中国人民大學
RENMIN UNIVERSITY OF CHINA

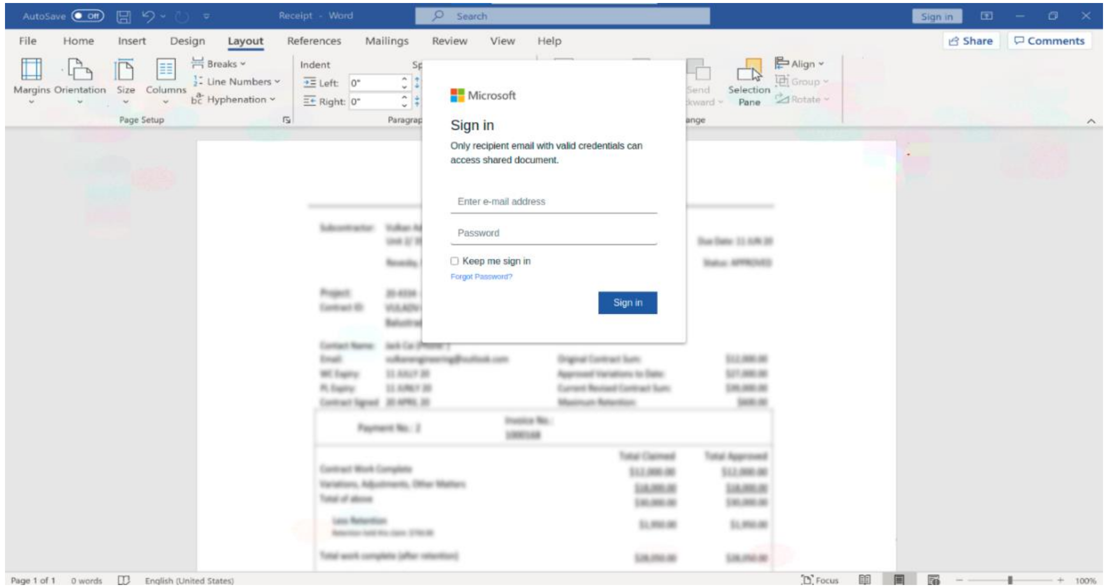
A Good Fishman Knows All the Angles:
A Critical Evaluation of Google’s Phishing Page Classifier

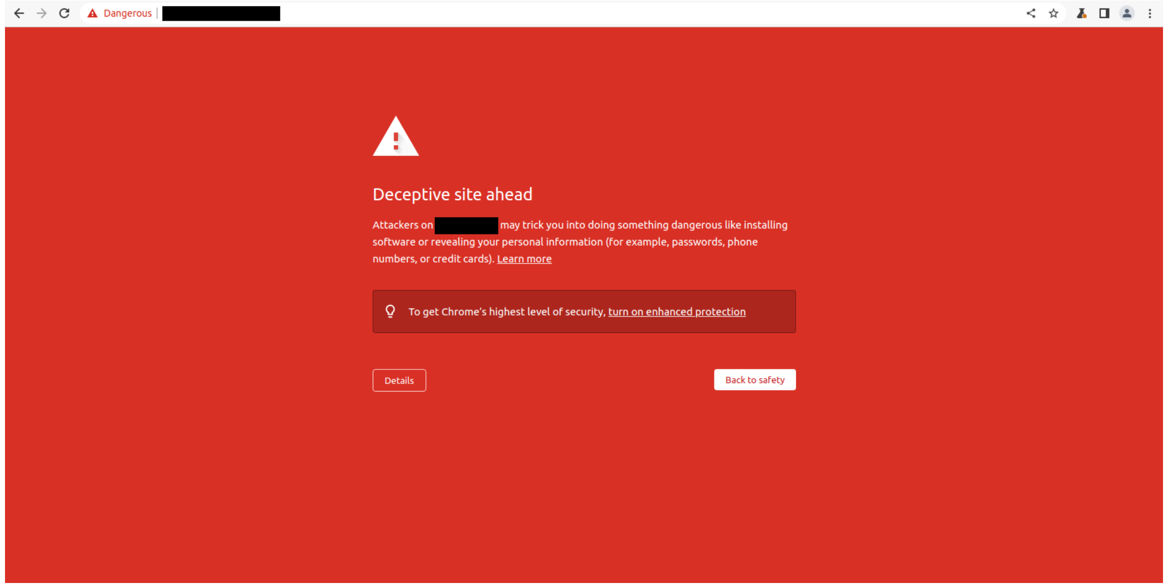
Changqing Miao, Jianan Feng, Wei You, Wenchang Shi, **Jianjun Huang**, Bin Liang

2023.11.29

Phishing Webpage

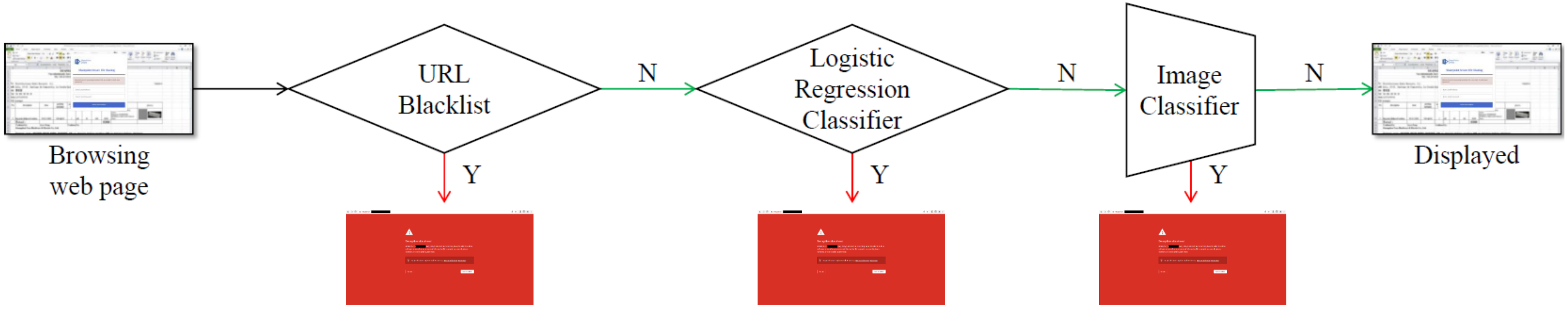
- **Left**: An example phishing page for Microsoft Login.
- **Right**: The phishing page is blocked by the browser.





Background

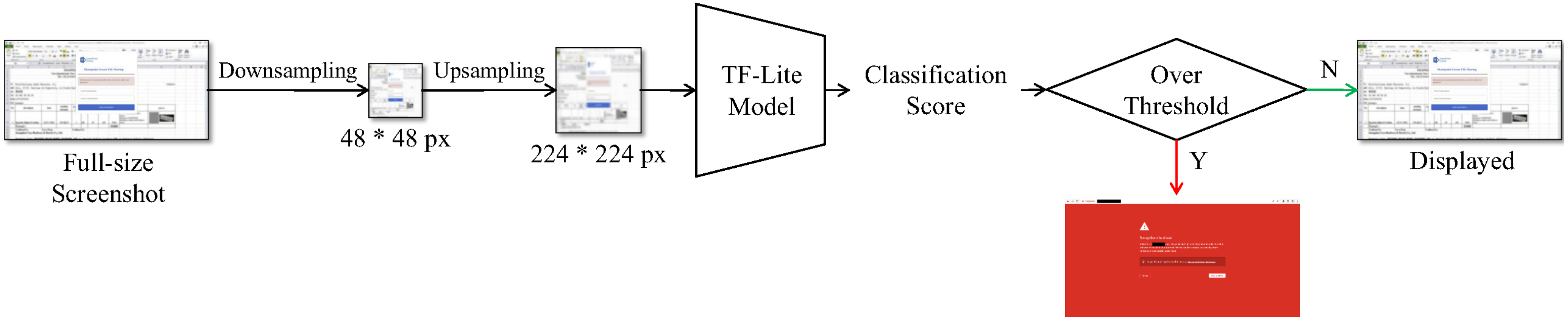
- Google's phishing page detection in Chrome/Chromium
  - URL blacklist & **Logistic regression classifier**: Evaded in 2016[1].
  - **CNN-based image classifier**: Evaded in this paper.



[1] Bin Liang, Miaoqiang Su, Wei You, Wenchang Shi, Gang Yang. *Cracking Classifiers for Evasion: A Case Study on the Google's Phishing Pages Filter*. In Proceedings of the 25th International World Wide Web Conference (**WWW 2016**).

Background: CNN-based Image Classifier

- The workflow of the image classifier-based phishing detection:
  - Capturing a screenshot of the webpage,
  - Downsampling it to a very small image (48*48 px),
  - Upsampling the small image to a larger one (224*224 px),
  - Feeding the image to the classifier,
  - Emitting a classification score with 18 dimensions for phishing categories plus one dimension for benign category,
  - If any of the phishing categories exceeds a threshold, the page would be potentially reported as a phishing page.



Research Problem & Challenges

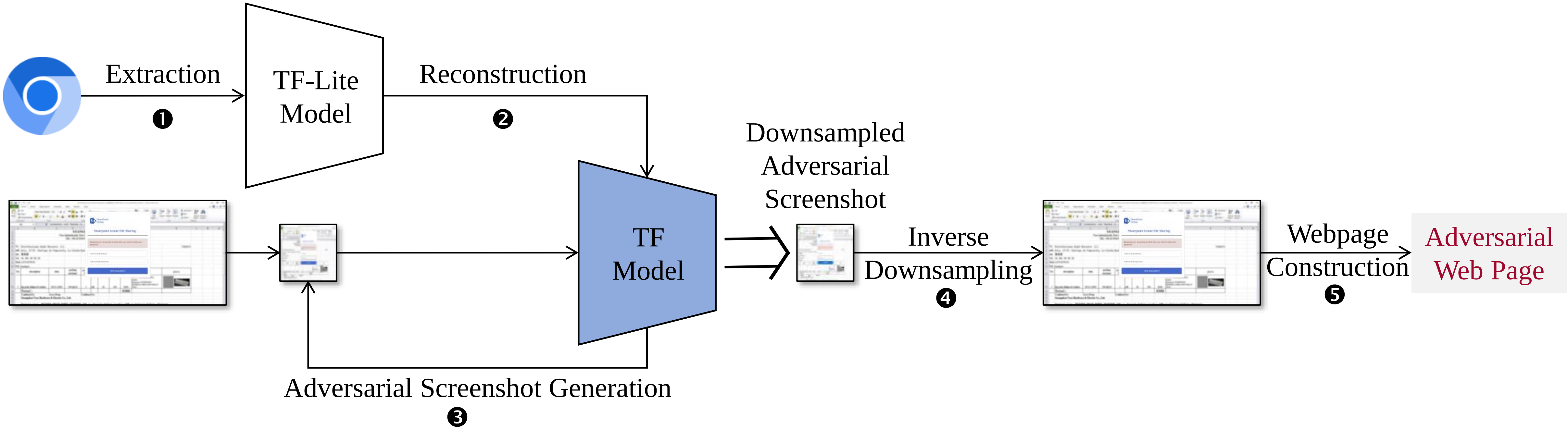
Research Problem

Can the new-gen phishing detection mechanism in Chrome/Chromium be bypassed, i.e., can we effectively evade the image classifier-based phishing detection and generate adversarial phishing page?

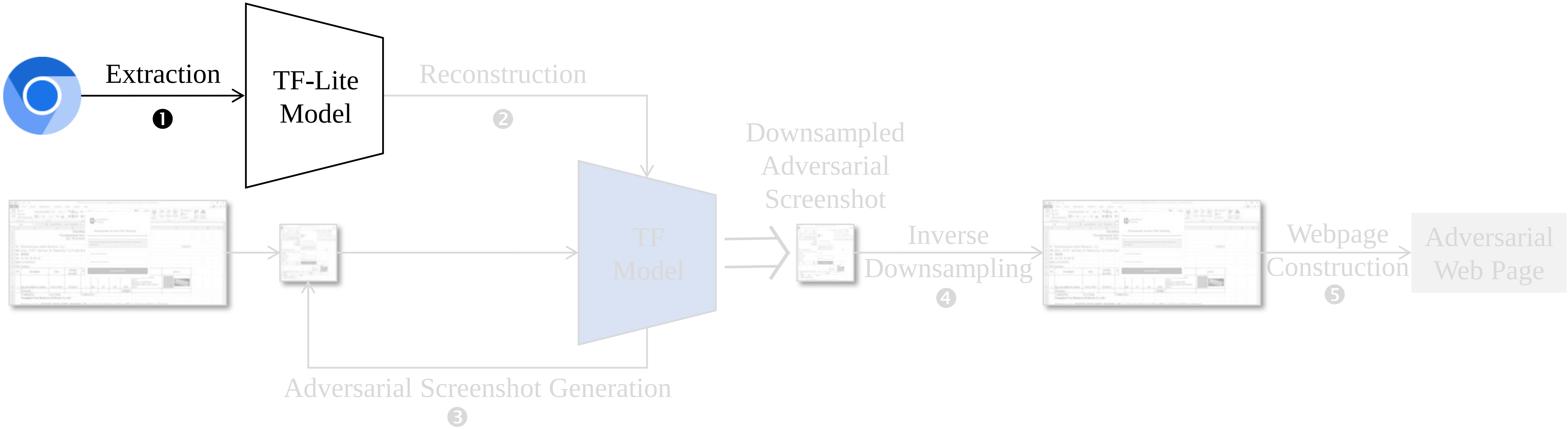
Challenges

- **Practical Evasion**: visual utility of the page should be preserved, the introduced perturbation should be imperceptible, and successful evasion should be reflected in real webpages.
- **Optimizable Model**: iterative optimization is not supported by the classification model in the browsers.
- **Feasible Computation**: search space is extremely large for generating a full-size adversarial image.

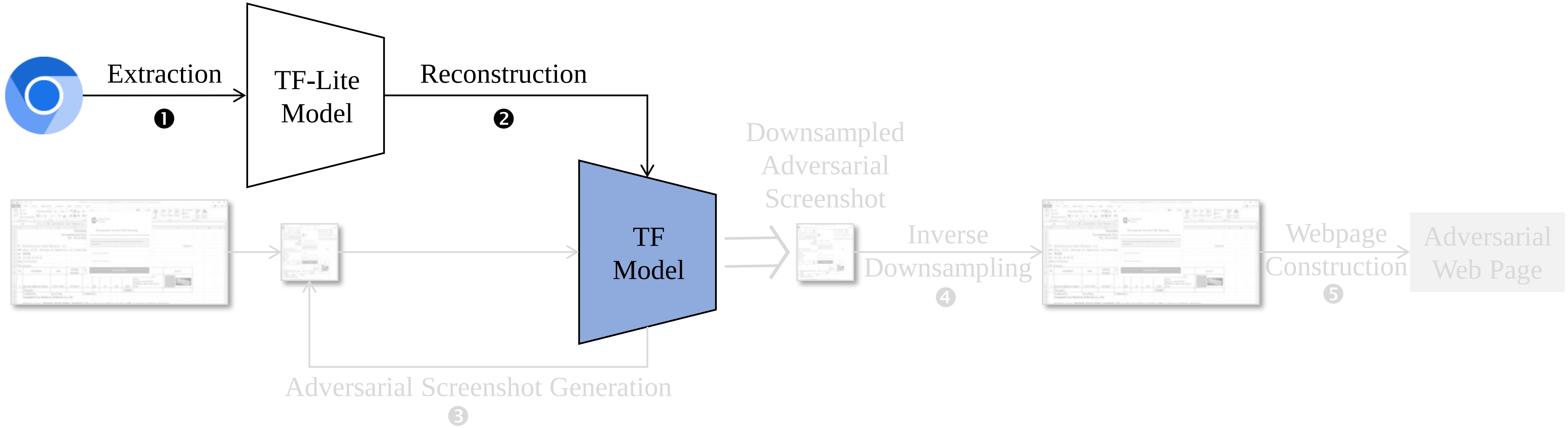
Approach


Extraction
❶
TF-Lite Model
Reconstruction
❷
TF Model
Downsampled Adversarial Screenshot
Inverse Downsampling
❹
Webpage Construction
❺
Adversarial Web Page
Adversarial Screenshot Generation
❸

