)

Website

Website space

Preprocessing space

Machine Learning space

Output space

Website-based Phishing Website Detector

Feature set $F$

Feature Extraction

$F_x$

ML model $\mathcal{M}$

$\mathcal{M}(F_x)$

Output space

Benign

Phishing

ACSAC’22 – Dec. 7th, 2022
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 ($y^\varepsilon_x$ 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’:
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 ($y^\varepsilon_x$ 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

https://www.63y3hfh-fj39f30-f30if0f-f392.weebly.com/ → https://www.legitimate123.weebly.com/
Website-space Perturbations (WsP) in practice – changing the HTML

\[ \varepsilon \text{ (WsP)} \]
Website-space Perturbations (WsP) in practice – changing URL+HTML

https://www.63y3hfh-fj39f30-f30if0f-f392.weebly.com/ → https://www.legitimate123.weebly.com/

\[ \varepsilon (\text{WsP}) \]
Why do we need all of this anyway? (first reason)

“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.”
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!

[5]: Eugene Bagdasaryan and Vitaly Shmatikov. 2021. Blind backdoors in deep learning models. In USENIX Sec. Symp.
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!)

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

Ours

| Evasion space | ML-PWD types (F) | ML Algorithms | Defense | Datasets (reprod.) | Stat. Val. |
|---------------|------------------|---------------|---------|-------------------|------------|
| Both          | $F^u, F^r, F^c$  | DL, SL        | ✓       | 2 (√)             | ✓          |
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 $fpr$)

Results comparable to the state-of-the-art 😊

- Let’s attack such ML-PWD
  - The $tpr$ will decrease!

| $\mathcal{A}$ | $F$ | $tpr$ | $fpr$ | $tpr$ | $fpr$ |
|-------------|-----|-----|-----|-----|-----|
| $CN$       | $F^u$ | 0.96±0.008 | 0.021±0.0077 | 0.55±0.030 | 0.037±0.0076 |
|            | $F^r$ | 0.88±0.018 | 0.155±0.0165 | 0.81±0.019 | 0.008±0.0020 |
|            | $F^c$ | 0.97±0.006 | 0.018±0.0088 | 0.93±0.013 | 0.005±0.0025 |
| $RF$       | $F^u$ | 0.98±0.0014 | 0.007±0.0055 | 0.45±0.022 | 0.003±0.0014 |
|            | $F^r$ | 0.93±0.013 | 0.025±0.0118 | 0.94±0.016 | 0.006±0.0025 |
|            | $F^c$ | 0.98±0.006 | 0.007±0.0046 | 0.97±0.007 | 0.001±0.0011 |
| $LR$       | $F^u$ | 0.95±0.009 | 0.037±0.0100 | 0.24±0.017 | 0.011±0.0026 |
|            | $F^r$ | 0.82±0.017 | 0.144±0.0171 | 0.74±0.025 | 0.018±0.0036 |
|            | $F^c$ | 0.96±0.007 | 0.025±0.0077 | 0.81±0.020 | 0.013±0.0037 |
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://tinyurl.com/spacephish-demo)
Demonstration: competition-grade ML-PWD

- [link](https://spacephish.github.io) (https://tinyurl.com/spacephish-demo)
- [link](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, 'r') as f:
    original_data = f.read()
original_response = re.findall(r'original_respons', original_data)

{
    "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": 0.800,
    "p_mod_05": 0.741,
    "p_mod_06": 0.806,
    "p_mod_07": 0.741
}
```

```python
In [8]: # TEST ADVERSARIAL
with open(output_file, 'w') as f:
    adversarial_data = f.write(adversarial_response) 
adversarial_response = re.findall(r'adversarial_respons', adversarial_data)

{
    "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://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>')
    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, 'r')
original_data = f.read()
original_response = response
print(original_response)

```

```
In [8]: # TEST ADVERSARIAL

with open(output_file, 'w')
adversarial_data = adversarial
adversarial_response = response
print(adversarial_response)
```

ACSAC’22 – Dec. 7th, 2022
Demonstration: competition-grade ML-PWD

- [https://spacephish.github.io](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>')</ind>
    return in_html[:ind]
```

### 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.

```json
{  
    "n_models": 8,
    "p_mod_00": 0.891,
    "p_mod_01": 0.891,
    "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
}
```
SpacePhish: The Evasion-space of Adversarial Attacks against Phishing Website Detectors using Machine Learning

Giovanni Apruzzese, Mauro Conti, Ying Yuan