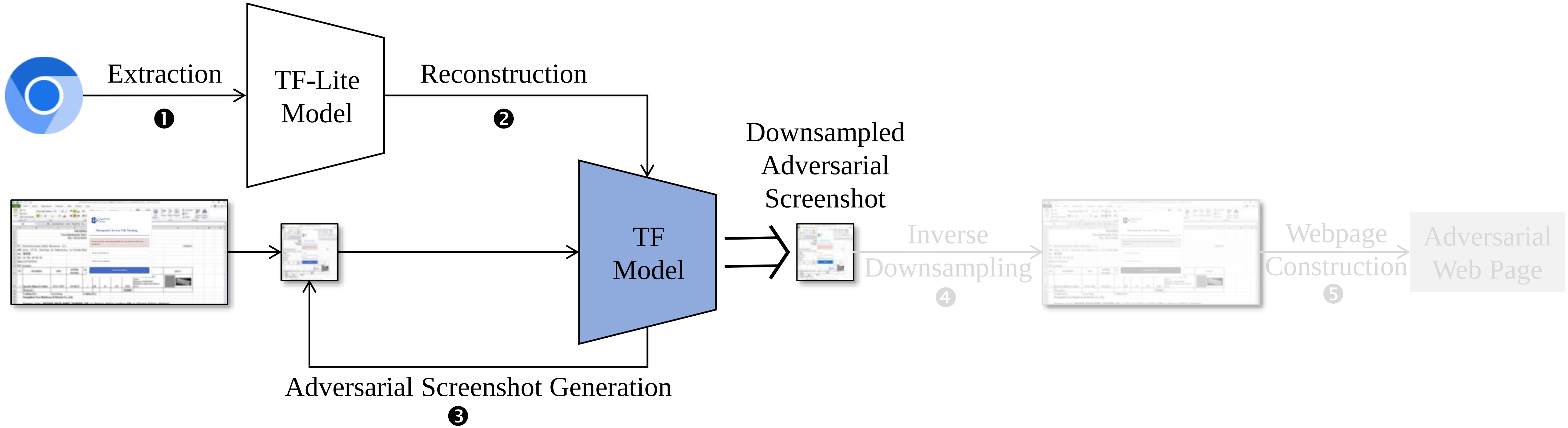
Approach


Extraction
❶
TF-Lite Model
Reconstruction
❷
Downsampled Adversarial Screenshot
TF Model
Inverse Downsampling
❹
Webpage Construction
❺
Adversarial Web Page
Adversarial Screenshot Generation
❸

Approach

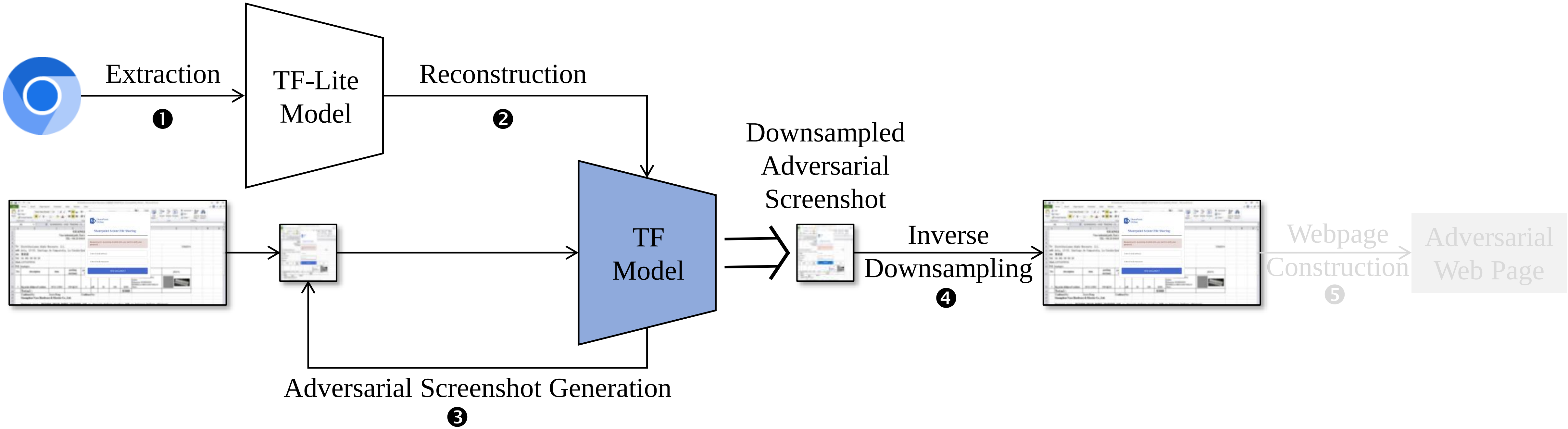

Extraction
❶
TF-Lite
Model
Reconstruction
❷
TF
Model
Downsampled
Adversarial
Screenshot
Inverse
Downsampling
④
Webpage
Construction
⑤
Adversarial
Web Page
Adversarial Screenshot Generation
③

Approach



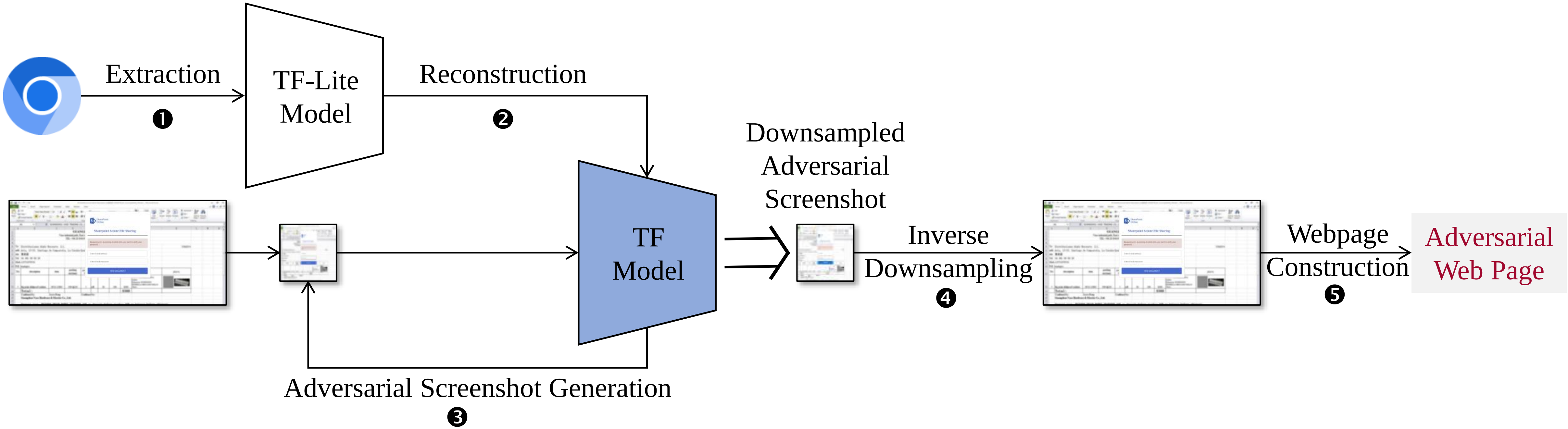
Extraction
❶
TF-Lite Model
Reconstruction
❷
TF Model
Downsampled Adversarial Screenshot
Adversarial Screenshot Generation
❸
Inverse Downsampling
④
Webpage Construction
⑤
Adversarial Web Page

Approach



Extraction
❶
TF-Lite
Model
Reconstruction
❷
TF
Model
Adversarial Screenshot Generation
❸
Downsampled
Adversarial
Screenshot
Inverse
Downsampling
❹
Webpage
Construction
❺
Adversarial
Web Page

Approach


Extraction
❶
TF-Lite
Model
Reconstruction
❷
TF
Model
Downsampled
Adversarial
Screenshot
Inverse
Downsampling
❹
Webpage
Construction
❺
Adversarial
Web Page
Adversarial Screenshot Generation
❸

Model Extraction

- **Locating** model use in source code
  - Searching the source code of Chromium and identifying candidate functions that contain related keywords, e.g, *phishing*, *classifier*, *classification*, etc;
  - Recompiling Chromium and running it in a debug mode, setting breakpoints on the candidate functions;
  - Visiting phishing webpages and collecting execution traces;
  - Analyzing the traces to identify the functions that are invoked in each triggering of the block pages.

- **Extracting** model data
  - Dumping the model data from the memory with the help of the debugger.

Model Extraction

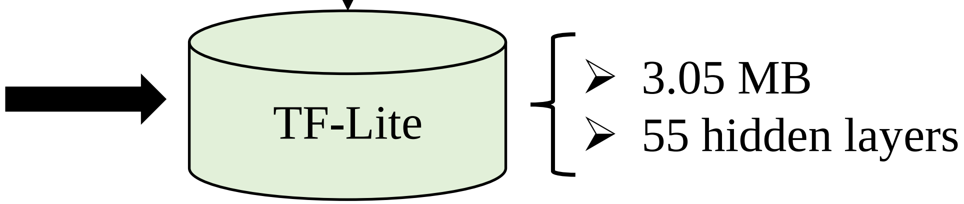
- **Locating** model use in source code
  - Searching the source code of Chromium and identifying candidate functions that contain related keywords, e.g, *phishing*, *classifier*, *classification*, etc;
  - Recompiling Chromium and running it in a debug mode, setting breakpoints on the candidate functions;
  - Visiting phishing webpages and collecting execution traces;
  - Analyzing the traces to identify the functions that are invoked in each triggering of the block pages.

- **Extracting** model data
  - Dumping the model data from the memory with the help of the debugger.

Model Extraction

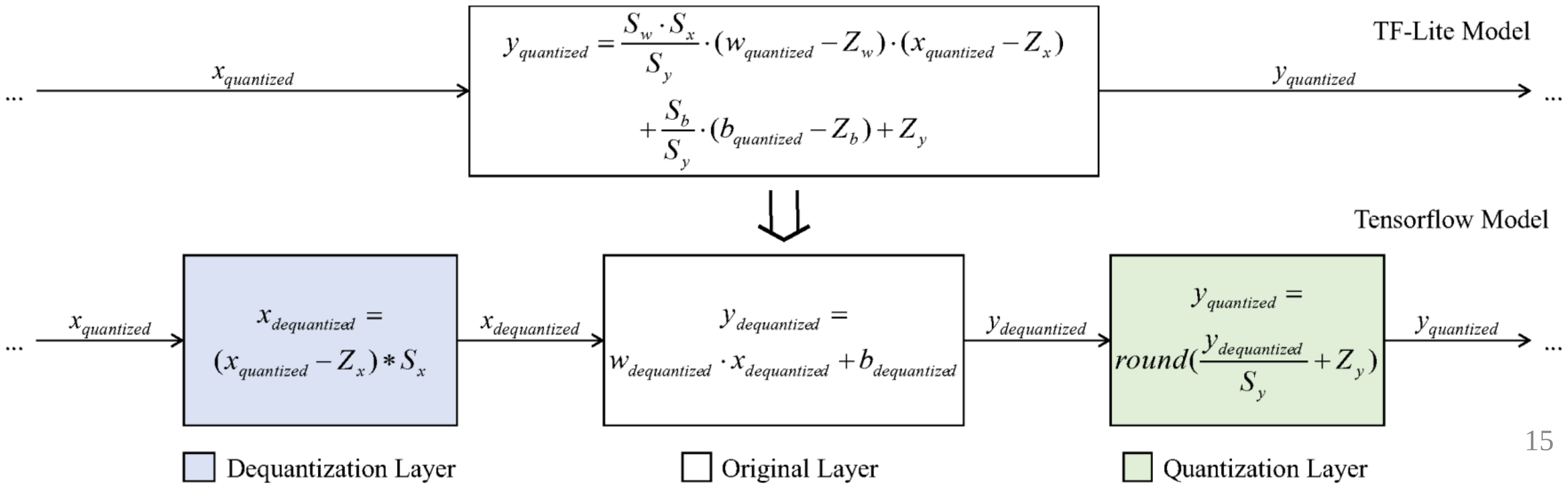
- **Locating** model use in source code
  - Searching the source code of Chromium and identifying candidate functions that contain related keywords, e.g, *phishing*, *classifier*, *classification*, etc;
  - Recompiling Chromium and running it in a debug mode, setting breakpoints on the candidate functions;
  - Visiting phishing webpages and collecting execution traces;
  - Analyzing the traces to identify the functions that are invoked in each triggering of the block pages.

- **Extracting** model data
  - Dumping the model data from the memory with the help of the debugger.



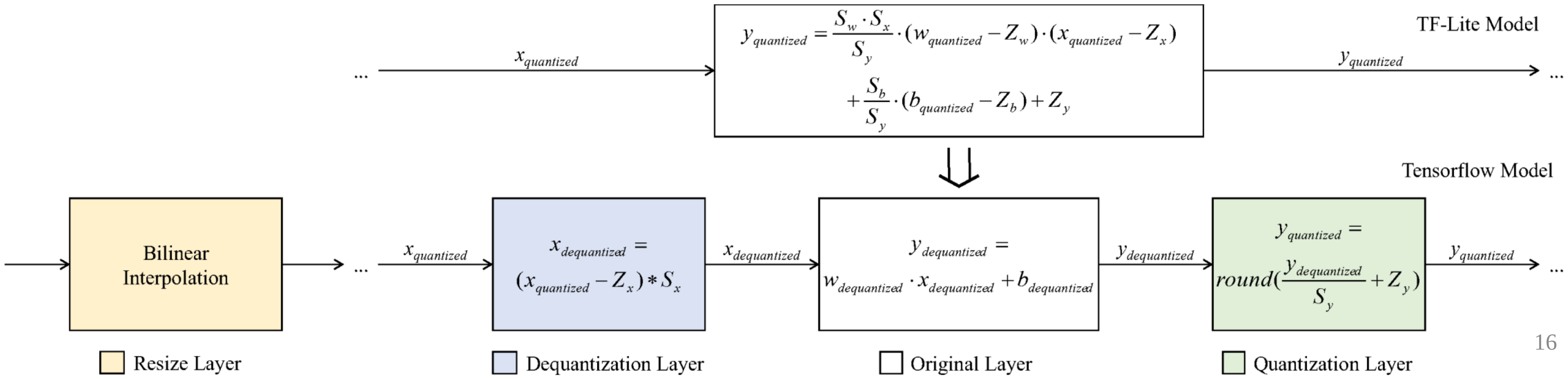
Model Reconstruction

- The **TF-Lite model**: lightweight, storage and computing resource efficient, discarding most functions irrelevant to classification, *impossible to implement gradient-based evasion attacks* as gradient calculations are not supported.
- **Reconstructing** TF-Lite model to a standard TensorFlow model
  - A **Dequantization** layer and a **Quantization** layer is added before and after each hidden layer in the TF model, to simulate the quantized computation in the TF-Lite model.
  - Gradient calculations are now supported in the TF model.
- A **Resize** layer is added to upsample the input (48*48px) to a 224*224px image.



Model Reconstruction

- The **TF-Lite model**: lightweight, storage and computing resource efficient, discarding most functions irrelevant to classification, *impossible to implement gradient-based evasion attacks* as gradient calculations are not supported.
- **Reconstructing** TF-Lite model to a standard TensorFlow model
  - A **Dequantization** layer and a **Quantization** layer is added before and after each hidden layer in the TF model, to simulate the quantized computation in the TF-Lite model.
  - Gradient calculations are now supported in the TF model.
- A **Resize** layer is added to upsample the input (48*48px) to a 224*224px image.



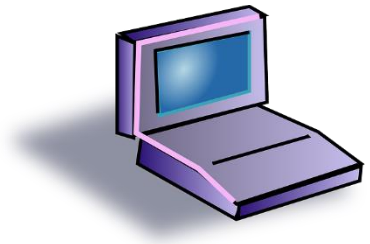
Adversarial Screenshot Generation

- **Input**: downsampled webpage screenshot (48*48 px)
- **Output**: downsampled adversarial webpage screenshot (48*48 px)
- **Method**:
  - Selecting candidate pixels for modification,
  - Modifying candidate pixels.

Adversarial Screenshot Generation

- **Input**: downsampled webpage screenshot (48*48 px)
- **Output**: downsampled adversarial webpage screenshot (48*48 px)
- **Method**:
  - Selecting **candidate pixels** for modification,
  - Modifying candidate pixels.

$score(x)=contribution(x) - saliency(x)$



Goal: Visual Utility

- **Low visual saliency**: browser users may not notice the changes.
- **High classification contribution**: modifying a few pixels can succeed the evasion.

Top k pixels as candidates

Adversarial Screenshot Generation

- **Input**: downsampled webpage screenshot (48*48 px)
- **Output**: downsampled adversarial webpage screenshot (48*48 px)
- **Method**:
  - Selecting candidate pixels for modification,
  - Modifying candidate pixels.

Calculating upper/lower limit → Calculating gradient & modification degree → Modifying and clipping pixels →

Adversarial sample

Iterative optimization *or* Stop

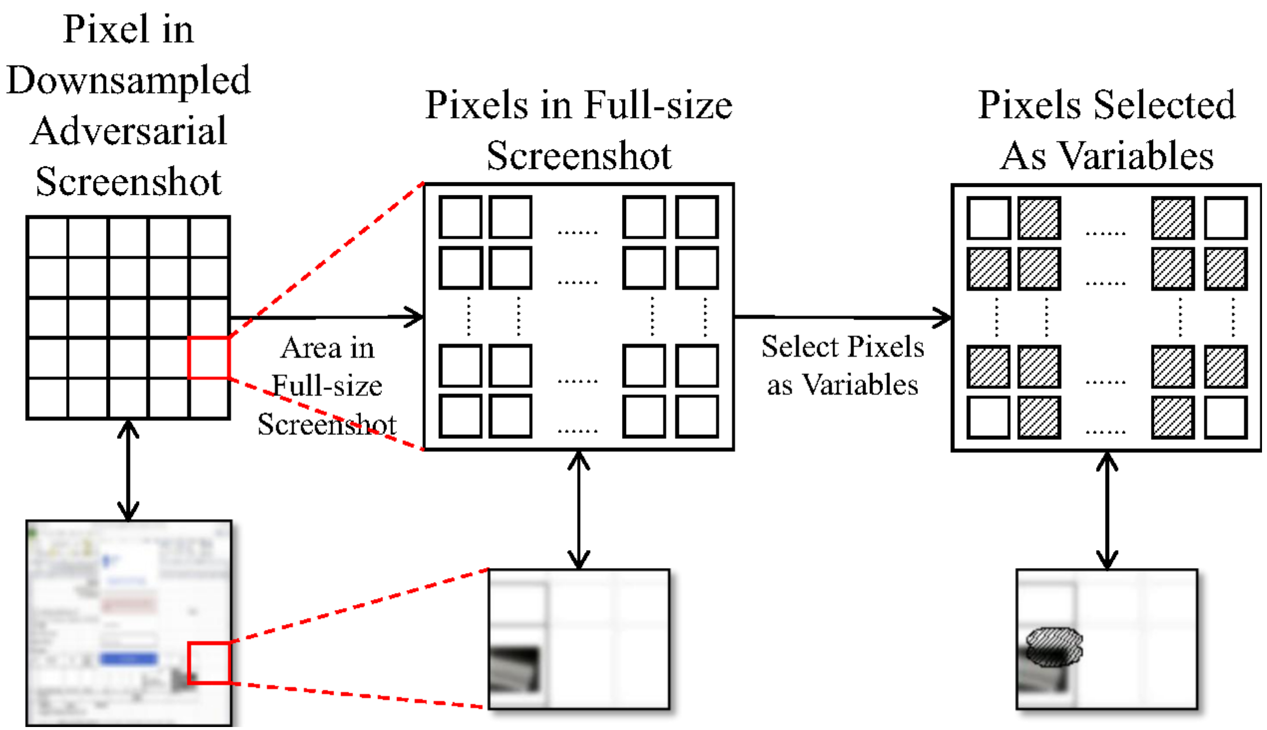
Inverse Downsampling

- Search space for generating a full-size adversarial screenshot from the downsampled adversarial webpage screenshot (48*48 px) is super large, and combinatorial explosion problem leads to unacceptable computation overhead.
- **Input**: downsampled adversarial webpage screenshot (48*48 px)
- **Output**: full-size adversarial webpage screenshot
- **Goal**: Efficiency & Visual utility
- **Method**:
  - Converting inverse downsampling to a integer linear programming problem,
  - Leveraging GUROBI optimizer[2] to solve the problem.

[2] GUROBI optimization, https://www.gurobi.com/

Inverse Downsampling – Variable Choice

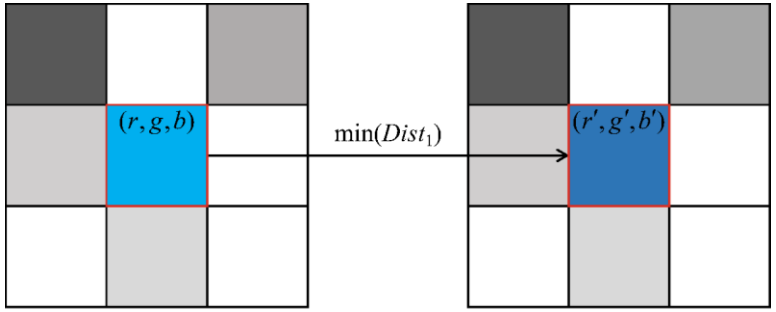
- **Variables**: pixels in the full-size adversarial screenshot, which are corresponding to the modified pixels in the downsampled adversarial screenshot.
- **Question**: which pixels to choose for modification, to *deceive browser users*?
- **Strategy**: changing the shape of the modified zone from a rectangle to the inscribed ellipse.



- **Rectangle**: sharp corners may attract the users about the abnormality.
- **Ellipse**: smooth edges make the modified zone look like a dripping stain on the monitor.

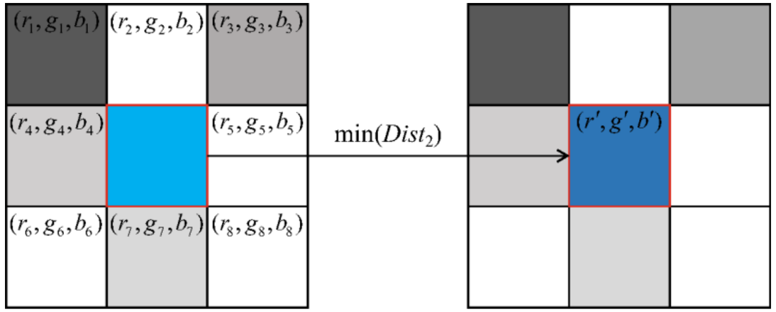
Inverse Downsampling – Objective Function

- **Goal**: optimizing the visual utility.
  - *As small perturbation as possible* (left) ---- The difference between a modified pixel in the adversarial screenshot and the counterpart in the original screenshot is imperceptible.
  - *As non-obtrusive perturbation as possible* (right) ---- The modified pixel looks close to surrounding pixels.



$Dist_1 = |r-r'| + |g-g'| + |b-b'|$

$|x_{adv} - x_{origin}|$



$Dist_2 = |r'-r_a| + |g'-g_a| + |b'-b_a| \quad r_a = \frac{1}{8}\sum_{i=1}^{8} r_i,\ g_a = \frac{1}{8}\sum_{i=1}^{8} g_i,\ b_a = \frac{1}{8}\sum_{i=1}^{8} b_i$

$|x_{adv} - x_{surrounding}|$

$$\min\ Z = C_{origin} \cdot |x_{adv} - x_{origin}| + C_{surrounding} \cdot |x_{adv} - x_{surrounding}|$$

Inverse Downsampling – Constraints

- **Validity**:
  - (1) pixel values should be in the valid range,

$$0 \leq x_{adv} \leq 255$$

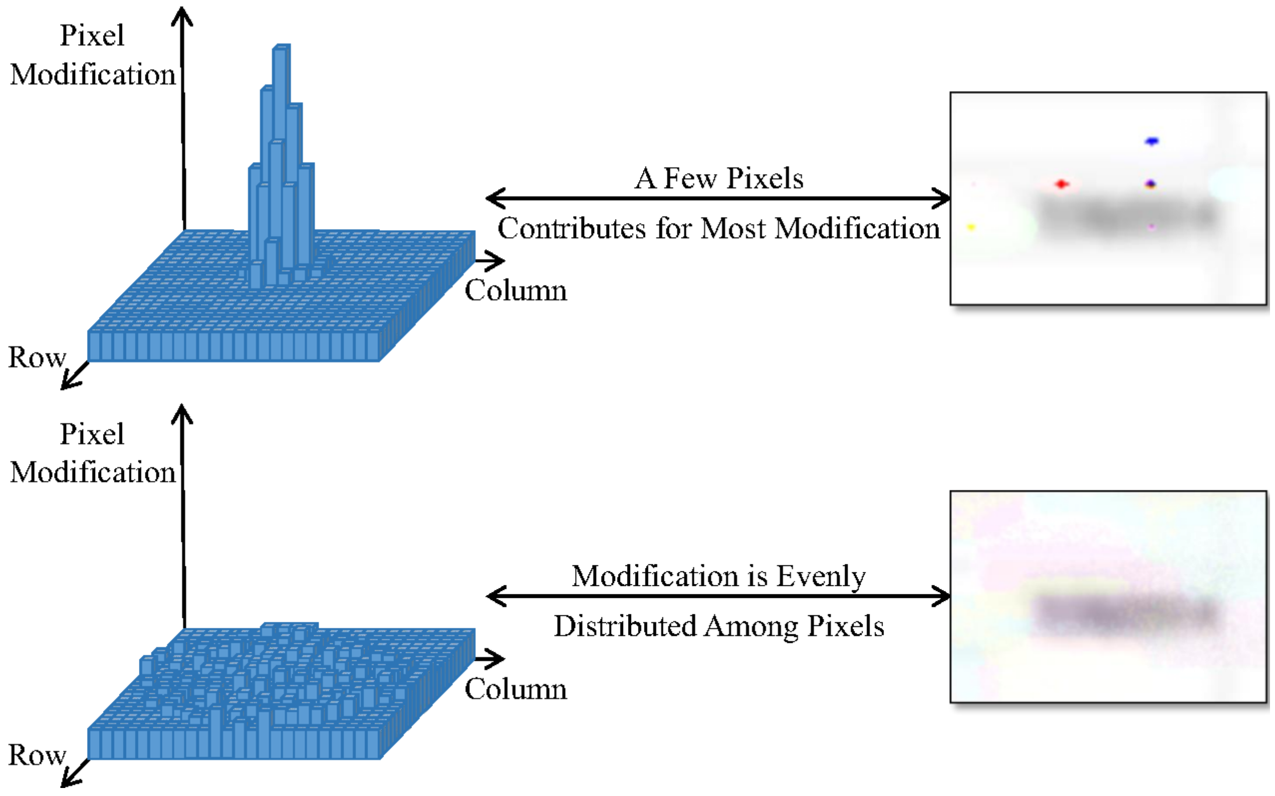
  -
  - (2) the constructed full-size screenshot should be downsampled to the input, i.e., the adversarial downsampled screenshot (48*48 px).

$$P_{downsized} = K \cdot x_{adv}$$

Inverse Downsampling – Constraints

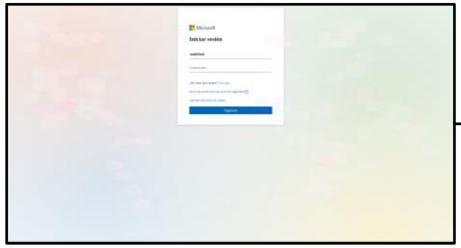
- **Visual Utility**: "*flattening*" the perturbations by forcing all pixels in the shape to be altered.

$$\delta \leq |x_{adv} - x_{origin}|$$



Webpage Construction

- Full-size SVG with modified pixels kept and the others transparent.
- Modified pixels overlapping with input fields are cut out as small PNG images.
- **Three layers** by the CSS *z-index* property:
  - *Bottom*: all HTML elements except the overlapped input fields in the original phishing webpage;
  - *Middle*: full-size SVG as the background of a *div*, with click operations going through the image and reaching the bottom layer;
  - *Top*: perturbated input fields with the corresponding PNG images as the background.

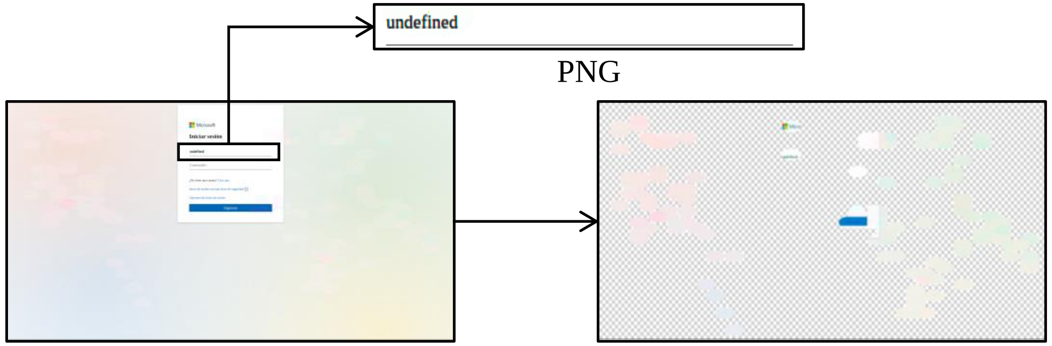


Full-size Adversarial Screenshot

SVG

Webpage Construction

- Full-size SVG with modified pixels kept and the others transparent.
- Modified pixels overlapping with input fields are cut out as small PNG images.
- **Three layers** by the CSS *z-index* property:
  - *Bottom*: all HTML elements except the overlapped input fields in the original phishing webpage;
  - *Middle*: full-size SVG as the background of a *div*, with click operations going through the image and reaching the bottom layer;
  - *Top*: perturbated input fields with the corresponding PNG images as the background.

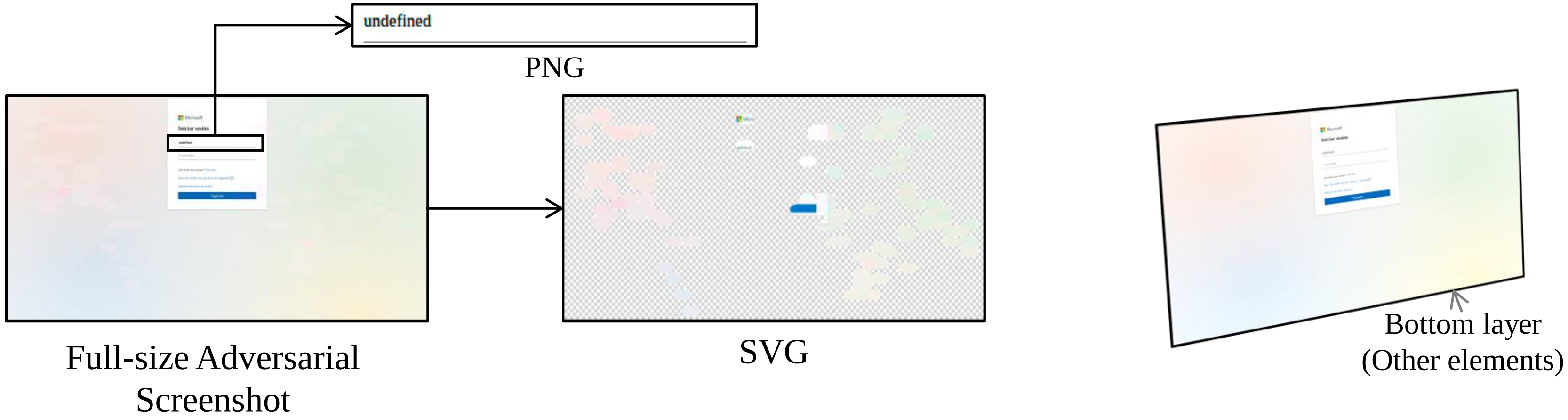


Full-size Adversarial Screenshot

SVG

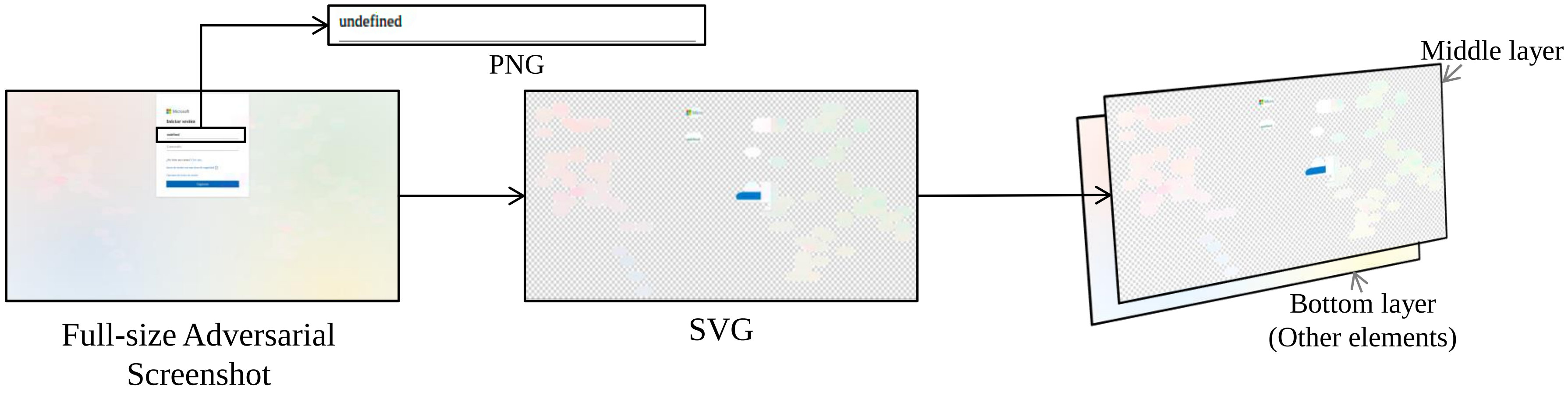
Webpage Construction

- Full-size SVG with modified pixels kept and the others transparent.
- Modified pixels overlapping with input fields are cut out as small PNG images.
- **Three layers** by the CSS *z-index* property:
  - *Bottom*: all HTML elements except the overlapped input fields in the original phishing webpage;
  - *Middle*: full-size SVG as the background of a *div*, with click operations going through the image and reaching the bottom layer;
  - *Top*: perturbated input fields with the corresponding PNG images as the background.



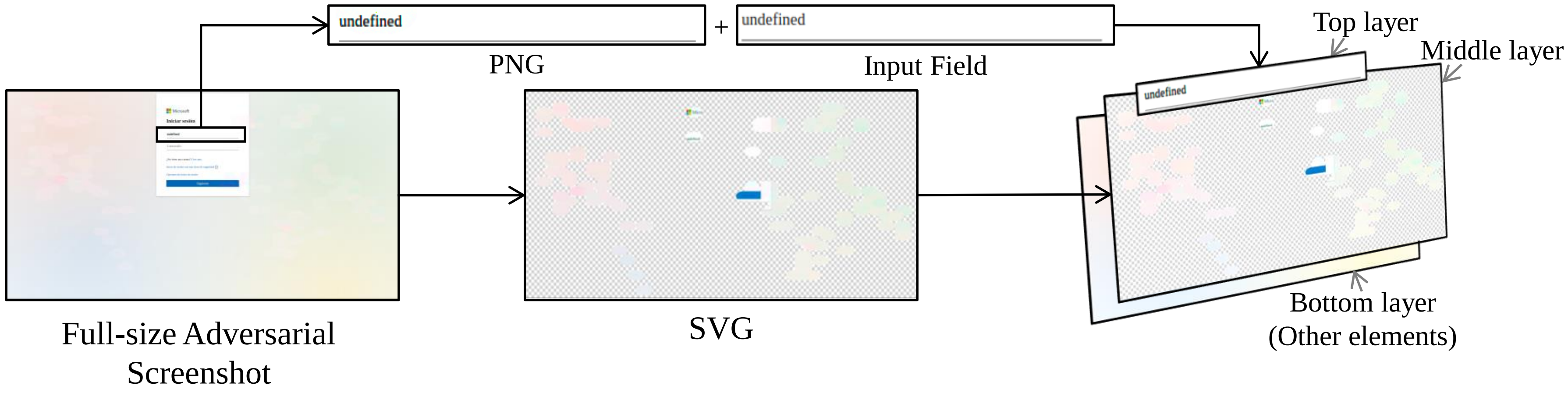
Webpage Construction

- Full-size SVG with modified pixels kept and the others transparent.
- Modified pixels overlapping with input fields are cut out as small PNG images.
- **Three layers** by the CSS *z-index* property:
  - *Bottom*: all HTML elements except the overlapped input fields in the original phishing webpage;
  - *Middle*: full-size SVG as the background of a *div*, with click operations going through the image and reaching the bottom layer;
  - *Top*: perturbated input fields with the corresponding PNG images as the background.



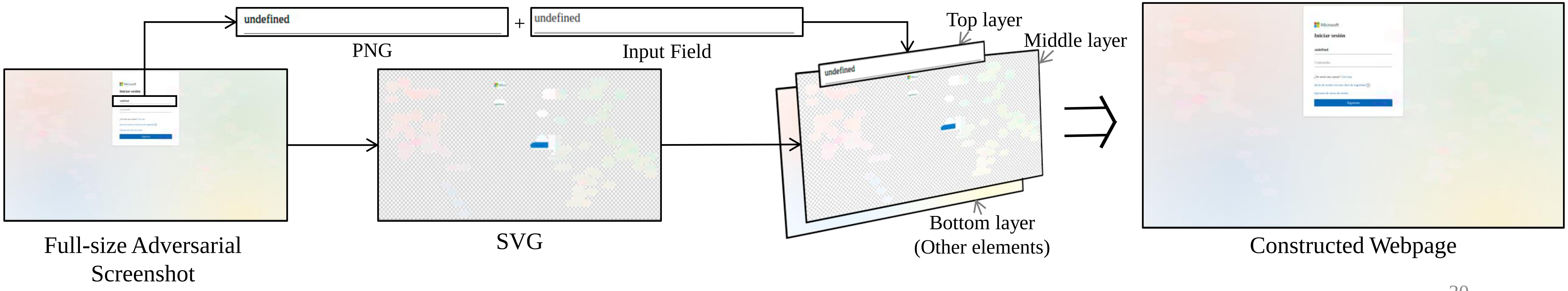
Webpage Construction

- Full-size SVG with modified pixels kept and the others transparent.
- Modified pixels overlapping with input fields are cut out as small PNG images.
- **Three layers** by the CSS *z-index* property:
  - *Bottom*: all HTML elements except the overlapped input fields in the original phishing webpage;
  - *Middle*: full-size SVG as the background of a *div*, with click operations going through the image and reaching the bottom layer;
  - *Top*: perturbated input fields with the corresponding PNG images as the background.



Webpage Construction

- Full-size SVG with modified pixels kept and the others transparent.
- Modified pixels overlapping with input fields are cut out as small PNG images.
- **Three layers** by the CSS *z-index* property:
  - *Bottom*: all HTML elements except the overlapped input fields in the original phishing webpage;
  - *Middle*: full-size SVG as the background of a *div*, with click operations going through the image and reaching the bottom layer;
  - *Top*: perturbated input fields with the corresponding PNG images as the background.



Evaluation

- **Datasets**: 135 real phishing webpages that can be identified by the CNN-based image classifier.
  - PhishBank: 50 phishing webpages
  - OpenPhish: 85 phishing webpages
- **Target**: Chromium 105

ID	Category	Bypass	AUC	Time	ID	Category	Bypass	AUC	Time	ID	Category	Bypass	AUC	Time
8120157	Outlook	✓	0.996	5.0	7945622	Shared Document	✓	0.995	16.5	op041	Shared Document	✓	0.963	6.1
8101033	Paypal	✓	0.996	2.4	7973561	Microsoft	✓	0.85	6.5	op042	Netflix	✓	0.994	11.4
8100905	Microsoft	✓	0.958	23.9	7969583	Shared Document	✓	0.994	4.3	op043	Microsoft	✓	0.959	3.7
8082292	Microsoft	✓	0.732	2.1	7958598	Netflix	✓	0.995	8.5	op044	Microsoft	✓	0.904	5.0
8082216	Facebook	✓	0.989	13.8	7897335	Shared Document	✓	0.988	4.1	op045	Microsoft	✓	0.998	1.8
8082078	Netflix	✓	0.998	19.8	op001	Microsoft	✓	0.946	3.0	op046	Microsoft	✓	0.998	1.9
8080988	Microsoft	✓	0.953	4.6	op002	Microsoft	✓	0.973	3.3	op047	Facebook	✓	0.997	0.4
8080214	Microsoft	✓	0.905	14.6	op003	Microsoft	✓	0.966	3.5	op048	Netflix	✓	0.993	53.8
8078574	Paypal	✓	0.996	2.9	op004	Microsoft	✓	0.977	14.5	op049	Outlook	✓	0.946	6.9
8052357	Outlook	✓	0.949	6.0	op005	Microsoft	✓	0.964	11.6	op050	Facebook	✓	0.998	7.5
8052355	Shared Document	✓	0.987	33.7	op006	Shared Document	✓	0.993	3.1	op051	Facebook	✓	0.998	7.4
8052333	Microsoft	✓	0.963	4.7	op007	Microsoft	✓	0.974	22.0	op052	Microsoft	✓	0.968	5.6
8042311	Microsoft	✓	0.964	3.9	op008	Microsoft	✓	0.9	87.8	op053	Outlook	✓	0.963	7.0
8042296	Netflix	✓	0.997	44.1	op009	Microsoft	✓	0.95	8.6	op054	Microsoft	✓	0.879	9.0
8042295	Netflix	✓	0.996	21.8	op010	Microsoft	✓	0.923	44.7	op055	Outlook	✓	0.963	6.0
8042290	Microsoft	✓	0.809	44.3	op011	Microsoft	✓	0.933	7.7	op056	Netflix	✓	0.998	11.3
8042289	Microsoft	✓	0.843	8.1	op012	Microsoft	✓	0.915	3.3	op057	Microsoft	✓	0.952	2.4
8040774	Paypal	✓	0.984	8.1	op013	Microsoft	✓	0.977	6.1	op058	Facebook	✓	0.997	5.5
8040771	Outlook	✓	0.952	6.2	op014	Facebook	✓	0.992	9.5	op059	Microsoft	✓	0.966	7.0
8040761	Shared Document	✓	0.987	32.0	op015	Microsoft	✓	0.958	1.4	op060	Microsoft	✓	0.944	9.5
8040752	Microsoft	✓	0.954	5.8	op016	Microsoft	✓	0.933	4.5	op061	Netflix	✓	0.998	12.4
8040698	Facebook	✓	0.987	12.0	op017	Microsoft	✓	0.939	6.8	op062	Microsoft	✓	0.950	11.2
8039424	Microsoft	✓	0.862	12.4	op018	Microsoft	✓	0.955	6.4	op063	Microsoft	✓	0.975	2.2
8039340	Microsoft	✓	0.956	7.0	op019	Microsoft	✓	0.956	5.5	op064	Shared Document	✓	0.998	3.7
7998977	Microsoft	✓	0.957	2.4	op020	Microsoft	✓	0.734	5.9	op065	Amazon	✓	0.993	14.7
7989793	Shared Document	✓	0.983	8.4	op021	Microsoft	✓	0.949	6.1	op066	Facebook	✓	0.997	6.2
7989787	Netflix	✓	0.997	46.8	op022	Microsoft	✓	0.997	6.0	op067	Microsoft	✓	0.965	17.5
7989781	Microsoft	✓	0.922	4.2	op023	Netflix	✓	0.992	20.4	op068	Netflix	✓	0.997	11.7
7989779	Microsoft	✓	0.939	5.7	op024	Microsoft	✓	0.954	5.6	op069	Microsoft	✓	0.965	7.7
7989777	Microsoft	✓	0.826	7.5	op025	Netflix	✓	0.992	50.6	op070	Amazon	✓	0.983	6.9
7984500	Microsoft	✓	0.955	19.8	op026	Outlook	✓	0.972	7.2	op071	Amazon	✓	0.993	8.5
7984484	Microsoft	✓	0.974	4.2	op027	Microsoft	✓	0.97	3.7	op072	Microsoft	✓	0.848	5.2
7983855	Outlook	✓	0.959	6.3	op028	Microsoft	✓	0.973	3.7	op073	Facebook	✓	0.998	8.2
7983589	Netflix	✓	0.997	7.9	op029	Microsoft	✓	0.965	3.2	op074	Microsoft	✓	0.960	6.4
7983585	Shared Document	✓	0.982	71.1	op030	Outlook	✓	0.965	6.7	op075	Microsoft	✓	0.972	0.9
7983549	Microsoft	✓	0.918	3.5	op031	Microsoft	✓	0.96	3.3	op076	Facebook	✓	0.987	7.5
7983503	Microsoft	✓	0.929	3.7	op032	Shared Document	✓	0.975	13.1	op077	Outlook	✓	0.983	6.8
7983245	Microsoft	✓	0.979	16.8	op033	Facebook	✓	0.995	6.1	op078	Facebook	✓	0.987	8.5
7982619	Shared Document	✓	0.992	3.3	op034	Microsoft	✓	0.926	4.1	op079	Facebook	✓	0.973	4.0
7981210	Microsoft	✓	0.899	3.3	op035	Facebook	✓	0.991	4.7	op080	Microsoft	✓	0.964	14.3
7949021	Microsoft	✓	0.903	7.5	op036	Facebook	✓	0.998	6.2	op081	Microsoft	✓	0.971	3.4
7949019	Microsoft	✓	0.968	7.0	op037	Microsoft	✓	0.947	5.3	op082	Outlook	✓	0.970	9.8
7949016	Microsoft	✓	0.997	6.4	op038	Netflix	✓	0.998	12.4	op083	Microsoft	✓	0.985	4.9
7948992	Shared Document	✓	0.984	8.2	op039	Microsoft	✓	0.907	23.3	op084	Outlook	✓	0.964	5.1
7948941	Microsoft	✓	0.850	6.6	op040	Microsoft	✓	0.971	3.6	op085	Outlook	✓	0.963	5.2
Average													**0.959**	**10.7**

Effectiveness & Efficiency

- **Effectiveness**: all the 135 adversarial phishing webpages evade the CNN-based image classifier.
- **Efficiency**: About ten minutes are required to generate the adversarial downsampled screenshot and the full-size adversarial screenshot.
  - Python 3.7.11, TensorFlow 2.7.0
  - Intel Core i7-10870H CPU @ 2.20 GHz
  - >92.6% samples take no more than half an hour

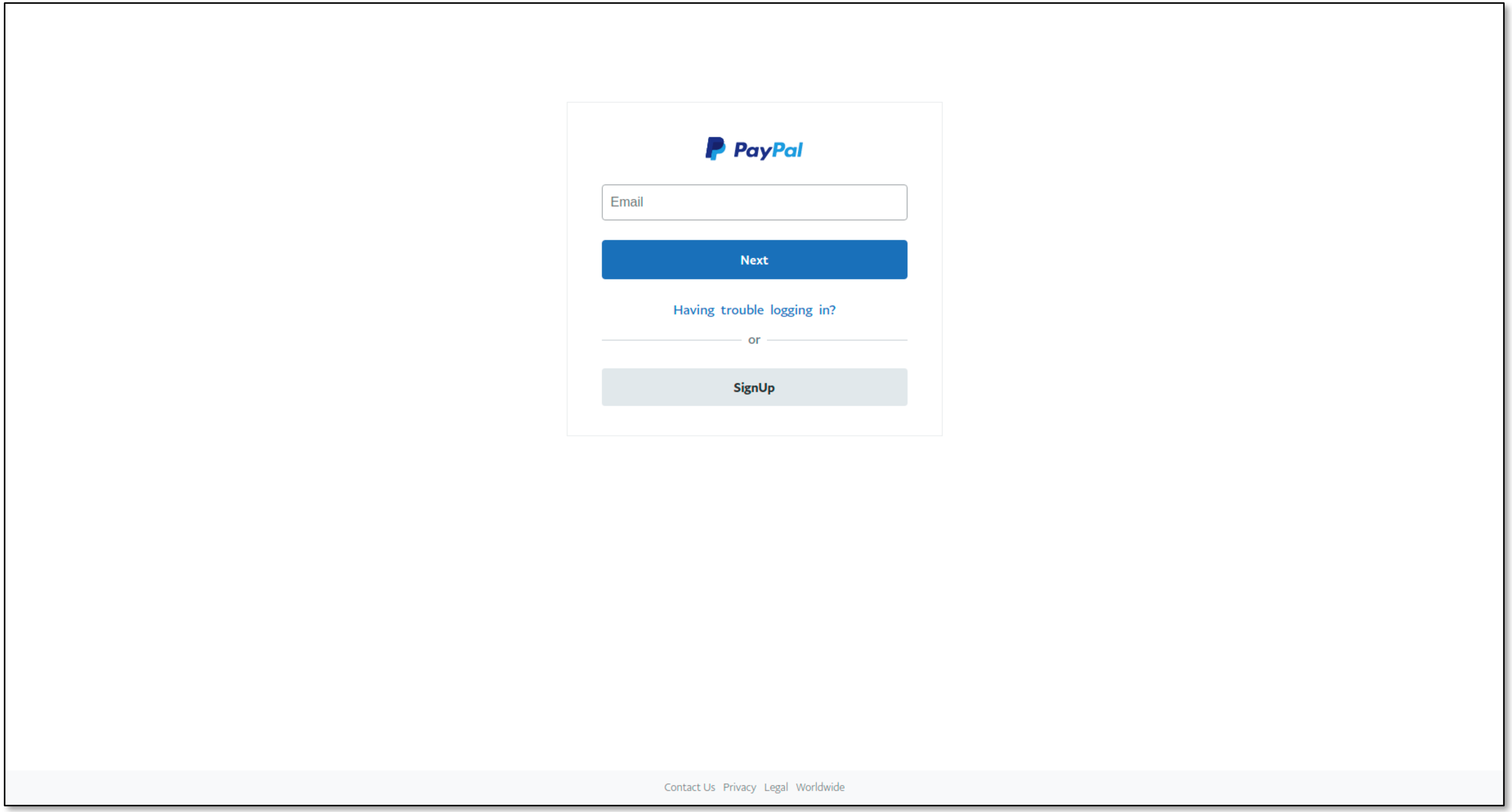
#Samples	#Successfully Evaded	Success Rate	Average Time Cost (Adv. Screenshot Generation + Inverse Downsampling)
135	135	**100%**	10.7 minutes

Visual Utility

- **Quantitative analysis**: average **AUC** is **0.959**, indicating *low page-wide saliency change* between full-size adversarial screenshot and the original screenshot.
- **User study**: 23 CS students to identify abnormal or incongruent regions on the displayed pages.
  - Randomly displaying 100 phishing webpages, 50 adversarial + 50 original
  - 1700 valid feedback entries
  - *Around half of the samples are inaccurately recognized;*
  - *None of the samples achieve unanimous correct identification across all participants;*
  - *Differentiating between the adversarial and original samples is challenging for the participants.*
- **Conclusion**: the visual utility of adversarial samples is well preserved.

	Original	Adversarial
Total (1700)	858	842
Correctly Identified (860)	574 (66.9%)	286 (34.0%)
Incorrectly Identified (840)	284 (33.1%)	556 (66.0%)

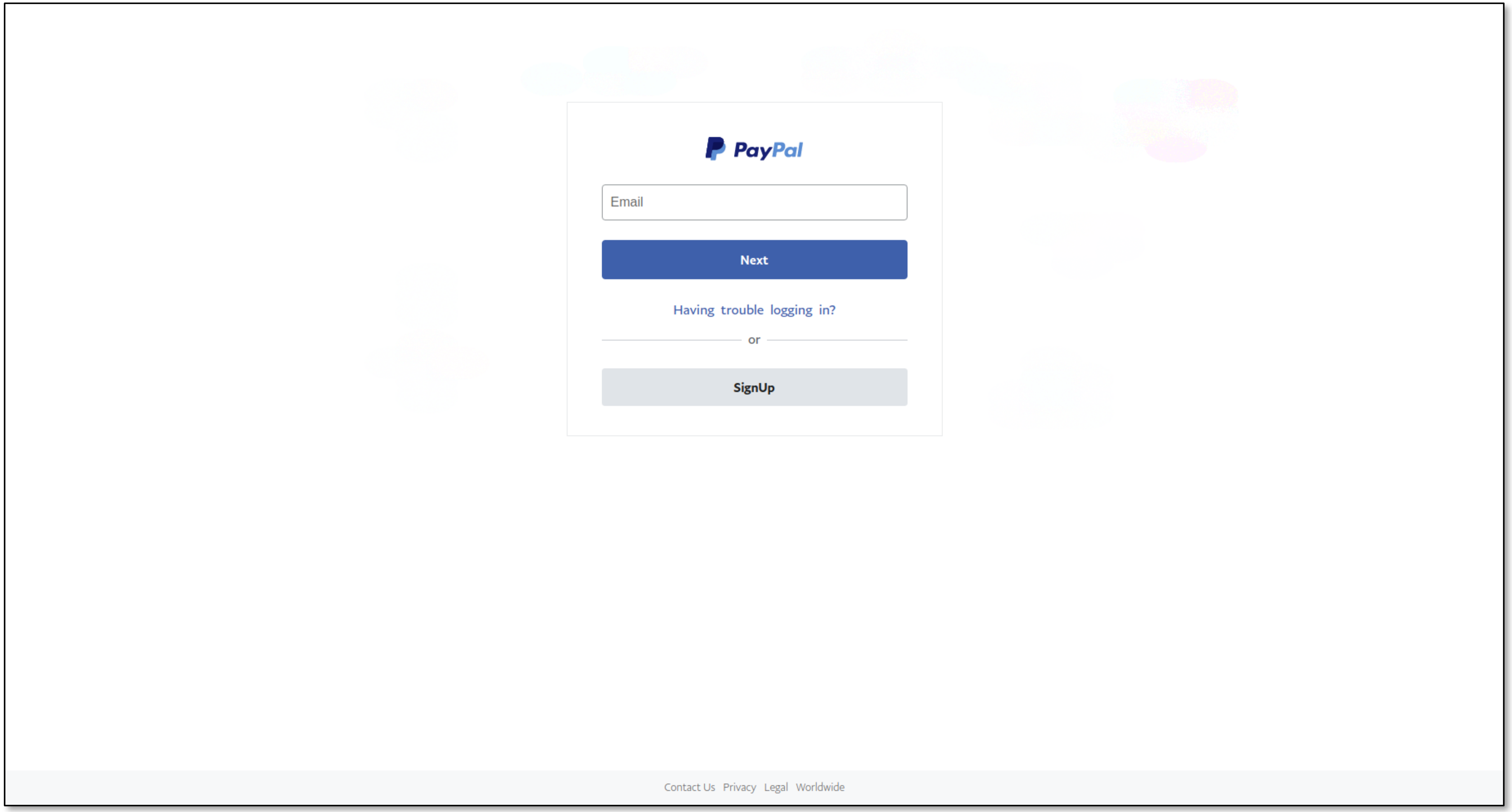
Case Study - Original



Classification score:

0.91 (phishing)

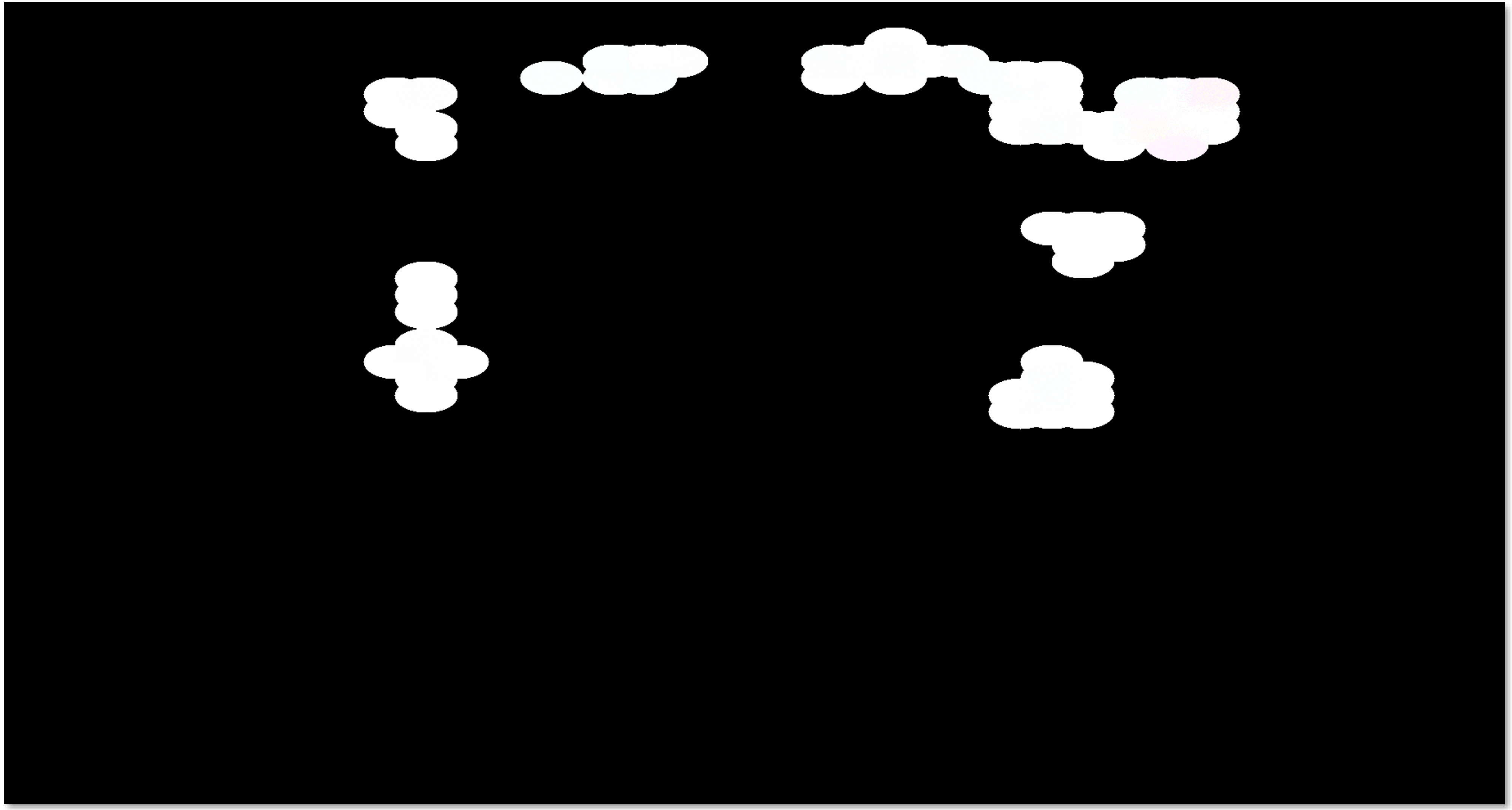
Case Study - Adversarial



Classification score:

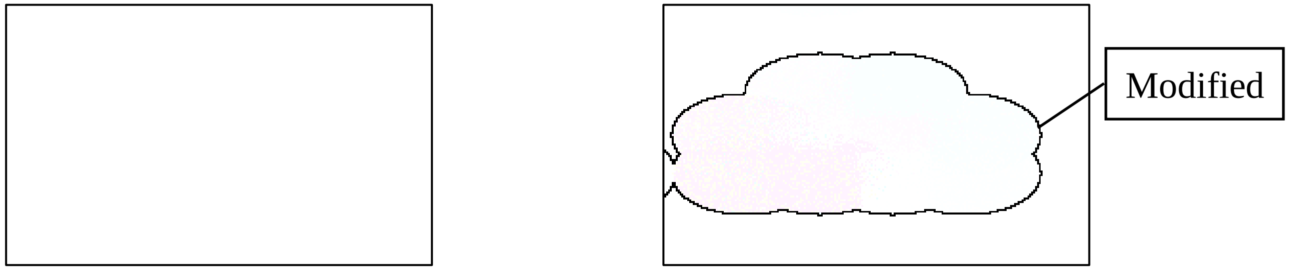
0.20 (benign)

Case Study – Perturbed Regions



Case Study – Perturbation Detail

- Smooth and flattened perturbations lead to a little saliency deviation and will not cause marked shift in attentional focus of browser users.



Original Part

Modified Part

Conclusion

- Proposing a feasible approach to bypassing Google's new-generation CNN-based phishing webpage detector;
- Bypassing the phishing image classifier with a success rate of 100%, with visual utility well preserved in the adversarial webpages;
- Generating the adversarial webpage is fast.

- Artifacts: *https://github.com/GoodPhishman/A-Good-Fishman-Knows-All-the-Angles*

Q & A

Adversarial Screenshot Generation

- **Input**: downsampled webpage screenshot (48*48 px)
- **Output**: downsampled adversarial webpage screenshot (48*48 px)
- **Method**:
  - Selecting candidate pixels for modification,
  - Modifying candidate pixels.

- Attacker chosen high-risk pixels, e.g., input box
- Medium-risk pixels, e.g., logo
- Low-risk pixels, e.g., bg images

Calculating upper/lower limit → Calculating gradient & modification degree → Modifying and clipping pixels →

Adversarial sample

Iterative optimization *or* **Stop**

Adversarial Screenshot Generation

- **Input**: downsampled webpage screenshot (48*48 px)
- **Output**: downsampled adversarial webpage screenshot (48*48 px)
- **Method**:
  - Selecting candidate pixels for modification,
  - Modifying candidate pixels.

➢ Ensuring modifications within upper/lower limit

Calculating upper/lower limit → Calculating gradient & modification degree → **Modifying and clipping pixels** →

Adversarial sample

Iterative optimization *or* **Stop**

Adversarial Screenshot Generation

- **Input**: downsampled webpage screenshot (48*48 px)
- **Output**: downsampled adversarial webpage screenshot (48*48 px)
- **Method**:
  - Selecting candidate pixels for modification,
  - Modifying candidate pixels.

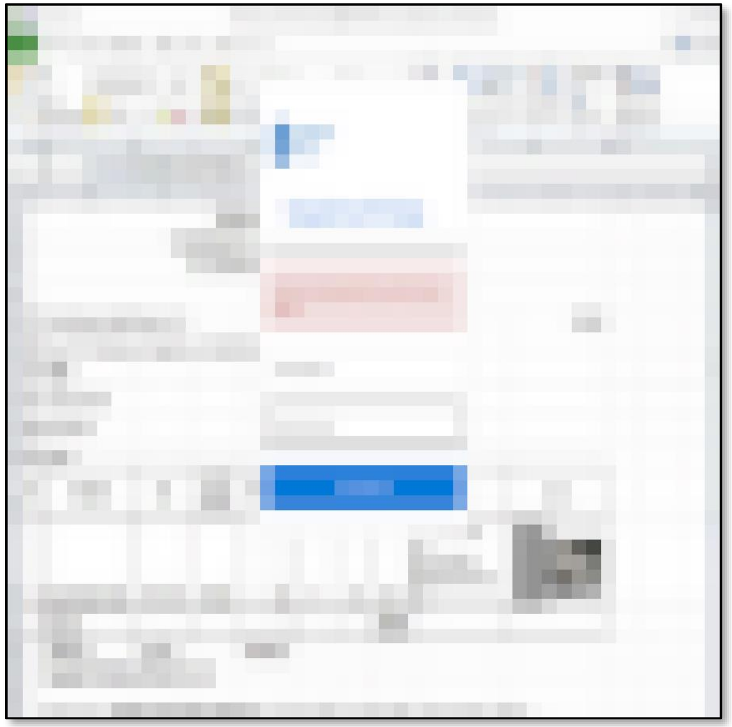
➢ Relaxing upper/lower limits, or
➢ Modifying more pixels

Calculating upper/lower limit → Calculating gradient & modification degree → Modifying and clipping pixels →

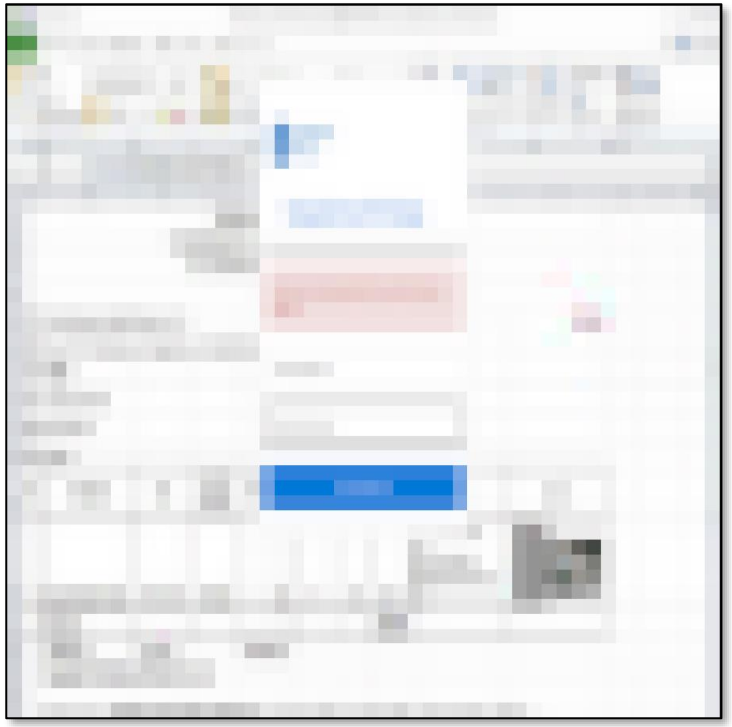
Adversarial sample

Iterative optimization *or* **Stop**

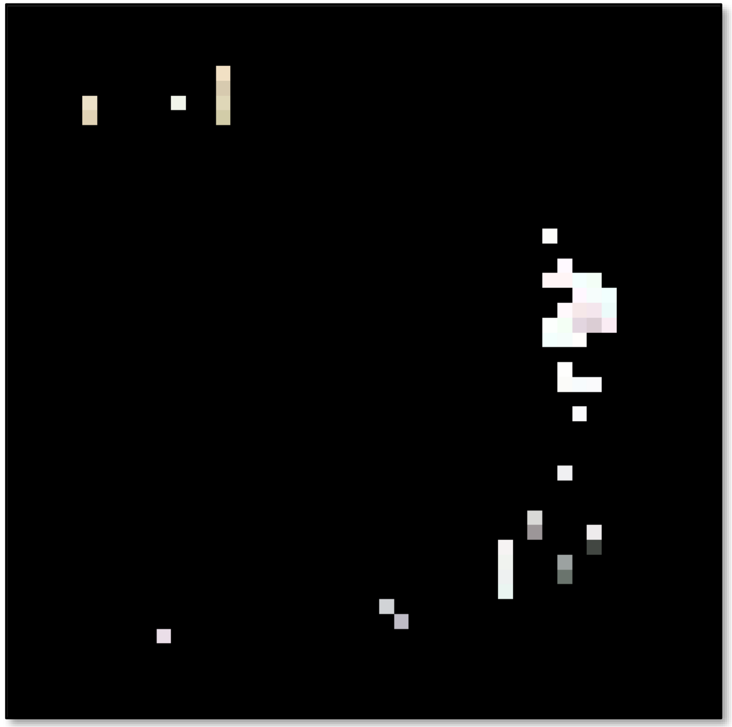
An Example – Adversarial Screenshot Generation



Original downsampled screenshot (48*48 px)

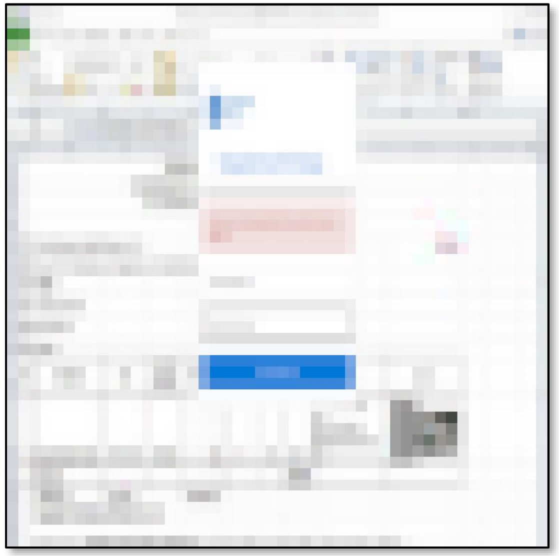


Adversarial downsampled screenshot (48*48 px)

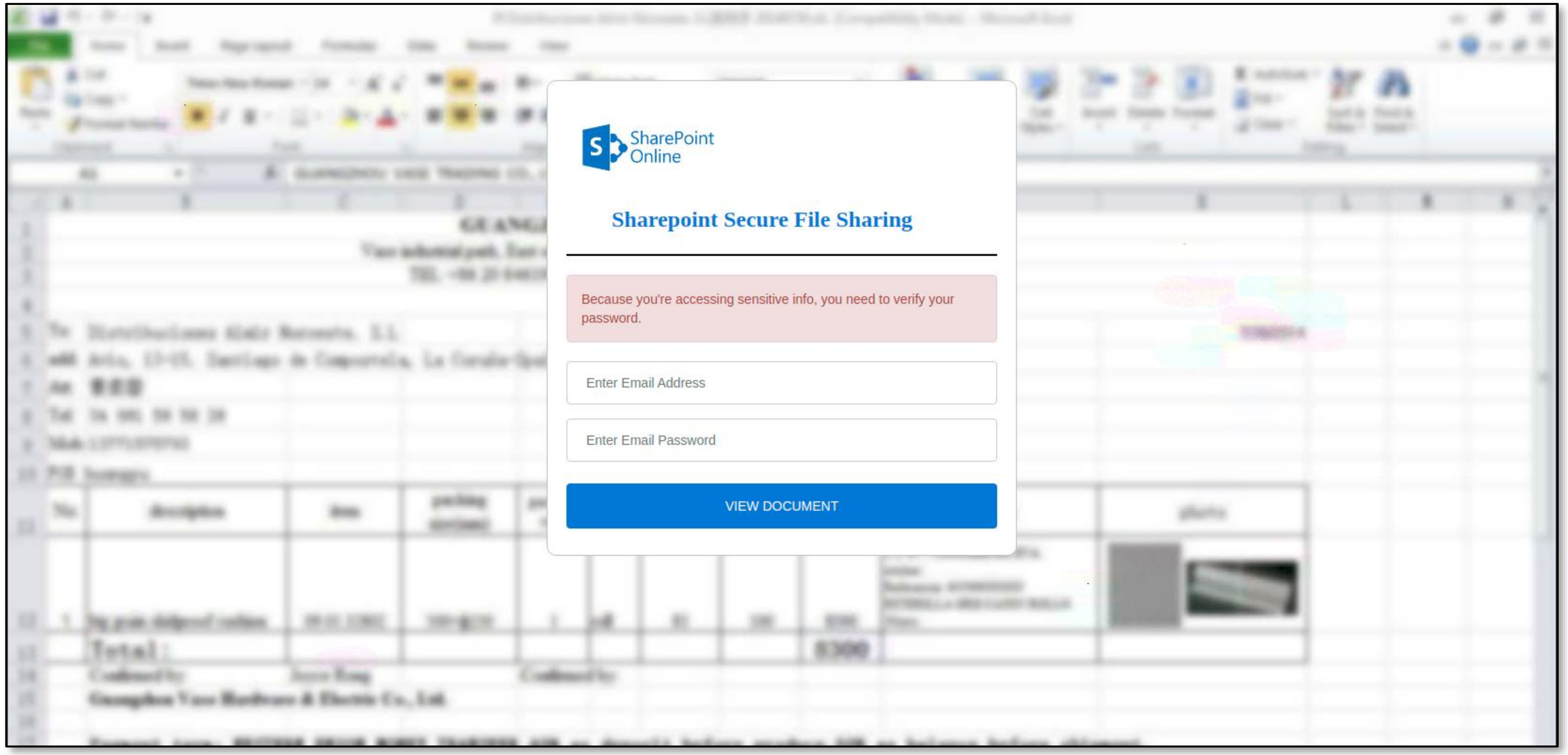


Perturbated pixels

An Example – Inverse Downsampling

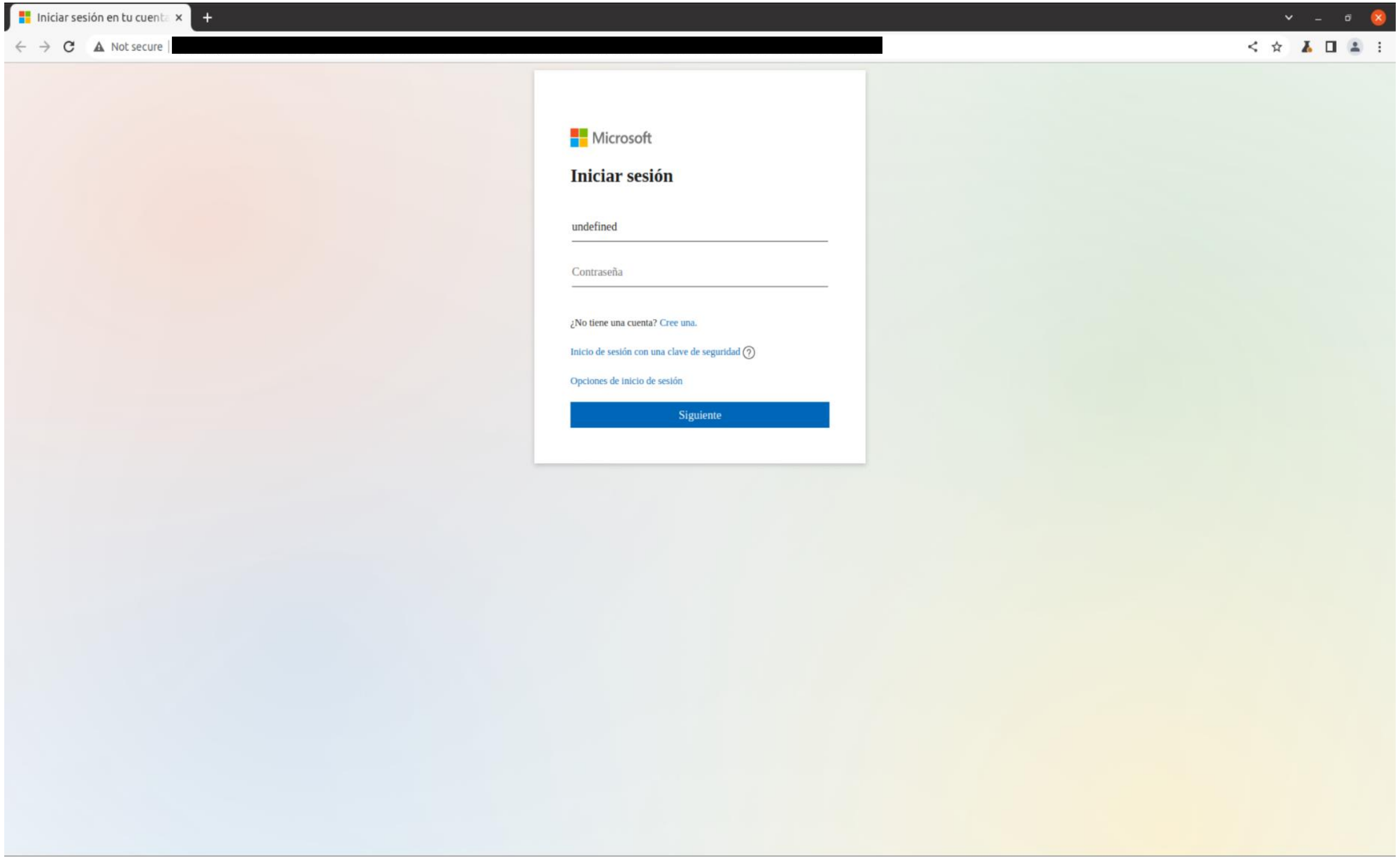


Adversarial downsampled screenshot (48*48 px)

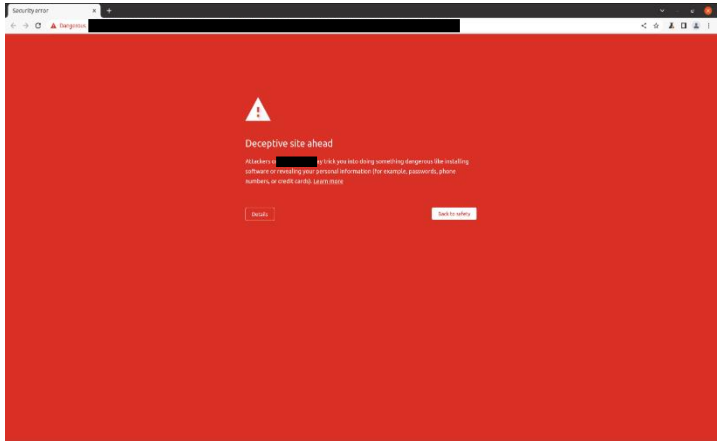


Full-size adversarial screenshot

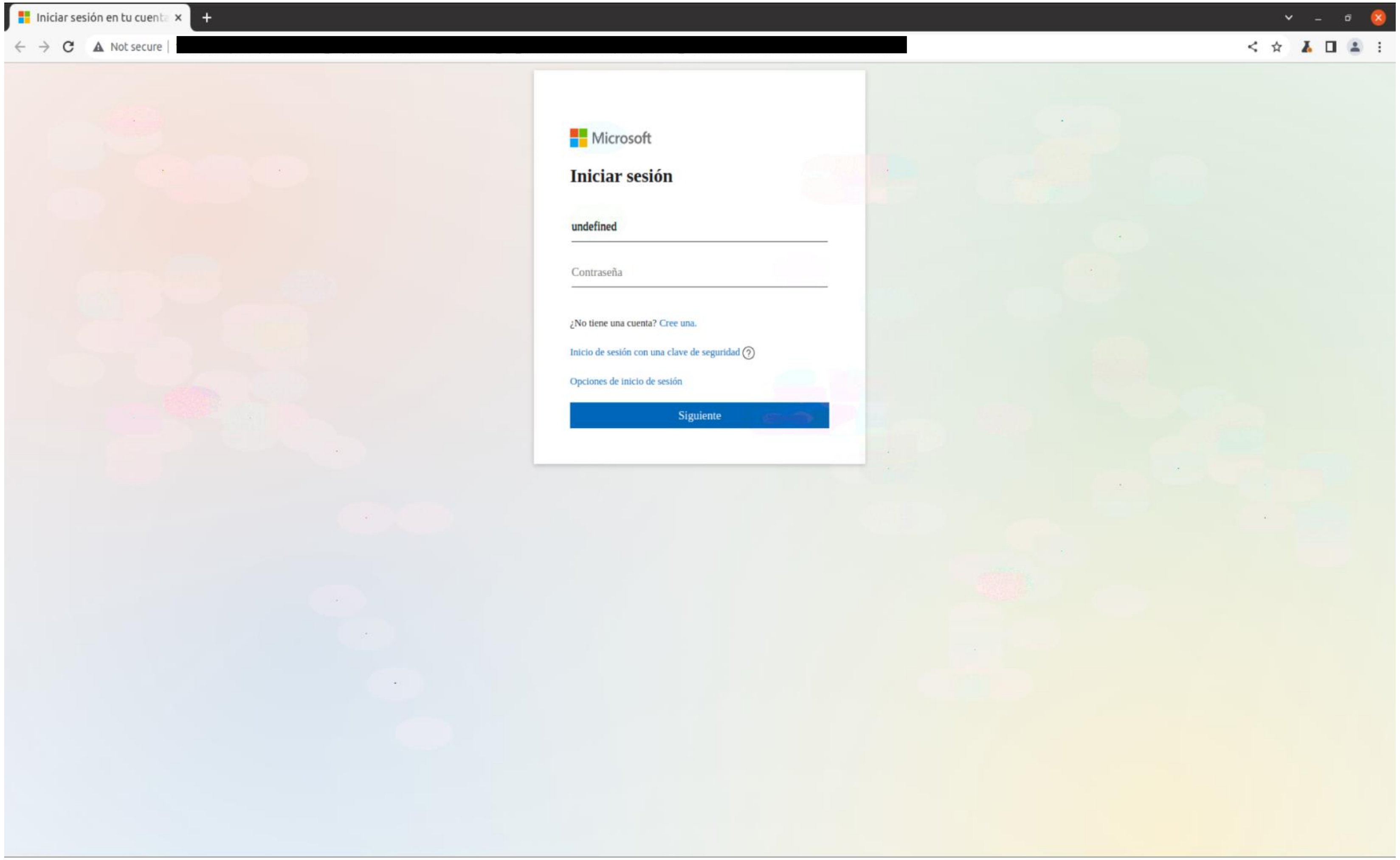
More Cases - 1



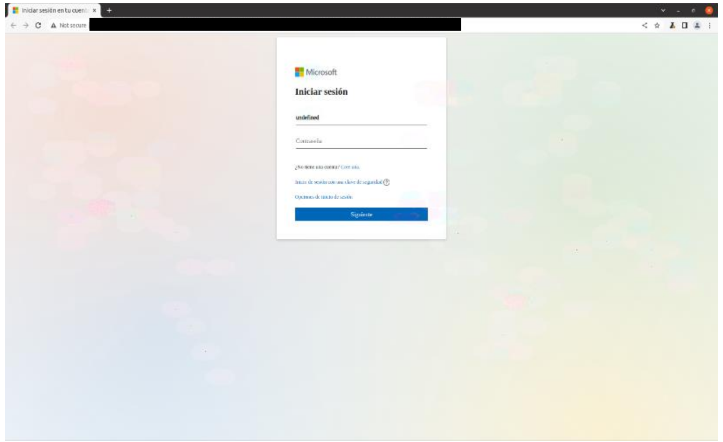
Original phishing page is blocked.



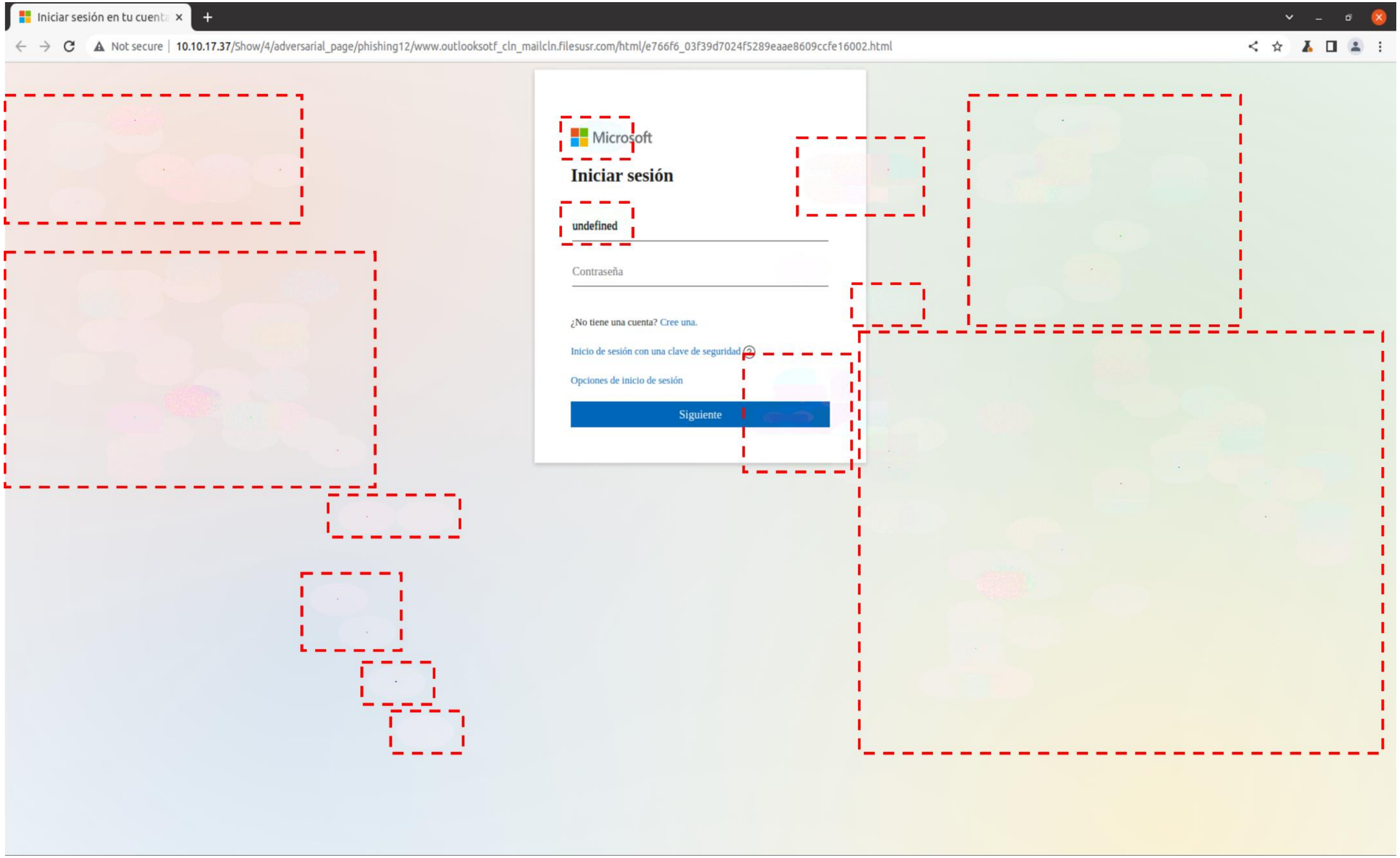
More Cases - 1



Adversarial phishing page is NOT blocked.

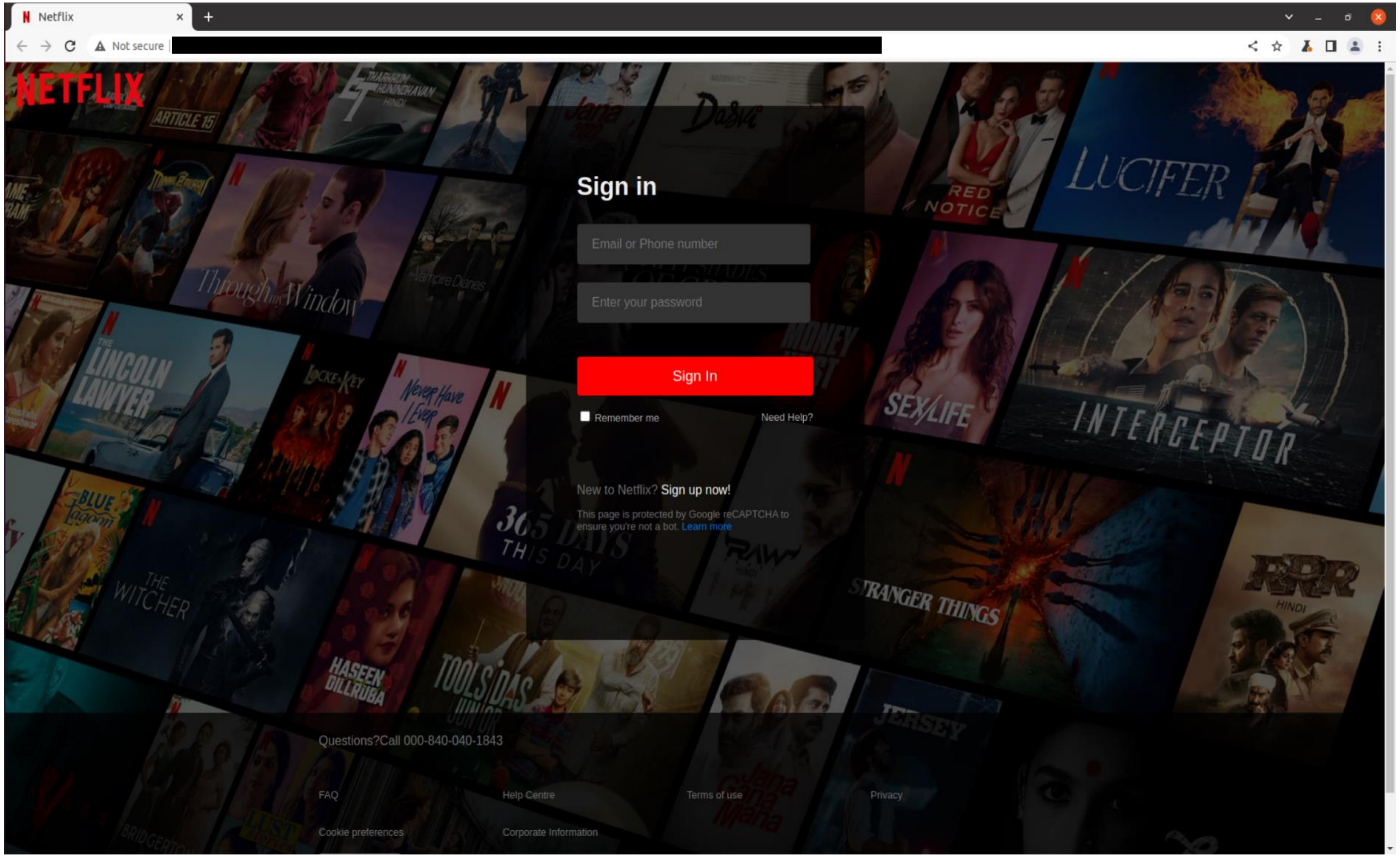


More Cases - 1

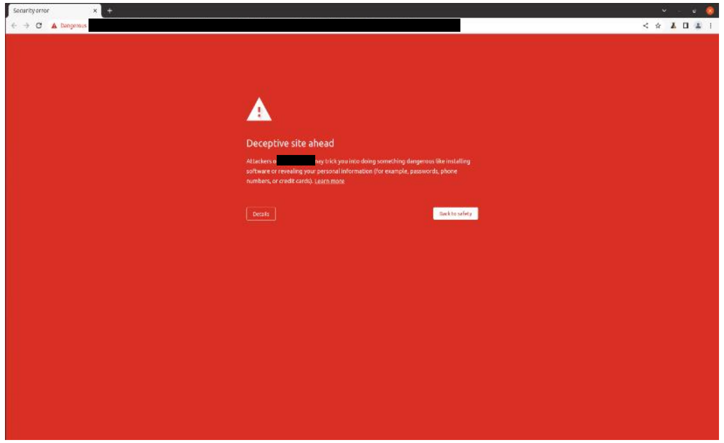


Perturbated regions.

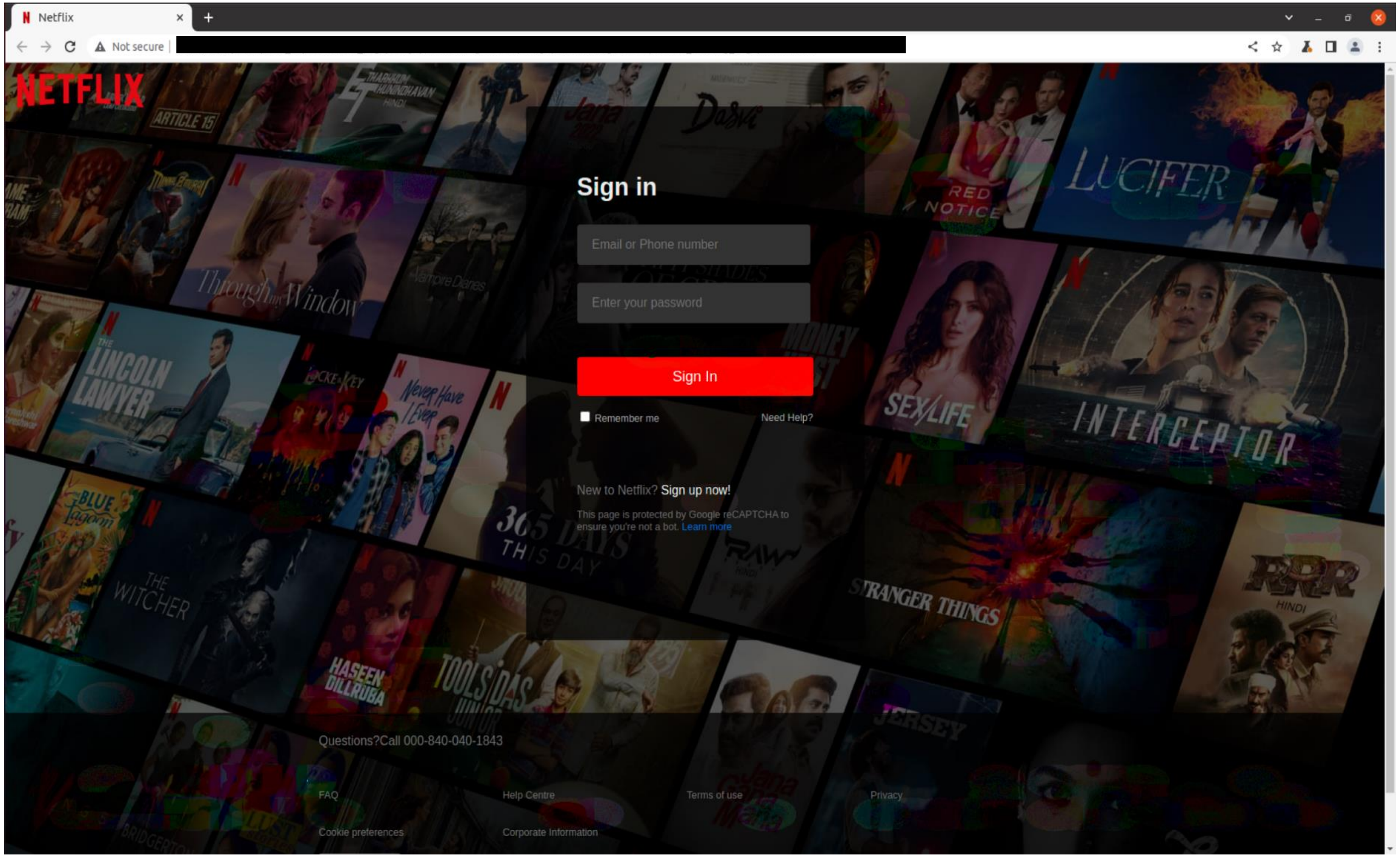
More Cases - 2



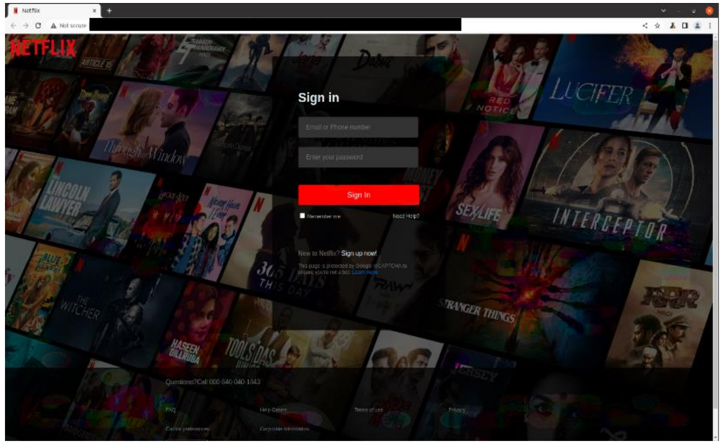
Original phishing page is blocked.



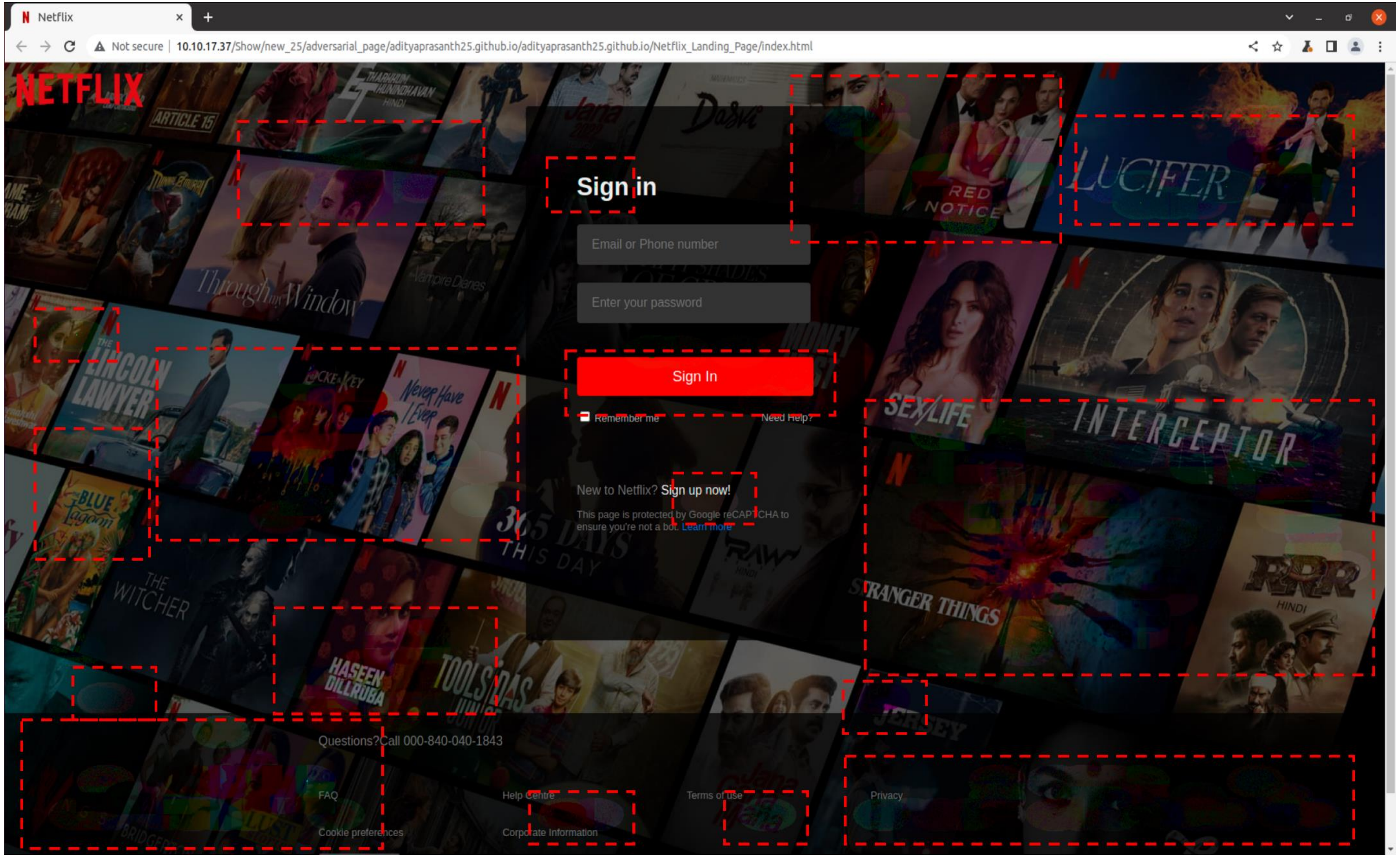
More Cases - 2



Adversarial phishing page is NOT blocked.

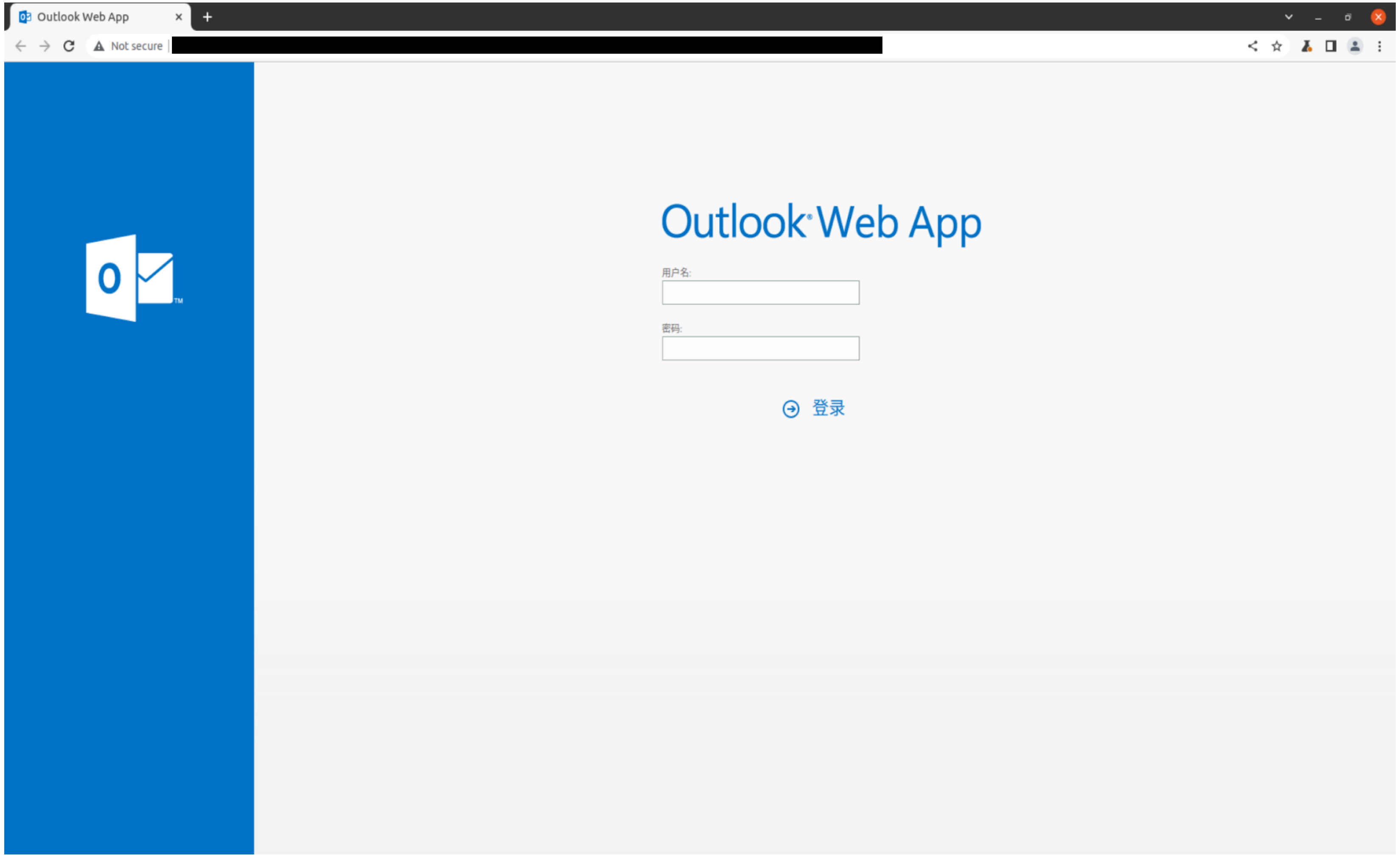


More Cases - 2

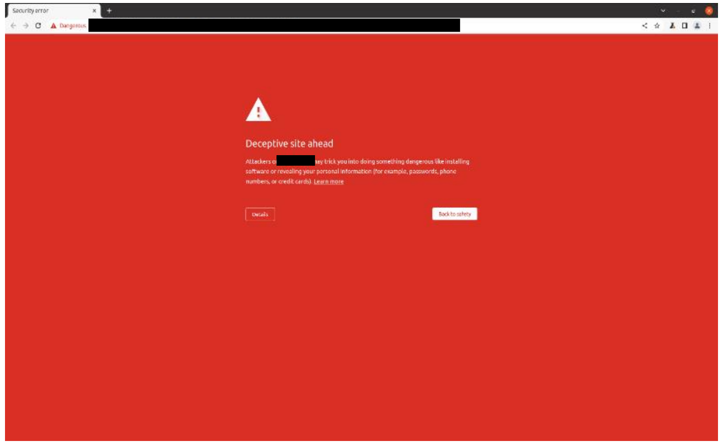


Perturbated regions.

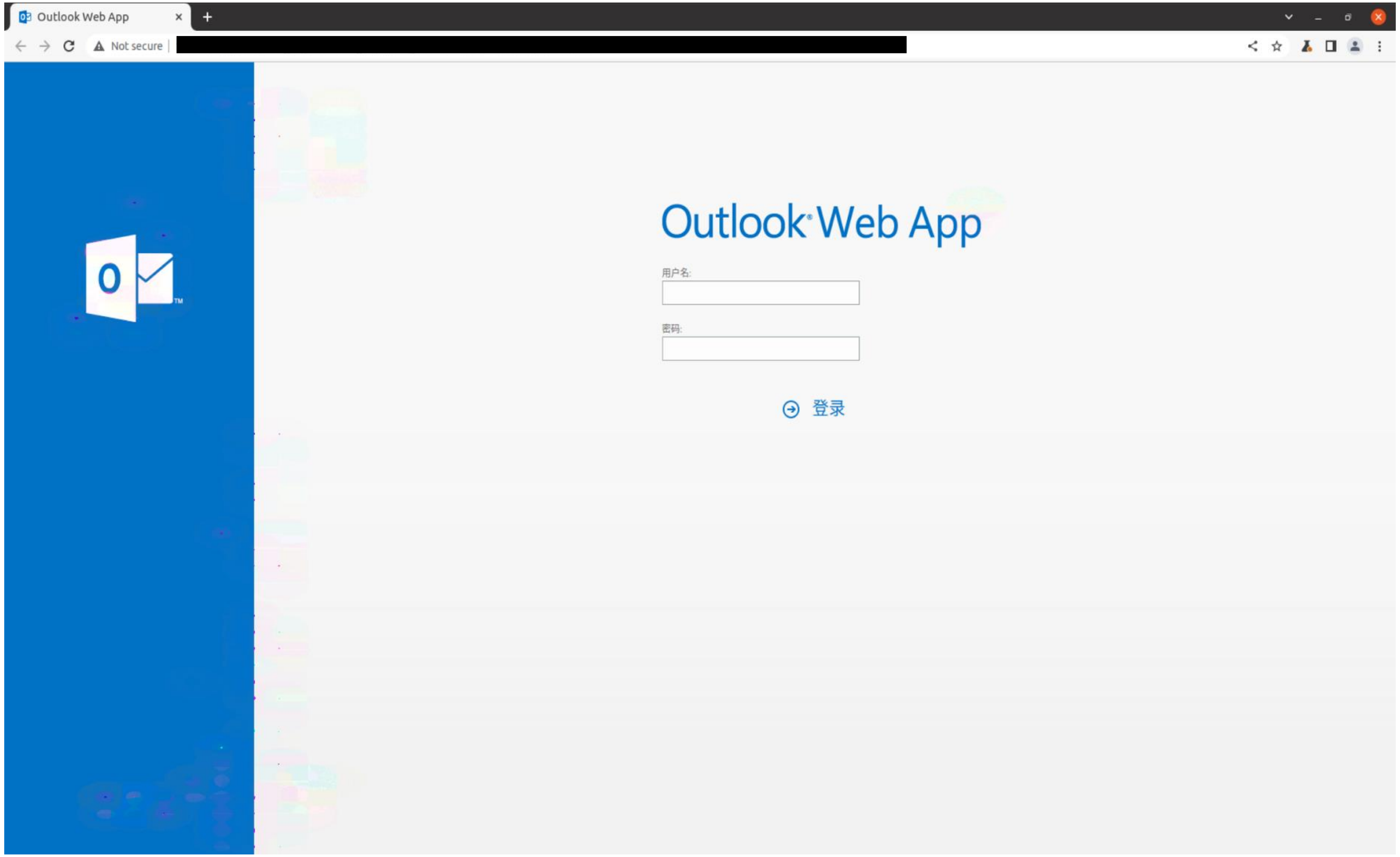
More Cases - 3



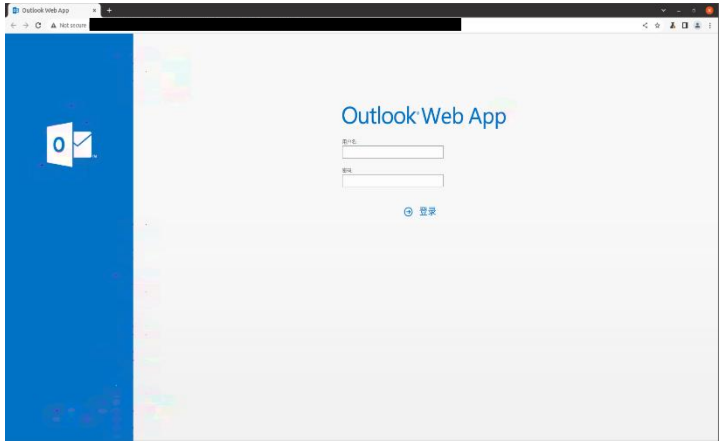
Original phishing page is blocked.



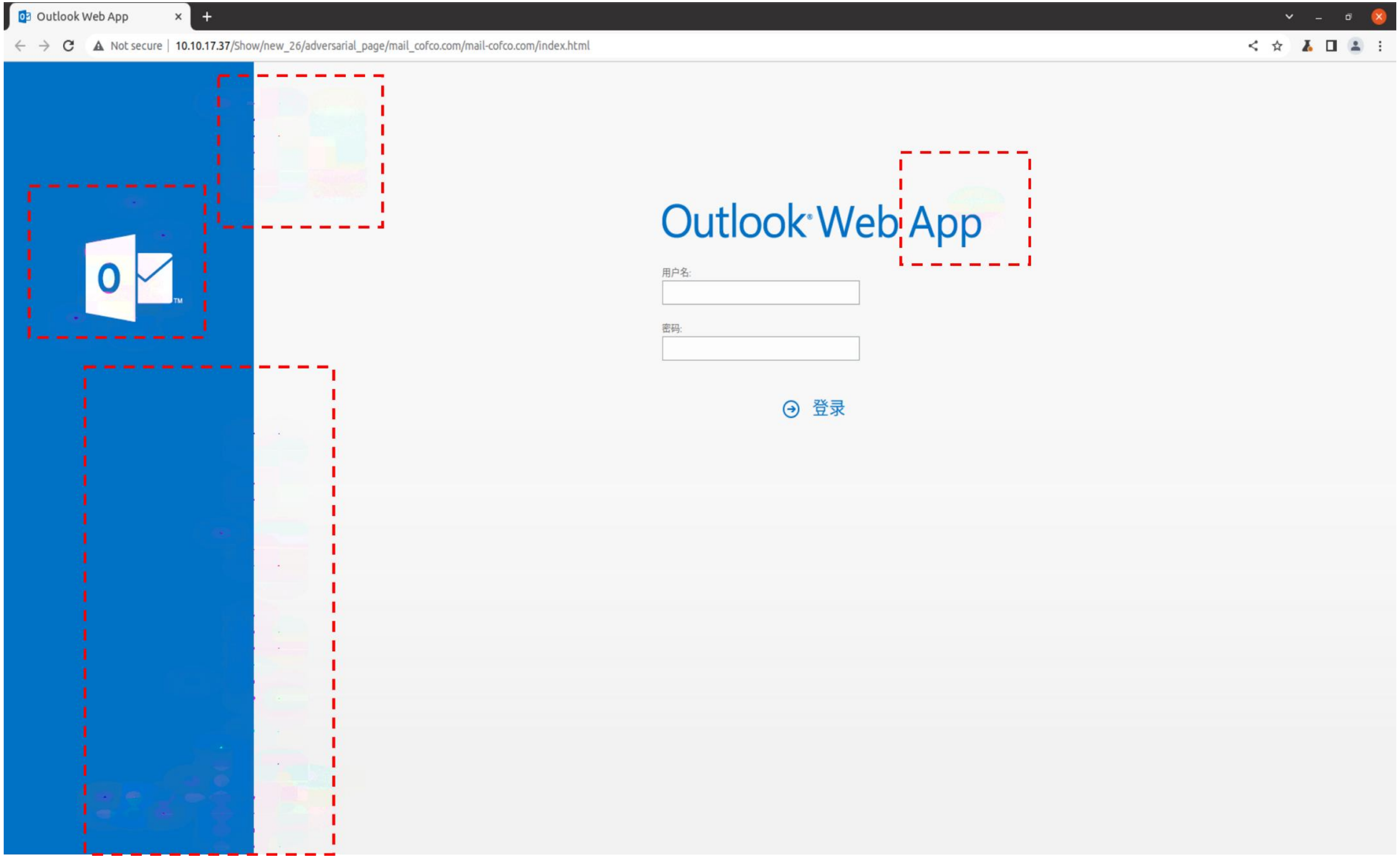
More Cases - 3



Adversarial phishing page is NOT blocked.



More Cases - 3



Perturbated regions.

Lightweight Defense against the Evasion

- **Adversarial Training**: 50 random samples as training set and the remaining 85 as testing set.
  - Fortunately, all the 85 in the testing set are able to be detected.
- **Noise Filtering**: treating perturbations as noise and applying a filtering technique.
  - 5*5 median filter and 9*9 Gaussian filter
  - 92 samples (68.1%) can be detected after the filtering.